Package 'mfrmr'

October 30, 2026

Type Package

Title Estimation and Diagnostics for Many-Facet Measurement Models

Version 0.2.4

Date 2026-10-27

Config/mfrmr/release-status candidate

Config/mfrmr/public-version 0.2.3.1

Description Estimates many-facet ordered-response measurement models with arbitrary facet counts, rating-scale and partial-credit parameterizations, a generalized partial-credit extension with discrimination varying across levels of one selected facet. Marginal maximum likelihood permits a different facet to own category steps; joint maximum likelihood requires the same facet for both roles. Conditional maximum likelihood fits many-facet rating-scale and partial-credit models by conditioning on person totals, with structural contrast intervals and no assumed ability distribution. Experimental two-family estimation and conditional scoring retain explicit uncertainty limitations. The fit, diagnose and report workflow covers anchoring, linking, bias and differential-functioning screening, with saved results for replay. Observed-score generalizability and decision studies support assessment planning under declared rating designs. Agreement, rater feedback and conditional classification summaries connect analysis to assessment decisions. See 'Andrich' (1978) <doi:10.1007/BF02293814>, 'Masters' (1982) <doi:10.1007/BF02296272>, and 'Muraki' (1992) <doi:10.1177/014662169201600206> for the underlying ordered-response models.

URL https://ryuya-dot-com.github.io/mfrmr/, https://github.com/Ryuya-dot-com/mfrmr

BugReports https://github.com/Ryuya-dot-com/mfrmr/issues

License MIT + file LICENSE

Language en

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports cli, dplyr, Matrix, grDevices, graphics, tidyr, tibble, purrr, stringr, psych, lifecycle, rlang, methods, parallel, stats, tools, utils

LinkingTo cpp11 (>= 0.5.4)

Suggests RTMB (>= 2.0), cluster, mice, nleqslv, testthat (>= 3.0.0), callr, pkgload, withr, covr, knitr, rmarkdown, igraph, ergm (>= 4.0), network, lme4, minqa, nloptr, numDeriv, ordinal, kableExtra, flextable, future, future.apply, shiny, ggplot2 (>= 3.4.0), mirt, TAM, eRm, lpSolve

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown, plotly, htmlwidgets

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Ryuya Komuro [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9205-0926>)

Maintainer Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp>

Repository CRAN

Date/Publication 2026-10-30 13:40:02 UTC

Contents

mfrmr-package 7
analyze_dff 21
analyze_facet_equivalence 28
analyze_hierarchical_structure 31
analyze_residual_pca 35
anchor_to_baseline 41
apa_table 45
apply_empirical_bayes_shrinkage 50
as.data.frame.mfrm_fit 52
assess_mfrm_recovery 54
as_flextable 58
as_flextable.apa_table 59
as_ggplot 60
as_kable 65
as_kable.apa_table 66
automated_scoring_table 67
bias_count_table 68
bias_interaction_report 70
bias_iteration_report 73
bias_pairwise_report 74

bootstrap_mfrm_gpcm . . . 78
build_apa_outputs . . . 81
build_conquest_overlap_bundle . . . 86
build_equating_chain . . . 90
build_fixed_reports . . . 96
build_linking_review . . . 99
build_mfrm_manifest . . . 101
build_mfrm_network_review . . . 104
build_mfrm_replay_script . . . 107
build_mfrm_resampling_spec . . . 110
build_mfrm_sim_spec . . . 113
build_misfit_casebook . . . 119
build_model_choice_review . . . 121
build_peer_review_sim_spec . . . 123
build_summary_table_bundle . . . 125
build_visual_summaries . . . 132
build_weighting_review . . . 135
category_curves_report . . . 138
category_structure_report . . . 142
classification_table . . . 144
compare_mfrm . . . 147
compatibility_alias_table . . . 156
compute_facet_design_effect . . . 158
compute_facet_icc . . . 160
compute_information . . . 164
compute_person_fit_indices . . . 169
confint.mfrm_fair_average . . . 172
confint.mfrm_fit . . . 180
confint.mfrm_multivariate_d_study . . . 188
confint.mfrm_multivariate_gstudy . . . 212
confint.mfrm_random_rater . . . 221
confint.mfrm_testlet . . . 223
data_quality_report . . . 224
describe_mfrm_data . . . 226
detect_anchor_drift . . . 236
detect_facet_nesting . . . 241
diagnose_mfrm . . . 244
dif_interaction_table . . . 253
dif_report . . . 256
displacement_table . . . 258
draw_mfrm_resamples . . . 261
ej2021_data . . . 262
estimate_all_bias . . . 264
estimate_bias . . . 270
estimation_iteration_report . . . 279
evaluate_mfrm_design . . . 280
evaluate_mfrm_diagnostic_screening . . . 287
evaluate_mfrm_signal_detection . . . 291

export_mfrm_bundle . . . 298
export_mfrm_model . . . 303
export_mfrm_results . . . 306
export_mfrm_tables . . . 309
export_summary_appendix . . . 312
extract_mfrm_sim_spec . . . 315
facets_chisq_table . . . 317
facets_feature_coverage . . . 319
facets_fit_df_guide . . . 322
facets_fit_review . . . 323
facets_output_contract_review . . . 325
facets_output_file_bundle . . . 327
facets_positioning_guide . . . 330
facets_term_crosswalk . . . 331
facets_visual_contract . . . 332
facet_quality_dashboard . . . 332
facet_small_sample_review . . . 336
facet_statistics_report . . . 339
fair_average_table . . . 341
fit_measures_table . . . 347
fit_mfrm . . . 350
fit_mfrm_imputed . . . 391
fit_mfrm_random_rater . . . 393
fit_mfrm_testlet . . . 399
fit_rater_accuracy . . . 405
fit_rater_network . . . 407
gpcm_capability_matrix . . . 410
gwet_ac2_table . . . 418
import_erm_fit . . . 420
import_mirt_fit . . . 421
import_tam_fit . . . 423
interaction_effect_table . . . 425
interrater_agreement_table . . . 427
krippendorff_alpha_table . . . 431
launch_mfrmr_viewer . . . 433
list_mfrmr_data . . . 435
load_mfrmr_data . . . 436
make_anchor_table . . . 438
mantel_haenszel_table . . . 440
measurable_summary_table . . . 443
mfrmr_agreement_data . . . 445
mfrmr_compatibility_layer . . . 446
mfrmr_example_data . . . 457
mfrmr_example_operational_design . . . 459
mfrmr_gtheory_data . . . 460
mfrmr_interval_guide . . . 461
mfrmr_linking_and_dff . . . 464
mfrmr_methods_reference . . . 474

mfrmr_model_maps . . . 484
mfrmr_output_guide . . . 486
mfrmr_plot_migration . . . 490
mfrmr_reporting_and_apa . . . 492
mfrmr_reports_and_tables . . . 498
mfrmr_visual_diagnostics . . . 503
mfrmr_workflow_methods . . . 514
mfrm_calibration_capabilities . . . 532
mfrm_calibration_methods . . . 533
mfrm_calibration_score_methods . . . 534
mfrm_calibration_workflow . . . 537
mfrm_cluster . . . 544
mfrm_cluster_compare . . . 546
mfrm_cluster_hierarchical . . . 549
mfrm_cluster_imputed . . . 552
mfrm_cluster_kmeans . . . 555
mfrm_curve_intervals . . . 558
mfrm_dff_methods . . . 561
mfrm_d_study . . . 562
mfrm_d_study_main_effects . . . 569
mfrm_facet_intervals . . . 572
mfrm_features . . . 586
mfrm_fit_extractors . . . 588
mfrm_fit_responses . . . 591
mfrm_generalizability . . . 594
mfrm_g_study . . . 597
mfrm_misfit_thresholds . . . 605
mfrm_multivariate_d_compare . . . 606
mfrm_network_analysis . . . 611
mfrm_pca . . . 614
mfrm_population_intervals . . . 617
mfrm_random_rater_intervals . . . 623
mfrm_report . . . 627
mfrm_residual_network . . . 635
mfrm_response_diagnostics . . . 639
mfrm_response_imputations . . . 644
mfrm_results . . . 647
mfrm_results_interactive . . . 655
mfrm_screening_performance . . . 657
mfrm_screening_sensitivity . . . 659
mfrm_threshold_profiles . . . 663
mml_quadrature_sensitivity . . . 664
normalize_conquest_overlap_exports . . . 667
normalize_conquest_overlap_files . . . 669
normalize_conquest_overlap_tables . . . 672
ordinal_logistic_table . . . 675
pairwise_icc_table . . . 678
pairwise_kappa_table . . . 680

plot.mfrm_bundle . . . 682
plot.mfrm_clusters . . . 685
plot.mfrm_design_evaluation . . . 689
plot.mfrm_diagnostic_screening . . . 690
plot.mfrm_dif . . . 692
plot.mfrm_dstudy_intervals . . . 694
plot.mfrm_d_study . . . 696
plot.mfrm_facet_nesting . . . 699
plot.mfrm_fit . . . 700
plot.mfrm_multivariate_d_study . . . 709
plot.mfrm_multivariate_gstudy . . . 715
plot.mfrm_recovery_simulation . . . 719
plot.mfrm_slope_intervals . . . 721
plot_data . . . 723
plot_data_components . . . 724
pool_mfrm_imputed . . . 725
precision_review_report . . . 729
predict.mfrm_random_rater . . . 731
predict.mfrm_testlet . . . 733
predict_mfrm_population . . . 737
predict_mfrm_units . . . 741
print.mfrm_apa_text . . . 750
q3_statistic . . . 751
rater_centrality_table . . . 754
rater_halo_network_analysis . . . 757
rater_network_analysis . . . 761
rating_scale_table . . . 765
read_facets_fit_table . . . 770
read_facets_results . . . 772
read_mfrm_posterior . . . 777
recode_missing_codes . . . 781
recommend_mfrm_design . . . 782
reference_case_benchmark . . . 787
reference_case_review . . . 789
reporting_checklist . . . 791
response_time_review . . . 795
review_accessors . . . 797
review_conquest_overlap . . . 798
review_mfrm_anchors . . . 802
review_peer_design . . . 808
run_mfrm_facets . . . 809
run_qc_pipeline . . . 814
sample_mfrm_plausible_values . . . 819
score_comparison_table . . . 823
score_mfrm_persons . . . 826
score_mfrm_random_rater . . . 829
shrinkage_report . . . 833
simulate_mfrm_data . . . 834

simulate_mfrm_recovery 839
specifications_report 845
subset_connectivity_report 847
summary.mfrm_bundle 849
summary.mfrm_design_evaluation 851
summary.mfrm_diagnostics 853
summary.mfrm_diagnostic_screening 856
summary.mfrm_facets_run 858
summary.mfrm_fit 859
summary.mfrm_jeffreys_fit 865
summary.mfrm_multivariate_d_study 866
summary.mfrm_population_prediction 869
summary.mfrm_signal_detection 871
unexpected_after_bias_table 872
unexpected_response_table 874
vcov.mfrm_multivariate_gstudy 877
visual_reporting_template 880
write_mfrm_residual_file 881
write_mfrm_subset_file 883

Index **885**

`mfrmr-package` *mfrmr: Many-Facet Ordered-Response Modeling in R*

Description

`mfrmr` provides estimation, diagnostics, and reporting utilities for many-facet ordered-response measurement models: the Rasch-family `RSM` / `PCM` route and a `GPCM` extension in which one selected facet supplies level-specific discriminations. MML permits a different facet to supply category steps; JML requires the same facet for both roles. `confint.mfrm_fit()` with `parm = "slopes"` supplies approximate relative-slope intervals for eligible MML fits; model ranking and matched PCM/GPCM tests use separate checks in `compare_mfrm()`. `gpcm_capability_matrix()` explains the supported uses. An experimental two-family `GPCM` route also provides estimation and conditional scoring, with explicit uncertainty limitations. Native RSM/PCM conditional maximum likelihood (`CML`) conditions on Person totals without assuming an ability distribution. Its separate result class supplies structural contrast intervals through `mfrm_facet_intervals()`, without Person abilities, a population SD or ordinary fit diagnostics. See `fit_mfrm()` for the supported conditional estimation scope.

For assessment planning from observed scores, `mfrm_g_study()` and `mfrm_d_study()` separate sources of variation and project dependability for specified future rating plans. For rater feedback and assessment decisions, `mfrm_report()`, `interrater_agreement_table()` and `classification_table()` provide their respective reports, agreement measures and fixed-plan classification summaries. These analyses have different assumptions and uncertainty targets; follow the linked help for the question you want to answer.

Details

Start with the complete script in the Examples section:

1. Load the package and example ratings with `load_mfrmr_data()`.
2. Fit a model with `fit_mfrm()`.
3. Draw the Wright map with `plot(fit)`.
4. Save `results <- summary(fit)` and inspect `results$person_overview` and `results$facet_overview`.

Each data row is one rating event. The `person`, `facets`, and `score` arguments name columns. The example uses marginal maximum likelihood (`MML`) and a rating-scale model (`RSM`) with shared category thresholds. `head(ratings)` displays the first six rows. Only `Person`, `Rater`, `Criterion`, and `Score` are used by this model; the example's `Study` and `Group` columns are additional labels. `<-` saves an object, and `$` selects a named part: `results$person_overview` displays one table from the saved summary. The overview tables summarize distributions; `as.data.frame(fit)` returns individual person, rater, and criterion estimates. The assignment `ratings <- load_mfrmr_data("example_operational")` reads an existing synthetic dataset supplied with mfrmr. You do not need to create `ratings` first; the name on the left stores what the function returns.

Before interpreting or reporting estimates, read `results$decision` and follow its `NextAction`. The default summary does not compute diagnostics. For your own data, first inspect the design and score categories with `describe_mfrm_data()`.

Optional packages

The basic example below uses packages installed with mfrmr. Its base-R plots do not need ggplot2. Additional packages support particular features; you do not need to install all suggested packages to get started.

- `as_ggplot()` needs ggplot2 for ggplot2 rendering.
- `as_kable()` needs knitr and `as_flextable()` needs flextable for their respective table formats.
- `fit_mfrm_random_rater()` needs RTMB version 2.0 or later for shared random-rater estimation.
- Corrected JML in `fit_mfrm()`, requested with `jml_correction_order`, needs nleqslv to solve the adjusted equations.
- `ordinal_logistic_table()` needs ordinal for ordinal-logistic DIF.

These are common examples; other optional analyses name their dependencies in their own help. A missing-package message identifies what to install. For example, use `install.packages("ggplot2")` if you want `as_ggplot()`.

A saved `mfrm_results()` bundle can be reopened with `base::readRDS()` and passed to `mfrm_report()` without refitting or installing optional estimation packages. Reporting reuses stored results and retains unavailable diagnostics as unavailable. Updating an analysis or computing new diagnostics is a separate operation and may need the original dependencies. Save the complete bundle with `base::saveRDS()` and retain its matching package version/source; see "Updating saved analyses for 0.2.4" in mfrmr_workflow_methods before reusing older saved analyses.

Measurement scale

Native response models use the logistic metric (L-metric), with scaling constant $D = 1$; no extra 1.7 normal-ogive approximation factor is used. This is distinct from choosing a normal ability distribution. See the "Logistic metric (D = 1)" section of `fit_mfrm()` for logit interpretation with fixed or varying slopes and comparison with other software.

Where to go next

- mfrmr_methods_reference for the question-to-calculation map, plain-language explanations and links to the methodological literature.
- mfrmr_workflow_methods and `vignette("mfrmr-workflow", package = "mfrmr")` for your own CSV, column mapping, input troubleshooting, and the full analysis workflow. If vignettes are not installed, start with `describe_mfrm_data()` and `recode_missing_codes()`.
- mfrmr_visual_diagnostics for choosing follow-up figures.
- mfrmr_reporting_and_apa and mfrmr_reports_and_tables for reporting.
- mfrmr_linking_and_dff for linking and differential facet functioning.
- `classification_table()` for fixed-plan future classification accuracy, consistency and conditional precision under ordinary RSM/PCM calibration.
- gpcm_capability_matrix for the `GPCM` extension.
- `mfrmr_output_guide()` for the broader purpose-to-function map.

A printable reference card is available at `system.file("cheatsheet", "mfrmr-cheatsheet.pdf", package = "mfrmr")`.

Portable fixed calibration

A saved, versioned calibration artifact can be created from an eligible one-scale `RSM` or `PCM` MML fit under the fixed standard-normal scoring basis. Use `mfrm_calibration_capabilities()` before extraction, then follow mfrm_calibration_workflow to review, validate, freeze, save, load, and score the artifact. Review the returned batch with mfrm_calibration_score_methods before using its estimates. One-family GPCM MML also supports an estimated intercept-only normal population, shared or separate slope/step owners, unit weights and no anchors/interactions, subject to conditional source and per-batch numerical scoring checks. Estimated- population RSM/PCM and latent-regression MML remain fitted-object-only. RSM/PCM JML supports portable post-hoc EAP with a standard-normal reference prior, unit weights, no anchors/interactions and passing finite identified source checks. Shared-owner GPCM JML also supports conditional portable EAP after its joint-likelihood/curvature checks, preserving incomplete global audits. Explicit corrected GPCM JML instead uses adjusted-equation checks and preserves the correction order in format 5; residual calibration bias may remain. That reference prior is not estimated by JML. Experimental two-family GPCM MML uses format 6 with both ordered slope owners and a fixed N(0,1) prior, separate source/batch checks and no prior overrides. Artifact score uncertainty is conditional on the frozen point calibration and its recorded prior; loading validates consistency but does not authenticate an untrusted file.

Advanced scope

After the basic route above:

- the package supports latent-regression MML for ordered-response RSM / PCM models with a one-dimensional conditional-normal population model and explicit one-row-per-person covariates expanded through stats::model.matrix()
- GPCM support is summarized by gpcm_capability_matrix()
- GPCM supports the core fit/summary/scoring/information path, direct Wright/pathway/CCC plots, residual-PCA follow-up, and the residual-based diagnostics tables/plots as exploratory tools
- posterior-predictive checks and MCMC estimation are not available for GPCM; use external Bayesian software when they are required
- direct GPCM data generation through build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data() is available when the specification carries both thresholds and slopes with the same owner; general simulation/design workflows do not support separate owners
- slope-aware fair_average_table() and estimate_bias() are available for GPCM with explicit caveats; build_apa_outputs(), build_visual_summaries(), run_qc_pipeline(), build_mfrm_manifest(), build_mfrm_replay_script(), and export_mfrm_bundle() are available as caveated partial reporting/export surfaces; score-side FACETS compatibility remains limited, and fully arbitrary-facet planning is outside the documented planning scope
- predict_mfrm_population() remains a scenario-level forecast helper and should not be described as the latent-regression estimator itself
- the current simulation/planning layer remains role-based for two non-person facets rather than fully arbitrary-facet planning, with boundaries exposed through planner metadata such as planning_scope, planning_constraints, and planning_schema
- latent-class mixture models and response-time / careless-rating adjustment are not estimated by mfrmr; use residual, person-fit, local-dependence, and rater-drift diagnostics as screening layers rather than as mixture-model substitutes

Equal weighting versus GPCM

The package's operational reference route is the Rasch-family RSM / PCM branch. That route enforces fixed discrimination and therefore preserves an equal-weighting scoring interpretation across observed ratings.

GPCM is supported because some users want a slope-aware model- comparison or sensitivity layer inside the same many-facet workflow. However, the package does not treat GPCM as a universal replacement for the Rasch-family route. A better fit under GPCM should be read as evidence about discrimination-based reweighting, not as an automatic reason to discard the equal-weighting model.

Observation weights are a different concept again. Optional Weight columns change how observed rating events enter estimation and summaries, but they do not create a free-form facet-weighting scheme and do not alter the fixed-discrimination meaning of RSM / PCM.

Public entry map:

- First-screen results: mfrm_results(), summary(res)$next_actions, and mfrmr_output_guide("entry")
- Interactive exploration: mfrm_results_interactive() only when prompts are explicitly wanted at the console

Function families:

- Model fitting: fit_mfrm(), summary.mfrm_fit(), plot.mfrm_fit()
- Legacy-compatible workflow wrapper: run_mfrm_facets(), mfrmRFacets()
- Diagnostics: diagnose_mfrm(), summary(diag), analyze_residual_pca(), plot(x, type = "residual_pca")
- Bias and interaction: estimate_bias(), estimate_all_bias(), summary(bias), bias_interaction_report(), plot(x, type = "bias")
- Differential functioning: analyze_dff(), analyze_dif(), mantel_haenszel_table() for separately matched binary responses, ordinal_logistic_table() for matched cumulative-logit comparisons, dif_interaction_table(), plot(x, type = "heatmap"), dif_report()
- Design simulation: build_mfrm_sim_spec(), extract_mfrm_sim_spec(), simulate_mfrm_data(), simulate_mfrm_recovery(), assess_mfrm_recovery(), evaluate_mfrm_design(), evaluate_mfrm_signal_det predict_mfrm_population(), predict_mfrm_units(), sample_mfrm_plausible_values() (including fit-derived empirical / resampled / skeleton-based simulation specifications; fitted-object posterior scoring supports MML fits directly, latent-regression MML fits through the fitted population model when scored units also provide one-row-per-person background data, and JML fits through a post hoc reference-prior EAP layer; fit-derived simulation specifications also support direct GPCM data generation, recovery checks, role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers with documented caveats; curve reports, graph-only exports, fair-average tables, and bias screening are also available for GPCM with documented caveats)
- Reporting: build_apa_outputs(), build_visual_summaries(), reporting_checklist(), apa_table() for the full RSM / PCM route; GPCM uses these as caveated partial surfaces and retains direct table, plot, checklist, and summary-appendix routes
- Weighting review: compare_mfrm(), build_weighting_review(), build_model_choice_review(), compute_information(), plot(x)
- Case review: build_misfit_casebook(), plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise")
- Linking and scale maintenance: review_mfrm_anchors(), detect_anchor_drift(), build_equating_chain(), build_linking_review(), plot(x)
- Dashboards: facet_quality_dashboard(), plot(x, type = "facet_dashboard")
- Export / reproducibility: build_mfrm_manifest(), build_mfrm_replay_script(), build_conquest_overlap_bun normalize_conquest_overlap_exports(), normalize_conquest_overlap_files(), normalize_conquest_over review_conquest_overlap(), export_mfrm_bundle() for the diagnostics-compatible Rasch-family route; GPCM supports caveated partial manifest, replay, and bundle output as well as export_summary_appendix()
- Equivalence: analyze_facet_equivalence(), plot(x, type = "facet_equivalence")
- Data and anchors: describe_mfrm_data(), review_mfrm_anchors(), make_anchor_table(), load_mfrmr_data()

Data interface:

- Input analysis data is long format (one row per observed rating).
- Required columns are one person column, one ordered score column, and one or more non-person facet columns named in facets = c(...).

- Score values should be ordered integer categories. Binary `0/1` or `1/2` input is supported as the two-category Rasch-family special case; by contrast, fractional score values should be recoded before fitting rather than relying on automatic coercion.
- If `keep_original = FALSE`, unused intermediate categories are collapsed to a contiguous internal scale and the mapping is stored in `fit$prep$score_map`.
- If the intended scale has unused boundary categories, such as a 1-5 scale with only 2-5 observed, set `rating_min = 1, rating_max = 5` so the zero-count boundary category remains explicit in the data-support review. A missing boundary remains review evidence for the separate element- boundary contract. If unused intermediate categories should remain in the original scale, set `keep_original = TRUE`; a retained zero-count internal category in a polytomous fitted ladder creates an unsupported adjacent- step contrast, and `fit_mfrm()` returns a structured pre-optimization error.
- `summary(describe_mfrm_data(...))` reports retained zero-count categories in `Notes`, printed `Caveats`, and `$caveats`; `summary(fit)` carries full structured rows into printed `Caveats` and `$caveats`, with `Key warnings` as a short triage subset. Summary-table exports route those rows through `score_category_caveats` or `analysis_caveats`.
- A category can be observed globally but unused by one rater or other facet level. Review that local pattern with `data_quality_report()` rather than treating it as a globally missing step or an automatic reason to select `GPCM`.
- Optional columns such as `Subset`, `Weight`, and `Group` support linking, weighted analysis, and fairness-focused follow-up workflows.
- Packaged synthetic data is available via `load_mfrmr_data()` or `data()`. Use `list_mfrmr_data(details = TRUE)` to distinguish the applied teaching, idealized, planted-effect, and larger sparse-design datasets.

Interpreting output

Core object classes are:

- `mfrm_fit`: fitted model parameters and metadata.
- `mfrm_diagnostics`: fit, facet-level reliability, and flag diagnostics, plus inter-rater agreement when one facet is treated as a rater facet.
- `mfrm_bias`: interaction bias estimates.
- `mfrm_dff` / `mfrm_dif`: differential-functioning contrasts and screening summaries.
- `mfrm_population_prediction`: scenario-level forecast summaries for one future design.
- `mfrm_unit_prediction`: posterior summaries for future or partially observed persons under the fitted scoring basis.
- `mfrm_plausible_values`: posterior draws for future or partially observed persons under the fitted scoring basis.
- `mfrm_bundle` families: summary/report bundles and draw-free plot data.

Typical workflow

1. Prepare long-format data.
2. Fit with `fit_mfrm()`.

3. For RSM / PCM, diagnose with diagnose_mfrm() and prefer diagnostic_mode = "both" for final MML runs.
4. For RSM / PCM, run analyze_dff() or estimate_bias() when fairness or interaction questions matter; GPCM also supports estimate_bias() as a conditional screening review.
5. For RSM / PCM, report with build_apa_outputs() and build_visual_summaries().
6. For design planning, move to build_mfrm_sim_spec(), evaluate_mfrm_design(), mfrm_generalizability(), mfrm_d_study(), and predict_mfrm_population(). GPCM also supports direct simulation via extract_mfrm_sim_spec() / simulate_mfrm_data() and caveated role-based evaluate_mfrm_design() / predict_mfrm_population() routes. Those helpers assume two non-person facet roles even though the estimation core supports arbitrary facet counts. Treat evaluate_mfrm_design() as Monte Carlo design evaluation, and use mfrm_d_study() for analytic G/Phi design projections. Always read the IdentificationStatus, GStatus, and PhiStatus columns before reporting those projections; boundary or singular mixed-model fits are design-identification warnings, not high-stakes-ready reliability evidence. predict_mfrm_population() remains the scenario-level forecast helper, not the latent-regression estimator.
7. For future-unit scoring, retain an MML calibration when you want the fitted marginal model directly, use an active latent-regression MML fit when scored units also provide one-row-per-person background data, or use a JML calibration when a post hoc fitted-object EAP layer is acceptable; then score with predict_mfrm_units() or sample_mfrm_plausible_values().
8. For GPCM, use summary.mfrm_fit(), diagnose_mfrm(), analyze_residual_pca(), predict_mfrm_units(), sample_mfrm_plausible_values(), compute_information(), plot(x, type = "qc"), plot.mfrm_fit(), category_structure_report(), category_curves_report(), graph-only facets_output_file_bundle(), direct simulation-spec generation/data generation, recovery checks, fair_average_table(), estimate_bias(), and the residual-based table helpers with their documented caveats. Caveated APA/QC/export bundles and exploratory linking review plus role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers are available, while full score-side FACETS review, posterior-predictive checks, and MCMC estimation are not available for GPCM. Use gpcm_capability_matrix() as the formal boundary statement.

Model formulation

The Rasch-family branch used by RSM and PCM extends the basic Rasch model by incorporating multiple measurement facets into a single additive linear predictor on the log-odds scale.

RSM/PCM adjacent-category equation

For an observation where person n with ability θ_n is rated by rater j with severity δ_j on criterion i with difficulty β_i, the probability of observing category k (out of K ordered categories) is:

$$P(X_{nij} = k \mid \theta_n, \delta_j, \beta_i, \tau) = \frac{\exp\left[\sum_{s=1}^{k}(\theta_n - \delta_j - \beta_i - \tau_s)\right]}{\sum_{c=0}^{K}\exp\left[\sum_{s=1}^{c}(\theta_n - \delta_j - \beta_i - \tau_s)\right]}$$

where τ_s are the Rasch-Andrich threshold (step) parameters in the RSM reference case and $\sum_{s=1}^{0}(\cdot) \equiv 0$ by convention. Additional facets enter as additive terms in the linear predictor $\eta = \theta_n - \delta_j - \beta_i - \ldots$.

This additive predictor generalises to any number of facets; the facets argument to fit_mfrm() accepts an arbitrary-length character vector.

Rating Scale Model (RSM)

Under the RSM (Andrich, 1978), all levels of the step facet share a single set of threshold parameters $\tau_1, \ldots, \tau_K$.

Partial Credit Model (PCM)

Under the PCM (Masters, 1982), each level of the designated `step_facet` has its own threshold vector on the package's common observed score scale. In the current implementation, threshold locations may vary by step-facet level, but the fitted score range is defined by one global category set taken from the observed data.

Generalized Partial Credit Model (GPCM)

With one slope family under `GPCM` (Muraki, 1992), the adjacent-category partial-credit kernel is multiplied by a positive slope α_g for the designated slope-facet level g:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k-1)} = \alpha_g(\theta_n - \delta_j - \beta_i - \tau_{hk}).$$

MML allows a slope owner g distinct from the step owner h; JML requires `slope_facet == step_facet`. Slopes are identified on the log scale with geometric mean 1. This makes `GPCM` a slope-aware sensitivity/extension route, not a replacement for the equal-weighting `RSM`/`PCM` interpretation. A separate provisional two-family MML route fits the multiplicative Uto–Ueno equation with explicit fixed-N(0,1) identification, exactly two facets and second-owner steps. The first family's slopes have geometric mean one and the second family's slopes are free. Fixed-grid EM and adaptive direct MML supply summaries and conditional fitted curves without intervals. Both support separately checked experimental component log-Wald and location/contrast intervals and descriptive posterior residuals; component profiles preserve the fitted fixed-grid or adaptive integration and require separate numerical checks. `predict_mfrm_units()` and the portable calibration workflow supply experimental conditional new-Person EAP and posterior intervals, excluding calibration uncertainty. Coverage and population transport remain unqualified. The one-family diagnostic and inference routes below do not apply to it. See the Two slope families section of `fit_mfrm()`. Unit slopes reduce to the equal-discrimination PCM kernel. Under default one-family MML, an intercept-only person distribution $N(\beta_0, \sigma^2)$ is estimated. The geometric-mean-one slopes are relative discriminations and $\sigma\alpha_g$ are their equivalent fixed-latent-standard-deviation values, so the conventional common discrimination degree of freedom is retained. The legacy `gpcm_mml_identification = "fixed_standard_normal"` mode fixes both $\sigma = 1$ and the slope geometric mean to one and is therefore a narrower relative-discrimination model. Under JML, geometric-mean-one is required to resolve the scale of jointly estimated person coordinates. The former "bounded" label described limited model/workflow scope, not finite optimizer box constraints. Use GPCM with explicit scope restrictions. The GPCM JML objective is unpenalized. Certified extreme-person or facet recession is reported through typed primary boundary results; a finite optimizer iterate is retained only as a numerical trace and is not a finite JML maximum.

Ordered-response scope

The implemented response-model scope is ordered categorical only. Binary responses are the $K = 1$ special case of the same formulation, so they are handled through the ordinary ordered-score interface. This means `mfrmr` supports ordered binary and ordered polytomous data under `RSM` and `PCM`, plus `GPCM` with one designated `slope_facet` and, under MML, a possibly different `step_facet`. Unordered nominal/multinomial response models are outside the documented model scope, as are Poisson, negative-binomial, and grouped binomial-trial count-response families. A

positive observation `weight` weights the conditional ordered-rating contribution; it is not a general collapsed-person frequency table and does not change the response family or model dependence among replicated ratings. Under MML, powering conditional responses within one Person is not the same as replicating a complete Person pattern after marginalization. Consequently, integer count values supplied as scores are modeled as ordered category codes, not as counts from a Poisson or related distribution.

Estimation methods

Marginal Maximum Likelihood (MML)

MML integrates over the person ability distribution using Gauss-Hermite quadrature, in the broader marginal-likelihood framework introduced by Bock & Aitkin (1981) for IRT:

$$L = \prod_n \int P(\mathbf{X}_n \mid \theta, \boldsymbol{\delta})\, \phi(\theta)\, d\theta \approx \prod_n \sum_{q=1}^{Q} w_q\, P(\mathbf{X}_n \mid \theta_q, \boldsymbol{\delta})$$

where $\phi(\theta)$ is the assumed normal prior and (θ_q, w_q) are quadrature nodes and weights. Person estimates are obtained post-hoc via Expected A Posteriori (EAP):

$$\hat{\theta}_n^{\mathrm{EAP}} = \frac{\sum_q \theta_q\, w_q\, L(\mathbf{X}_n \mid \theta_q)}{\sum_q w_q\, L(\mathbf{X}_n \mid \theta_q)}$$

MML does not estimate one fixed person parameter per respondent and therefore avoids that JML incidental-parameter mechanism. Its structural inference is nevertheless conditional on the specified population distribution, response model, quadrature approximation, and regularity conditions; it is not an automatic small-sample guarantee.

Note: Bock & Aitkin (1981) is the canonical citation for the Gauss-Hermite-quadrature MML *framework*. The default mfrmr engine (`mml_engine = "direct"`) optimises this marginal log-likelihood by direct gradient methods (BFGS / L-BFGS-B), not by Bock & Aitkin's signature EM algorithm. The `"em"` and `"hybrid"` engines do follow the EM template but use the selected gradient-based M-step rather than B&A's probit IRLS, because the target is the polytomous Rasch family rather than B&A's 2PL probit model.

Conditional Maximum Likelihood (CML)

For supported RSM/PCM models, CML conditions on each Person's total and observed assignments, removing ability parameters without specifying their population distribution. A generalized elementary-symmetric-function recursion evaluates the conditional likelihood and information. Identification and finite-MLE checks precede structural contrast intervals; their Wald interpretation is asymptotic, not a general finite-sample coverage guarantee. The separate `mfrm_cml_fit` and comparison-only results supply no Person abilities or population SD. Use `mfrm_facet_intervals()` for their native-scale intervals. CML does not support GPCM or ordinary diagnostics.

Joint Maximum Likelihood (JML)

JML estimates all person and facet parameters simultaneously as fixed effects by maximising the joint log-likelihood $\ell(\boldsymbol{\theta}, \boldsymbol{\delta} \mid \mathbf{X})$ directly. It does not assume a parametric person distribution, which can be advantageous when the population shape is strongly non-normal. Its incidental-parameter concern is not simply a small-person-sample issue: structural-parameter bias can persist as the number of persons increases when the number of ratings or items per person remains fixed (Neyman

& Scott, 1948). The package still accepts `"JMLE"` as a backward-compatible alias, but user-facing summaries and documentation use `"JML"` as the public label.

See `fit_mfrm()` for practical guidance on choosing an estimator. General GPCM inference uses MML; JML remains useful for descriptive point estimates and FACETS comparisons.

Strict marginal diagnostics

For `RSM` / `PCM`, `diagnose_mfrm(..., diagnostic_mode = "both")` returns two complementary targets: the `legacy` residual / EAP diagnostics and a `marginal_fit` layer whose expected counts and pairwise summaries are integrated over the posterior quadrature bundle rather than plugged in at the EAP point. The screen is structured as limited-information evidence (Orlando & Thissen, 2000; Haberman & Sinharay, 2013; Sinharay & Monroe, 2025), not as an omnibus accept / reject test, and it complements rather than replaces separation / reliability and inter-rater agreement summaries. The full derivation, with notation and pairwise local-dependence events, lives in `vignette("mfrmr-mml-and-marginal-fit", package = "mfrmr")`.

Statistical background

Key statistics reported throughout the package:

Infit (Information-Weighted Mean Square)

Weighted average of squared standardized residuals, where weights are the model-based variance of each observation:

$$\text{Infit}_j = \frac{\sum_i Z_{ij}^2 \, \text{Var}_i \, w_i}{\sum_i \text{Var}_i \, w_i}$$

Expected value is 1.0 under model fit. Values below 0.5 suggest overfit (Mead-style responses); values above 1.5 suggest underfit (noise or misfit). Infit is most sensitive to unexpected patterns among on-target observations (Wright & Masters, 1982).

Note: The 0.5–1.5 range is the general "productive for measurement" band given by Linacre (2002, *RMT* 16(2), 878). Context-specific bands come from Wright & Linacre (1994, *RMT* 8(3), 370): 0.8–1.2 for high-stakes MCQ, 0.7–1.3 for run-of-the-mill MCQ, 0.6–1.4 for rating-scale surveys, 0.5–1.7 for clinical observation, and 0.4–1.2 for judged performance. See also Bond & Fox (2015) for textbook summaries of these conventions.

Outfit (Unweighted Mean Square)

Simple average of squared standardized residuals:

$$\text{Outfit}_j = \frac{\sum_i Z_{ij}^2 \, w_i}{\sum_i w_i}$$

Same expected value and flagging thresholds as Infit, but more sensitive to extreme off-target outliers (e.g., a high-ability person scoring the lowest category).

ZSTD (Standardized Fit Statistic)

Wilson-Hilferty (1931) cube-root transformation that converts the mean-square chi-square ratio to an approximate standard normal deviate:

$$\mathrm{ZSTD} = \frac{\mathrm{MnSq}^{1/3} - (1 - 2/(9\,df))}{\sqrt{2/(9\,df)}}$$

Values near 0 indicate expected fit. The conventional $|\mathrm{ZSTD}| \geq 2$ and $|\mathrm{ZSTD}| \geq 3$ cutoffs are heuristic two- and three-standard-deviation reference bands (Wright & Linacre, 1994; see also Wilson & Hilferty, 1931), not calibrated 5\ tests. Parameter estimation, sparse cells, the selected df convention, and repeated screening across elements all affect that interpretation. Diagnostics default to FACETS fourth-moment df, admitting df > 0 and capping ZSTD at +/-9. The explicit engine convention withholds df < 1. `whexact = FALSE` uses the expression above (FACETS YES); `TRUE` removes its additive q/3 term (FACETS NO). Both use a cube root. The former linear TRUE transformation is available only as `whexact = "legacy_linear"`.

Residual basis under MML vs JMLE engines

For `method = "MML"` fits, the standardized residuals behind Infit, Outfit, and ZSTD are evaluated at EAP person measures, which are shrunken toward the population mean. JMLE engines such as FACETS evaluate the same formulas at unshrunken JMLE estimates, so MnSq and ZSTD values are not numerically interchangeable across the two residual bases, most visibly for extreme-scoring persons. Use `method = "JML"` when an external FACETS fit comparison requires a JMLE-style residual basis, and see `facets_fit_df_guide()` for the separate standardization-side df/ZSTD conventions.

PTMEA (Point-Measure Correlation)

Pearson correlation between observed scores and estimated person measures within each facet level. A positive value is directionally consistent with the fitted orientation, but it is not a confirmatory test of alignment. A negative value prompts review of orientation, response patterns, and model specification; it does not by itself establish a scoring error.

Separation

Package-reported separation is the ratio of adjusted true standard deviation to root-mean-square measurement error:

$$G = \frac{\mathrm{SD}_{\mathrm{adj}}}{\mathrm{RMSE}}$$

where $\mathrm{SD}_{\mathrm{adj}} = \sqrt{\mathrm{ObservedVariance} - \mathrm{ErrorVariance}}$. Higher values indicate the facet discriminates more statistically distinct levels along the measured variable. In `mfrmr`, `Separation` is the model-based value and `RealSeparation` provides a more conservative companion based on `RealSE`.

Reliability

$$R = \frac{G^2}{1 + G^2}$$

Analogous to Cronbach's alpha or KR-20 for the reproducibility of element ordering. In `mfrmr`, `Reliability` is the model-based value and `RealReliability` gives the conservative companion based on `RealSE`. For `MML`, these are anchored to observed-information `ModelSE` estimates for non-person facets; `JML` keeps them as exploratory summaries.

The facet's role determines the interpretation. High Person separation reliability indicates distinguishable abilities. High rater separation reliability indicates distinguishable severities, **not agreement** between raters. When raters are intended to be interchangeable, reliable severity differences merit review rather than a high-reliability quality label. Low rater separation reliability can also result from imprecise estimates; it does not establish agreement or equivalence. The index neither tests every rater pair nor quantifies the practical size of severity differences. Its Person interpretation is analogous to test reliability, but it is not numerically identical to Cronbach's alpha.

For the person facet under `MML`, the same G and R formulas are applied to EAP person measures with posterior SDs in the error slot. EAP measures are shrunken, so their observed variance is already deflated (approximately the true variance times the reliability), and subtracting the mean posterior variance deflates it again. The reported MML person separation/reliability is therefore a conservative summary: it is systematically lower than the IRT empirical-reliability convention $\mathrm{Var}(\mathrm{EAP})/(\mathrm{Var}(\mathrm{EAP}) + \overline{\mathrm{PSD}^2})$ and is not numerically comparable to JMLE-based person separation reliability from FACETS. The gap is small when measurement is precise and grows as precision drops. Person rows can still carry the model-based precision tier because posterior SDs are model-based quantities; that tier describes the SE source, not FACETS comparability. Use `method = "JML"` when a FACETS-style person separation table is required, and treat MML person rows as conservative summaries.

This is a Rasch/FACETS-style separation reliability on the fitted logit scale, not an intra-class correlation. Use `compute_facet_icc()` only when you want the complementary random-effects variance-share view on the observed-score scale; for non-person facets, large ICC values indicate systematic facet variance rather than desirable measurement reliability.

Strata

Number of statistically distinguishable groups of elements:

$$H = \frac{4G + 1}{3}$$

Three or more strata are commonly used as a practical target (Wright & Masters, 1982), but in this package the estimate inherits the same approximation limits as the separation index.

Key references

- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43, 561–573.
- Bond, T. G., & Fox, C. M. (2015). *Applying the Rasch model* (3rd ed.). Routledge.
- Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, 46, 443–459.
- Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed.). Springer. (AIC / BIC weights and Delta-IC bands used by `compare_mfrm()`.)
- Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology*, 38(1), 67–86. (Source for the `lz` person-fit statistic implemented in `compute_person_fit_indices()`.)

- Haberman, S. J., & Sinharay, S. (2013). Generalized residuals for general models for contingency tables with application to item response theory. *Journal of the American Statistical Association*, 108, 1435–1444.
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly*, 2, 197–221.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176. (Source for the GPCM response-probability kernel used by the package's bounded `fit_mfrm(model = "GPCM")` route. The slope-aware `fair_average_table()` and `estimate_bias()` extensions are package-specific, caveated adaptations built from that kernel, not procedures proposed in Muraki 1992.)
- Muraki, E. (1993). Information functions of the generalized partial credit model. *Applied Psychological Measurement*, 17(4), 351–363. (Companion paper to Muraki 1992 that derives the GPCM item information identity $I_j(\theta) = D^2 a_j^2 \mathrm{Var}(T \mid \theta)$ via Samejima's (1974) polytomous information formula. This is the canonical reference for `compute_information()` under `GPCM`.)
- Samejima, F. (1974). Normal ogive model on the continuous response level in the multidimensional latent space. *Psychometrika*, 39, 111–121. (General polytomous information formula that Muraki 1993 specializes to the GPCM.)
- Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika*, 66(3), 331–342. (Source for the `lz_star` correction in `compute_person_fit_indices()` when person estimates come from the JML/fixed-effect route. MML/EAP person scores are left uncorrected because EAP does not satisfy the Snijders estimating-equation setup.)
- Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.
- Linacre, J. M. (2002). What do Infit and Outfit, mean-square and standardized mean? *Rasch Measurement Transactions*, 16(2), 878.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47, 149–174.
- Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. *Applied Psychological Measurement*, 24, 50–64.
- Orlando, M., & Thissen, D. (2003). Further investigation of the performance of S-X2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement*, 27, 289–298.
- Sinharay, S., Johnson, M. S., & Stern, H. S. (2006). Posterior predictive assessment of item response theory models. *Applied Psychological Measurement*, 30, 298–321.
- Sinharay, S., & Monroe, S. (2025). Assessment of fit of item response theory models: A critical review of the status quo and some future directions. *British Journal of Mathematical and Statistical Psychology*, 78, 711–733.
- Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press.
- Wright, B. D., & Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions*, 8(3), 370.
- Wilson, E. B., & Hilferty, M. M. (1931). The distribution of chi-square. *Proceedings of the National Academy of Sciences of the United States of America*, 17(12), 684-688.

Model selection

RSM vs PCM

The Rating Scale Model (RSM; Andrich, 1978) assumes all levels of the step facet share identical threshold parameters. The Partial Credit Model (PCM; Masters, 1982) allows each level of the `step_facet` to have its own set of thresholds on the package's shared observed score scale. Use RSM when the rating rubric is identical across all items/criteria; use PCM when category boundaries are expected to vary by item or criterion. In the current implementation, PCM assumes one common observed score support across the fitted data, so it should not be described as a fully mixed-category model with arbitrary item-specific category counts.

MML, CML and JML

Marginal Maximum Likelihood (MML) integrates over the person ability distribution using Gauss-Hermite quadrature and does not directly estimate person parameters; person estimates are computed post-hoc via Expected A Posteriori (EAP). Joint Maximum Likelihood (JML) estimates all person and facet parameters simultaneously as fixed effects; `"JMLE"` remains a backward-compatible alias.

Estimator choice should follow the inferential target and assumptions. MML avoids estimating one fixed effect per person, but its structural inference depends on the specified population distribution and response model. JML does not assume a parametric person distribution and can be lighter computationally, but structural-parameter bias can persist as the number of persons increases when the number of ratings or items per person remains fixed. Sample size alone therefore does not determine the preferred route. For supported RSM/PCM structural contrasts, CML instead conditions out Person ability without a population model. It estimates neither abilities nor a population SD; same-fit population-SD standardization requires MML.

See `fit_mfrm()` for usage.

Fixed-calibration scoring after fitting

`predict_mfrm_units()` and `sample_mfrm_plausible_values()` score future or partially observed persons on a quadrature grid under the fitted scoring basis. For ordinary `MML` fits, these summaries inherit the fitted marginal calibration directly. For latent-regression `MML` fits, they use the fitted one-dimensional conditional normal population model and therefore require one-row-per-person background data for the scored units when the fitted population model includes covariates. Intercept-only latent-regression fits (`population_formula = ~ 1`) can reconstruct that minimal person table from the scored person IDs. For `JML` fits, `mfrmr` uses the fitted facet and step parameters together with a standard normal reference prior introduced only for the post hoc scoring layer. This is useful for practical fixed-scale scoring, but it should still be described as a limited approximation rather than as full ConQuest-style population modeling.

ConQuest overlap

The package supports a scoped latent-regression `MML` model, but the overlap with ConQuest should still be described conservatively. The documented comparison is binary `RSM` / `PCM`, one latent dimension, exactly one item facet, complete rectangular responses, and exactly one numeric person covariate beyond the intercept in a conditional-normal person population model. This is a scoped file-based comparison, not a claim of broad ConQuest numerical equivalence for polytomous, multidimensional, incomplete, or arbitrary imported designs.

Author(s)

Maintainer: Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp> (ORCID) [copyright holder]

Authors:

- Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp> (ORCID) [copyright holder]

See Also

Useful links:

- https://ryuya-dot-com.github.io/mfrmr/
- https://github.com/Ryuya-dot-com/mfrmr
- Report bugs at https://github.com/Ryuya-dot-com/mfrmr/issues

Examples

```
# Load the package
library(mfrmr)

# Load example ratings and look at the first six rows
ratings <- load_mfrmr_data("example_operational")
head(ratings)

# Fit the model
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

# Plot the results (Wright map)
plot(fit)

# Save the summary, then display its tables
results <- summary(fit)
results$person_overview # One row summarizing person ability estimates
results$facet_overview  # One row per facet: number of levels, mean, SD, range

# Check the interpretation status and recommended next step
results$decision
```

`analyze_dff` *Differential facet functioning analysis*

Description

Compares observed-minus-expected scores between groups, or describes linked subgroup facet estimates after refitting. Residual differences do not isolate differential functioning and are returned without tests or classifications.

`analyze_dif()` is retained for compatibility with earlier package versions. In many-facet workflows, prefer `analyze_dff()` as the primary entry point.

Usage

```
analyze_dff(
  fit,
  diagnostics,
  facet,
  group,
  data = NULL,
  focal = NULL,
  method = c("residual", "refit"),
  min_obs = 10,
  p_adjust = "holm",
  comparison_review = NULL
)

analyze_dif(...)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`. With `method = "refit"`, it must match `fit`; see mfrmr_compatibility_layer for saved-diagnostic migration. The residual method does not use this argument.
`facet`	Character scalar naming the facet whose elements are tested for differential functioning (for example, `"Criterion"` or `"Rater"`).
`group`	Character scalar naming the column in the data that defines the grouping variable (e.g., `"Gender"`, `"Site"`).
`data`	Optional data frame containing at least the group column and the same person/facet/score columns used to fit the model. If `NULL` (default), mfrmr tries to recover the data from `fit$prep$data`. That slot only holds the columns that `fit_mfrm()` actually modelled; if the grouping column was not among them (common for DIF screening), pass the original data frame via `data = <df>` explicitly. The same applies when the fit object has been serialized without the prep slot. Residual analyses also require the fitted weight column when one was used. See the response-matching contract below for subsets and repeated rows.
`focal`	Optional character vector of group levels to treat as focal. If `NULL` (default), all pairwise group comparisons are performed.
`method`	Analysis method: `"residual"` (default) uses the fitted model's residuals without re-estimation; `"refit"` re-estimates the model within each group subset. The

residual method is faster and avoids additional subgroup fits; neither method establishes adequate sample size.

`min_obs` Minimum number of observations per cell (facet-level x group). Cells below this threshold are flagged as sparse. Residual contrasts/scaled residuals and refit screening statistics are withheld; actual refit point estimates and contrasts remain available. Default `10`.

`p_adjust` Adjustment for refit screening tail areas; default `"holm"`. Retained but unused for residual comparisons, which return no p-values.

`comparison_review`
Optional result of `describe_mfrm_data()` with `comparisons` specified, or its `$comparison_review` component. The retained response rows, origin groups, facet and score coding must match this request. If `focal` is supplied it must equal the review's first group. The prospective interaction states are retained separately; they do not classify the residual/refit contrasts as identified, change estimates, or permit an otherwise refused fit.

`...` Passed directly to `analyze_dff()`.

Details

Differential facet functioning (DFF) occurs when the difficulty or severity of a facet element differs across subgroups of the population, after controlling for overall ability. In an MFRM context this generalises classical DIF (which applies to items) to any facet: raters, criteria, tasks, etc.

Differential functioning is a threat to measurement fairness: if Criterion 1 is harder for Group A than Group B at the same ability level, the measurement scale is no longer group-invariant.

Differences between group ability distributions are not themselves DFF. Residual screens inherit the fitted model's ability and population assumptions. With fixed-standard-normal RSM/PCM MML, subgroup refits retain that population assumption; linking anchors do not estimate subgroup ability distributions. A residual difference can therefore reflect an inadequately represented group difference as well as differential facet functioning. For MML, observation expectations are evaluated at each Person's EAP ability, rather than integrated over the conditional ability distribution. This nonlinear substitution can produce different mean residuals across groups even under a correctly specified no-DFF model. A zero mean-residual contrast and absence of differential functioning are therefore distinct null hypotheses; correcting the SE alone does not make them equivalent.

Two methods are available:

Residual method (`method = "residual"`): Uses the existing fitted model's observation-level residuals. For each facet-level $\times$ group cell, the observed and expected score sums are aggregated and a standardized residual is computed as:

$$z = \frac{\sum(X_{obs} - E_{exp})}{\sqrt{\sum \mathrm{Var}}}$$

Pairwise contrasts compare the mean observed-minus-expected score in Group1 minus that in Group2. Positive values mean higher residual scores in Group1, not greater rater leniency for that group. Differences use score units. `StdResidual` is a descriptive scaling by model response variance, not a t-statistic. No residual-contrast SE, confidence interval, p-value or positive/negative classification is provided. Compatibility columns `SE`, `t`, `df`, `p_value` and `p_adjusted` contain `NA`.

Refit method (`method = "refit"`): Subsets the data by group, refits the MFRM model within each subset, anchors all non-target facets back to the baseline calibration when possible, and compares the resulting facet-level estimates. When linking, convergence, model-based MML precision, finite positive SEs, disjoint Person rosters and cell-count screens all pass, a conditional plug-in Welch statistic is also shown:

$$t = \frac{\hat{\delta}_1 - \hat{\delta}_2}{\sqrt{SE_1^2 + SE_2^2}}$$

This provides group-specific point estimates on a common scale when linking anchors are available. However, the plug-in standard error conditions on the baseline anchors and omits baseline-anchor uncertainty and cross-refit covariance. Refit statistics therefore remain screening evidence and set `FormalInferenceEligible`, `SupportsFormalInference`, `PrimaryReportingEligible`, and `ETS_Eligible` to `FALSE`. The Welch degrees of freedom use a response-count approximation, not independently estimated degrees of freedom for model-based SEs. Disjoint Person IDs do not establish independence or valid matching.

`N_Group1` / `N_Group2` count retained responses for the particular target level in each group; `N_Subgroup1` / `N_Subgroup2` count all retained subgroup responses. Failed or skipped fits have unknown retained counts (`NA`), not zero; `$group_fits[[group]]$N_Input` retains the input count. An absent target in a successfully prepared subgroup has a known count of zero. All baseline target levels remain in the result. Failed/skipped fits and absent levels have missing estimates and contrasts, with `ComparisonStatus` and `ComparisonReason`; they never inherit the baseline estimate. Finite estimates from sparse cells or incomplete diagnostics are retained, while `ConditionalScreenReason` explains withheld screening uncertainty. `NPersons_Group1` / `NPersons_Group2` count Persons in each target cell, and `NPersons_Overlap` counts shared Persons across the entire two subgroup fits. Any overlap withholds the independent-subgroup screen.

`ContrastDirection` describes the location parameter only: `larger_location_in_group1`, `smaller_location_in_group1` or `equal_locations`. Translating this into higher/lower expected scores requires the facet orientation and the remaining response parameters. Subgroup step ladders or slopes can differ; a location difference alone does not order their entire response curves. The subgroup fits replay the baseline response family, resolved score range, step/slope facets, weighting, repeated-rating independence declaration, optimizer, MML engine, and numerical controls; the non-target anchors are the intentional linking change. Direct anchors on the target facet and JML Person anchors from the baseline are preserved. A target fixed by constraints in either subgroup keeps its point estimates, with `Fixed1` / `Fixed2` identifying the fixed values; its contrast, SE and screening tests are unavailable. Equal supplied anchors do not demonstrate absence of differential functioning. Use residual DFF to describe response departures, or explicitly fit a baseline with different target constraints before requesting free-parameter comparisons. Saved refit results created before this preservation rule require rerunning `analyze_dff()` from the baseline fit; loading an old result does not repair its subgroup estimates. Refit analysis fails closed when the baseline uses a user-specified active latent-regression population model, facet interactions, or group-anchor constraints, because those structures do not yet have a complete subgroup-replay and linking contract. The automatically generated intercept-only population model used for default GPCM-MML scale identification is replayed and is not treated as a substantive covariate model.

When `facet` refers to an item-like facet (for example `Criterion`), this recovers the familiar DIF case. When `facet` refers to raters or prompts/tasks, the same machinery supports DRF/DPF-style analyses.

Refit contrasts are not assigned ETS A/B/C labels. Even when subgroup point estimates share a linked logit scale, a validated joint, bootstrap, or replicate covariance contract is required before

formal refit inference.

Refit screening tail areas are adjusted using Holm's step-down procedure by default, jointly across the returned facet-level/group-pair rows in this call. Holm's family-wise error control requires valid unadjusted p-values; the approximate screening tail areas here have not been shown to meet that requirement. Adjustment therefore does not establish an error-rate guarantee or formal inference eligibility. Alternative adjustments can be specified via `p_adjust`; see `stats::p.adjust()` for their assumptions.

Value

An object of class `mfrm_dff` (with compatibility class `mfrm_dif`) with:

- `dif_table`: data.frame of differential-functioning contrasts. Refit rows include `Fixed1` and `Fixed2`; fixed-target rows have `ReportingUse = "not_applicable_fixed_target"` and no contrast or test.
- `cell_table`: (residual method) per-cell detail table.
- `summary`: counts by method-appropriate screening classification.
- `group_fits`: (refit method) per-group facet estimates.
- `gpcm_boundary`: for `GPCM` fits, a capability-boundary table.
- `config`: list with facet, group, method, min_obs, p_adjust settings.

Response matching for residual analyses

`data` selects fitted response rows using Person, all fitted facets, the observed score and fitted weight. Source column names, identifier trimming, the saved original/internal category map and the saved missing-code policy are respected. A changed score does not inherit a residual from another response with the same Person and facets. Rows with missing groups, invalid responses or no fitted match are excluded with reasons in `$cases`; unmatched complete rows also produce a message. Requesting more copies of a response than were fitted is an error. No requested match is an error.

Repeated identical response rows are matched by their multiplicities, with a canonical group order. They must have identical fitted expectations and variances. `FitRow` then identifies an exchangeable fitted copy for this calculation, not a recovered occasion/event identity. This permits descriptive repeated-rating summaries without first-row group assignment or replication of unselected rows. It does not supply cluster-adjusted inference.
`$cases` contains `InputRow` (row of supplied data, or `fit$prep$data` when `data = NULL`), `FitRow` (row of `fit$prep$data`), `GroupValue`, `Included` and `Reason`. Preserve the data and fit alongside this audit. `N` and `N_Group1`/`N_Group2` count retained response rows; `NPersons` and `NPersons_Group1`/`NPersons_Group2` count distinct Persons. The same Person may occur in multiple groups. Neither count establishes independence. Aggregates remain unweighted row summaries even when fitting used weights. Subsetting data retains the original model's fitted expectations; it does not recalibrate the model on the subset.

Choosing a method

Use `method = "residual"` to describe group differences in model residuals without fitting subgroup models. This does not test differential functioning. Use `method = "refit"` when you specifically want group-specific parameter estimates and can tolerate extra computation. Agreement between

methods is not guaranteed by a universal per-group sample-size threshold: stability and detection depend jointly on effect size, category support, response-pattern overlap, linking strength, slope heterogeneity, and estimator behavior. `min_obs` is only a cell-computability/sparsity guard; it is not evidence of power, parameter stability, or sample-size adequacy.

Comparing with MH or ordinal logistic DIF

`mantel_haenszel_table()` supplies a separate binary conditional comparison with explicit matching strata and one response per Person-target cell. `ordinal_logistic_table()` supplies a separate cumulative-logit comparison with uniform/nonuniform terms and a proportional-odds diagnostic. Neither is a `method` of this function. A comparison needs the same persons, target response, group orientation and defensible matching variable. With a common binary Rasch item set and invariant other items, the total including the studied item removes the ability distribution from the conditional item response; a rest score generally does not. This result does not justify pooling different rater/task plans at the same raw total.

Ordinary MH targets a common conditional odds ratio and can miss opposing effects across matching strata. Ordinal logistic models distinguish group and matching-by-group terms, but their cumulative proportional-odds assumption is not generally implied by RSM/PCM. Repeated ratings from one Person cannot be counted as independent persons for either ordinary method. Agreement or disagreement with these procedures does not by itself validate a DFF conclusion. See the linking-and-DFF vignette for the comparison scope and primary references.

Interpreting output

- `$dif_table`: one row per facet-level x group-pair with contrast, a residual mean difference or linked subgroup difference. Residual rows have no SE, test or binary classification. Includes `Method`, `N_Group1`, `N_Group2`, `EffectMetric`, `ClassificationSystem`, `ContrastBasis`, `SEBasis`, `StatisticLabel`, `ProbabilityMetric`, `DFBasis`, `ReportingUse`, `PrimaryReportingEligible`, and `sparse` columns.
- `$cell_table`: (residual method only) per-cell detail with N, NPersons, ObsScore, ExpScore, ObsExpAvg, StdResidual and an interpretation note.
- `$cases`: (residual method only) requested-row matching and exclusion audit, also retained by `summary()`.
- `$summary`: available/unavailable residual comparisons or linked- screening, insufficient-linking, fixed-target and unavailable rows (`method = "refit"`).
- `$group_fits`: (refit method only) list of per-group facet estimates and subgroup linking diagnostics, with `EstimateStatus` / `EstimateReason` and cell, subgroup and input counts.
- `$gpcm_boundary`: for `GPCM` fits, a capability-boundary table marking the DFF/DIF output as caveated screening evidence.
- `$comparison_review`: when supplied, the separate prospective interaction question, weights, counts, assumptions and identification states. Preserved by `summary()`, `dif_report()` and both DFF plot views. Its model can differ from the fitted screening model; read its `fit_arguments` and `notes`.

GPCM boundary

For `GPCM`, DFF/DIF rows are available as slope-aware screening evidence over the fitted expected-score and residual scale. Keep residual-method contrasts and interaction cells in screening language.

Refit contrasts require explicit subgroup linking and precision support for conditional screening, but remain in screening language.

Saved results

Residual results, summaries and reports saved by earlier versions must be recomputed with this function using the fitted model and original data. This does not require refitting the model. Previously exported tables and figures should also be regenerated. Refit results created before the subgroup availability/cell-count checks must instead be recomputed with `method = "refit"`. This reruns the subgroup fits from the saved baseline fit and original data. Reading an old result cannot repair borrowed estimates, dropped target levels or incorrect counts; summary, report and plotting methods reject these earlier refit objects.

Typical workflow

1. Fit a model with `fit_mfrm()`. For `RSM` / `PCM` fairness review, prefer `method = "MML"`.
2. Run `diagnose_mfrm()` and, for `RSM` / `PCM`, prefer `diagnostic_mode = "both"` so legacy and strict marginal screens remain visible together.
3. Run `analyze_dff(fit, diagnostics, facet = "Criterion", group = "Gender", data = my_data)`.
4. Inspect `$dif_table` for contrasts and their reporting status, and `$summary` for method-specific counts. Residual rows have no flags.
5. Use `dif_interaction_table()` when you need cell-level diagnostics.
6. Use plot(x, type = "heatmap") or `dif_report()` for communication.

See Also

`fit_mfrm()`, `estimate_bias()`, `compare_mfrm()`, `dif_interaction_table()`, plot(x, type = "heatmap"), `dif_report()`, `subset_connectivity_report()`, mfrmr_linking_and_dff

Examples

```
# Long-running workflow; run explicitly when needed.
# example("analyze_dff", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
dff <- analyze_dff(fit, diag, facet = "Rater", group = "Group", data = toy)
dff$summary
# Read the residual mean difference and the number of observations per group.
head(dff$dif_table[, c("Level", "Group1", "Group2", "Contrast",
                       "N_Group1", "N_Group2")])
# The residual contrast is an observed-minus-expected average contrast
# between groups. It is useful for screening, but it is not an ETS
# A/B/C logit-delta classification.
dff_refit <- analyze_dff(fit, diag, facet = "Rater", group = "Group",
```

```
                          data = toy, method = "refit")
unique(dff_refit$dif_table$ClassificationSystem)
# Look for: "descriptive". Linked refit contrasts remain screening-only
#   because their plug-in uncertainty omits anchor uncertainty and
#   cross-refit covariance.
sc <- subset_connectivity_report(fit, diagnostics = diag)
plot(sc, type = "design_matrix", draw = FALSE)
if ("ScaleLinkStatus" %in% names(dff_refit$dif_table)) {
  unique(dff_refit$dif_table$ScaleLinkStatus)
}
# Look for: "linked" in `ScaleLinkStatus` confirms the focal and
#   reference groups share enough common elements for a comparable
#   contrast; "demoted_*" rows lose linking under the refit branch
#   and should be read as exploratory.

## End(Not run)
```

`analyze_facet_equivalence` *Analyze practical equivalence within a facet*

Description

Analyze practical equivalence within a facet

Usage

```
analyze_facet_equivalence(
  fit,
  diagnostics = NULL,
  facet = NULL,
  equivalence_bound = 0.5,
  ci_level = 0.95,
  conf_level = NULL
)

## S3 method for class 'mfrm_facet_equivalence'
plot(x, ...)
```

Arguments

`fit`	Output from `fit_mfrm()`. Requires an inference-ready MML fit with unregularized observed-information covariance and estimable contrasts.
`diagnostics`	Optional matching output from `diagnose_mfrm()`. Supplied diagnostics must retain all target-facet levels, model-based SEs and ordinary-inference eligibility. Estimates and covariance are always taken from `fit`; supplying diagnostics cannot override its restrictions. Fixed levels instead require their fixed-value labels and absent sampling SEs.

facet	Character scalar naming a non-person facet. When NULL, a rater-like facet is preferred, otherwise the first model facet is used.
equivalence_bound	Positive practical-equivalence bound in logits. The default 0.5 is not a universal threshold. Choose the smallest practically meaningful difference for the intended use before inspecting the equivalence results.
ci_level	Confidence level for level and grand-mean-deviation intervals (default 0.95). Pairwise TOST always uses alpha 0.05 and 90% intervals.
conf_level	Deprecated alias for ci_level; when supplied it takes precedence and emits a lifecycle deprecation warning.
x	A result from analyze_facet_equivalence().
...	Use type = "forest" (default) for deviations and intervals or type = "rope" for normal confidence-distribution mass. Other controls are described in plot_facet_equivalence().

Details

Pair differences use the full constrained covariance from the MML observed information: $\mathrm{Var}(A - B) = \mathrm{Var}(A) + \mathrm{Var}(B) - 2\mathrm{Cov}(A, B)$. No model is refitted. JML, inference-ineligible fits, missing or regularized covariance, and singular contrast covariance stop with an error. Levels with fixed or unavailable contrasts are not silently dropped. Known anchors are treated as fixed; their uncertainty is excluded. Non-unit observation weights are inference-ineligible, including weights normalized to mean one. Older bundles must retain the current readiness contract as well as the covariance basis before they can be displayed. A numerically verified inverse of ill-conditioned information is retained with a warning, not silently treated as strong evidence. When applicable, cautions, information_review and table column InferenceCaution preserve that warning through summaries and plots. Numerical verification does not establish the accuracy of the normal or chi-square approximations.

The heterogeneity table uses a joint Wald chi-square test of equality of the facet levels. Non-significant heterogeneity is neither necessary nor sufficient for practical equivalence. FixedChiSq, FixedDF, and FixedProb retain their column names but use this joint contrast test. Separation and reliability remain descriptive summaries.

GrandMean is the equally weighted mean of the facet estimates. For each deviation from that mean, uncertainty includes the covariance with the estimated mean. ROPEPct is the mass of its normal confidence distribution inside the practical bound; it is descriptive, not a Bayesian posterior probability or a separate equivalence decision.

The former BIC/Bayes-factor heuristic is unavailable: a Wald statistic is not a fitted likelihood comparison. For compatibility, BF01 is NA and BF01Label states why no value is supplied.

plot() method: Plot a saved facet-equivalence analysis

For plot(): When an interval cannot be represented at the requested numeric precision, the forest retains its estimate as a grey point without a whisker. Inspect data$DeviationCIReason and note in the returned payload for the reason. This does not change the pairwise TOST decision stored in the analysis.

Value

A named list with class mfrm_facet_equivalence.

For `plot()`: The existing equivalence plot payload and inference restrictions. Grand-mean proximity does not establish pairwise equivalence.

What this analysis means

The analysis asks whether differences between facet levels fall within a prespecified practical bound under the fitted model. These are asymptotic normal-approximation tests; numerical eligibility does not establish finite-sample coverage or adequacy of the rating design.

What this analysis does not justify

A non-significant difference is not evidence of equivalence. Pairwise conclusions are unadjusted for multiplicity: selecting some positive pairs does not provide family-wise error control for that selected set. Estimated linking or anchor uncertainty, population transport, and model misspecification require separate evaluation.

Decision rule

Each pair uses two one-sided normal tests at alpha 0.05. `Equivalent` is true when both tests reject non-equivalence, equivalently when its 90% interval lies strictly inside the bound. `Decision` summarizes all pairs as `"all_pairs_equivalent"`, `"partial_pairwise_equivalence"`, or `"no_pairwise_equivalence_est`
No pair may be omitted from the all-pairs summary. Heterogeneity and ROPE summaries do not change this rule. The all-pairs conclusion is an intersection-union decision: if any pair is truly outside the bound, declaring all pairs equivalent requires falsely passing that pair's test. Its asymptotic type-I error is therefore at most alpha when each pair's approximation is valid. This does not provide simultaneous confidence intervals or error control for selected pairs.

Interpreting output

Start with `summary$Decision` and examine the corresponding `pairwise` differences, SEs and intervals. A negative result can reflect imprecision or a material difference. `chi_square` addresses exact equality, while `rope` and `forest` describe proximity to the facet mean.

How to read the main outputs

- `summary`: pairwise decision, covariance basis and multiplicity convention.
- `pairwise`: differences, covariance-aware SEs, 90% intervals and TOST tests.
- `chi_square`: joint Wald heterogeneity test and descriptive separation.
- `rope` / `forest`: `Measure`, marginal `SE` and `CI_Lower`/`CI_Upper`, plus `Deviation`, `DeviationSE` and `DeviationCI_Lower`/`DeviationCI_Upper` for the difference from the estimated facet mean. `Fixed` marks an anchored or constraint-fixed location: its conditional marginal `SE` is zero and its marginal interval is absent. Its deviation from an estimated mean, or contrast with an estimated level, can still have nonzero uncertainty. The forest plot displays these deviations, not the anchored locations. No uncertainty in supplied anchor values is propagated.
- `CIReason`, `DeviationCIReason`, and pairwise `CI90Reason` distinguish `"available"`, `"fixed"`, `"numerical_resolution"` (the requested width rounds away), and `"outside_numeric_range"`. Unavailable endpoints are `NA`; estimates and SEs remain. `ROPEStatus = "unavailable"` means the deviation interval could not be displayed, not that it is inside the bound. The fixed-alpha TOST p-values do not depend on the display `ci_level`.

Recommended next step

Review numerical integration and the model's uncertainty assumptions before interpreting a borderline result. Sensitivity to another practical bound should be reported transparently, without selecting a bound to obtain a desired decision.

Typical workflow

1. Fit and review an MML model with `fit_mfrm()`.
2. Prespecify the practical bound and run `analyze_facet_equivalence()`.
3. Read `summary` and `pairwise`.
4. Use plot(x, type = "facet_equivalence") for descriptive grand-mean proximity.

Output

A bundle with `summary`, `chi_square`, `pairwise`, `rope`, `forest`, and `settings`. Older bundles without the current inference/covariance basis must be recomputed before using `summary()`, `print()`, or plotting.

References

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657-680.

See Also

`facets_chisq_table()`, `fair_average_table()`, plot(x, type = "facet_equivalence")

mfrmr_plot_migration

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 31, maxit = 150)
eq <- analyze_facet_equivalence(fit, facet = "Rater")
eq$summary[, c("Facet", "Elements", "Decision", "MeanROPE")]
head(eq$pairwise[, c("ElementA", "ElementB", "Equivalent")])
```

---

analyze_hierarchical_structure

Analyze the hierarchical structure of a rating design

Description

One-stop review that combines the nesting, cross-tabulation, ICC, and design-effect reports into a single object. Its summary feeds into `reporting_checklist()` after checking the reviewed data against the fit.

Usage

```
analyze_hierarchical_structure(
  data,
  facets = NULL,
  person = "Person",
  score = "Score",
  compute_icc = TRUE,
  ci_method = c("none", "boot"),
  ci_level = 0.95,
  ci_boot_reps = 1000L,
  ci_boot_seed = NULL,
  igraph_layout = TRUE,
  icc_ci_method = NULL,
  icc_ci_level = NULL,
  icc_ci_boot_reps = NULL,
  icc_ci_boot_seed = NULL,
  missing = c("error", "omit")
)
```

Arguments

data	Data frame in long format, or an mfrm_fit (its prep$data is used).
facets	Character vector of facet column names. When data is an mfrm_fit, defaults to fit$prep$facet_names.
person	Person column name. Defaults to "Person".
score	Score column name. Defaults to "Score".
compute_icc	Logical; if TRUE and lme4 is available, adds ICC and design-effect tables.
ci_method	ICC confidence-interval method passed through to compute_facet_icc(). One of "none" (default, point estimate only) or "boot". The former "profile" method is refused because it did not calculate an ICC profile-likelihood interval. Deprecated alias: icc_ci_method (kept for backward compatibility, emits a lifecycle warning).
ci_level	Confidence level when ci_method != "none"; default 0.95. Deprecated alias: icc_ci_level.
ci_boot_reps	Number of bootstrap replicates when ci_method = "boot". Default 1000. Deprecated alias: icc_ci_boot_reps.
ci_boot_seed	Optional RNG seed for reproducible bootstrap CIs. Deprecated alias: icc_ci_boot_seed.
igraph_layout	Logical; if TRUE and igraph is available, adds a connectivity component summary using a bipartite graph over person x facet levels.
icc_ci_method, icc_ci_level, icc_ci_boot_reps, icc_ci_boot_seed	Deprecated compatibility spellings of the ci_* arguments above. Supplying a non-NULL value routes through lifecycle::deprecate_warn() and overrides the canonical ci_* argument.
missing	Missing-value policy for the ICC and design-effect tables, passed to compute_facet_icc(). Invalid input or model errors stop the requested ICC analysis. Nesting, cross-tabulation, and connectivity tables describe the supplied design, including rows

excluded from the ICC model. When data is a fit, ICC row counts start from its stored fitted rows; they cannot recover exclusions made before fitting the MFRM.

Value

A list of class mfrm_hierarchical_structure with:

- nesting: output of detect_facet_nesting().
- crosstabs: list of pairwise observation-count data.frames (long format, suitable for heatmap plotting).
- icc: output of compute_facet_icc() when requested.
- design_effect: output of compute_facet_design_effect() when requested.
- connectivity: named list with bipartite-graph component summary when igraph is available.
- summary: one-row summary used by downstream reporting helpers.
- facets: character vector of facet names that were reviewed (echoed for downstream reporting helpers that need to label rows by review scope).
- data_identity: reviewed column roles, input row count and a compact fingerprint for reuse checks. The optional person column also contributes to nesting pair counts; summary$NFacets counts the requested facets.

plot(x, type = "icc") draws variance shares and any available intervals without an embedded title or footnote. Save its returned object as p and use plot_data(p, "notes") to read the interval method or the reason bounds are unavailable. These notes survive saveRDS()/readRDS(); they do not change the underlying ICC estimates.

Interpreting output

- nesting: a detect_facet_nesting() object with every facet pair classified as Crossed / Partially / Near-perfectly / Fully nested.
- crosstabs: list of (LevelA, LevelB, N) long-format tables, one per facet pair. Plot via plot(x, type = "crosstab", pair = "FacetA__FacetB").
- icc: per-facet variance shares. See compute_facet_icc() for the two-scale interpretation.
- design_effect: per-facet design-effect approximations and descriptive equivalent row counts; not precision estimates for the full design.
- connectivity: number of bipartite components linking Person x facet levels. A single component is required for a common measurement scale; multiple components indicate a disconnected design.

Typical workflow

1. Optional: fit the MFRM with fit_mfrm().
2. Call analyze_hierarchical_structure(fit) (or on the raw data).
3. Read summary(x) for the condensed view.
4. Pass it as hierarchical_structure to reporting_checklist() and save the review with saveRDS() when its full tables are needed later. build_mfrm_manifest() records a separate fit-level rating-count review; it does not accept or archive this structural-review object.

Saved-result reuse

A reporting checklist checks the reviewed column roles and retained rating rows, including scores when an ICC table was computed. It accepts a subset of fitted facets and reports that scope explicitly. Reviews from raw data also work when the selected columns and rows match, using either canonical `Person`/`Score` names or the fit's original source-column names. Row order, unused factor levels, unused columns and MFRM estimates do not affect this match. Nesting and row counts are unweighted; when ICC is absent, the match does not depend on scores. These rules do not establish that a partial review covers all facets or that the design supports valid inference. A review of additional rows, other strata or columns absent from the fit remains usable on its own, but cannot mark this fit's checklist item as available. Recreate older reviews without `data_identity` from the saved fit or matching retained rows and original options; no MFRM refit is needed. See mfrmr_compatibility_layer.

References

McEwen, M. R. (2018). *The effects of incomplete rating designs on results from many-facets-Rasch model analyses* (Doctoral thesis, Brigham Young University). https://scholarsarchive.byu.edu/etd/6689/

Linacre, J. M. (2026). *A User's Guide to FACETS, Version 4.5.0*. Winsteps.com.

Kish, L. (1965). *Survey Sampling*. New York: Wiley.

Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine, 15*(2), 155-163.

See Also

detect_facet_nesting(), facet_small_sample_review(), compute_facet_icc(), compute_facet_design_effect()
reporting_checklist(), build_mfrm_manifest(), fit_mfrm().

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("analyze_hierarchical_structure", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
hs <- analyze_hierarchical_structure(toy,
                                     facets = c("Rater", "Criterion"),
                                     compute_icc = FALSE,
                                     igraph_layout = FALSE)
summary(hs)

# Full review when lme4 and igraph are available.
if (requireNamespace("lme4", quietly = TRUE) &&
    requireNamespace("igraph", quietly = TRUE)) {
  hs_full <- analyze_hierarchical_structure(toy,
                                            facets = c("Rater", "Criterion"))
  summary(hs_full)
  plot(hs_full, type = "icc")
}
```

```
## End(Not run)
```

`analyze_residual_pca` *Run exploratory residual PCA summaries*

Description

Legacy-compatible residual diagnostics can be inspected in two ways:

1. overall residual PCA on the person x combined-facet matrix
2. facet-specific residual PCA on person x facet-level matrices

Usage

```
analyze_residual_pca(
  diagnostics,
  mode = c("overall", "facet", "both"),
  facets = NULL,
  pca_max_factors = 10L,
  parallel = FALSE,
  parallel_reps = 200L,
  parallel_quantile = 0.95,
  parallel_method = c("residual_permutation", "model_bootstrap"),
  seed = NULL
)

## S3 method for class 'mfrm_residual_pca'
plot(x, type = "scree", mode = "overall", ...)
```

Arguments

`diagnostics`	Output from `diagnose_mfrm()` or `fit_mfrm()`.
`mode`	For `analyze_residual_pca()`: `"overall"`, `"facet"`, or `"both"`. For `plot()`: `"overall"` (default) or `"facet"`. Do not also supply `mode` when using a prefixed `type` such as `"facet_scree"`.
`facets`	Optional subset of facets. This selects the facet-specific analyses and the facets combined to define columns in the overall PCA.
`pca_max_factors`	Maximum number of components retained for loadings, subject to matrix dimensions; `"auto"` caps this at 10. Variance tables retain the full eigenvalue spectrum, not only those loading columns.
`parallel`	Logical; if `TRUE`, add a simulation reference to the PCA tables.
`parallel_reps`	Number of permutations or simulated-and-refitted data sets when `parallel = TRUE`. Each model-bootstrap replicate requires a full fit.

parallel_quantile
: Upper reference quantile used as the exploratory comparison cutoff. The default (0.95) follows the common parallel analysis convention.

parallel_method
: Reference method. "residual_permutation" (default) permutes standardized residuals within each column, preserving residual distributions and missingness. "model_bootstrap" generates ratings and refits a supported RSM/PCM MML model; supply the original fit, not detached diagnostics. See Model-generated reference below for restrictions.

seed
: Optional integer seed for reproducible simulation. When supplied, the caller's random-number state is restored after the calculation.

x
: A result from analyze_residual_pca().

type
: "scree", "parallel_scree", "parallel_excess", or "loadings". Existing "overall_*" and "facet_*" spellings remain supported.

...
: facet selects a facet for mode = "facet" (default: the first available facet); component = 1 selects a loading component; top_n = 20 limits displayed loadings. preset is "standard", "publication", "compact", or "monochrome"; draw = FALSE returns data without drawing.

Details

The function works on standardized residual structures derived from diagnose_mfrm(). When a fitted object from fit_mfrm() is supplied, diagnostics are computed internally.

Conceptually, this follows the Rasch residual-PCA tradition of examining structure in model residuals after the primary Rasch dimension has been extracted. In mfrmr, however, the implementation is an **exploratory many-facet adaptation**: it works on standardized residual matrices built as person x combined-facet or person x facet-level layouts, rather than reproducing FACETS/Winsteps residual-contrast tables one-to-one.

Residual PCA should therefore be reported as residual-structure evidence, not as a formal proof of unidimensionality. It also should not be described as DIMTEST or UNIDIM: those essential-unidimensionality tests require a separate item-response-layer definition that is not uniquely determined by a many-facet long data set. In applied MFRM reporting, residual PCA is best triangulated with global residual fit, element fit, and Q3-style local-dependence screens.

Correlations use persons with observed residuals for each pair. Every column and pair must have defined correlations, and the resulting matrix must be positive semidefinite (allowing numerical roundoff). Missing correlations are not set to zero, and invalid matrices are not smoothed. Unavailable analyses retain an explanation in errors. Older stored PCA calculations are recomputed from the supplied observations when this function is called.

Output tables use:

- Component: principal-component index (1, 2, ...)
- Eigenvalue: eigenvalue for each component
- Proportion: component variance proportion
- Cumulative: cumulative variance proportion

When parallel = TRUE, the variance tables additionally include reference summaries (the method is retained in ParallelMethod):

- ParallelMean: mean reference eigenvalue
- ParallelCutoff: parallel_quantile cutoff of reference eigenvalues
- ExcessOverParallelCutoff: observed eigenvalue minus the cutoff
- ExceedsParallelCutoff: whether the observed eigenvalue exceeds the selected reference cutoff

With parallel_method = "residual_permutation", the comparison conditions on the fitted residuals and missingness pattern; it does not simulate responses or refit the model. It does not account for fitted-parameter uncertainty and is not a calibrated dimensionality test. If any requested permutation has unavailable or invalid correlations, the comparison is withheld rather than conditioning on successful permutations. parallel_status retains the successful count and the reason. More permutations improve numerical stability of the reference quantile, but do not establish error-rate control for the fitted model.

For mode = "facet" or "both", by_facet_table additionally includes a Facet column.

summary(pca) is supported through summary(). plot(pca) dispatches on class mfrm_residual_pca. Use type = "scree", "parallel_scree", "parallel_excess" or "loadings", with mode = "overall" or mode = "facet", facet = "Criterion". Existing prefixed spellings such as "facet_scree" remain supported. Plotting a saved PCA does not generate or refit a parallel reference. Save the complete result with base::saveRDS(), including errors, warnings, parallel_status and any bootstrap_trials; export the desired tables separately with utils::write.csv(). as_ggplot() converts a saved plot payload without redoing PCA. A blank unavailable plot is not evidence of an absence of residual structure.

plot() method: Plot saved residual PCA

Display a saved residual PCA or its saved parallel-analysis comparison. Plotting does not generate a new parallel reference.

Value

A named list with:

- mode: resolved mode used for computation
- facet_names: facets analyzed
- overall: overall PCA bundle (or NULL)
- by_facet: named list of facet PCA bundles
- overall_table: variance table for overall PCA
- by_facet_table: stacked variance table across facets
- parallel_settings, parallel_overall_table, parallel_by_facet_table, and parallel_status: returned for every call; the parallel tables are populated when parallel = TRUE
- bootstrap_trials, bootstrap_settings: attempted replicate/scope records and generation settings for the model bootstrap; otherwise NULL.
- errors: explanations for unavailable overall or facet PCA analyses; unavailable analyses have empty variance tables.
- warnings: named list of non-fatal PCA warnings captured from the underlying PCA engine. These indicate exploratory boundary conditions, not confirmatory evidence.

- `InferenceTier`, `SupportsFormalInference`, `PrimaryReportingEligible`, `ReportingUse`, and `DecisionUse`: machine-readable guards that keep this route exploratory and prohibit an automatic dimensionality or subscore decision.

For `plot()`: A named list of plotting data (class `mfrm_plot_data`) with:

- `plot`: `"scree"`, `"parallel_scree"`, `"parallel_excess"`, or `"loadings"`
- `mode`: `"overall"` or `"facet"`
- `facet`: facet name (or `NULL`)
- `title`: plot title text
- `data`: underlying table used for plotting
- `InferenceTier`, `SupportsFormalInference`, `PrimaryReportingEligible`, `ReportingUse`, and `DecisionUse`: machine-readable exploratory-screening guards

Interpreting output

Use `overall_table` first:

- early components with noticeably larger eigenvalues or proportions suggest residual structure that may deserve follow-up. Small early components do not establish unidimensionality; inspect the residual aggregation, available person overlaps, fit and local-dependence screens.

Then inspect `by_facet_table`:

- describes patterns within each facet-level residual matrix. Different numbers of columns, aggregation and overlaps prevent treating eigenvalues across facets as a ranking of their contributions to a single decomposition.

Finally, inspect loadings via plot(x, type = "residual_pca") to identify which variables/elements drive each component.

Model-generated reference

`parallel_method = "model_bootstrap"` is an exploratory parametric-bootstrap reference for native RSM/PCM MML fits with a fixed standard-normal population, fixed quadrature, additive facets and unit weights. Anchors, dummy/positive facets, shrinkage, estimated population models, GPCM, JML and extended testlet/shared-rater models are not supported by this reference yet.

Each replicate draws one independent standard-normal ability per Person and shares it across that Person's retained rating rows. Fitted facet and step parameters generate conditionally independent ratings. The model is refitted with the original identification, category map, quadrature and optimization settings. Standardized residuals use the same scoring and aggregation as the observed analysis. One refit supplies every requested PCA scope.

The reference conditions on the analyzed assignment and observation pattern: it does not fill unassigned or missing ratings or simulate informative missingness. Sparse designs remain usable only when the required residual correlations are defined and their matrix is positive semidefinite. Source and refitted models must pass numerical convergence checks. The observed PCA itself can be unavailable despite successful fitting, for example because the assignment lacks shared-Person pairs. This is reported separately from refit failure. Every planned replicate is retained

in bootstrap_trials, including warnings and failures. An affected scope has no reference cutoff if any replicate fails; successful replicates are not substituted or selected to form its reference. Raw eigenvalue draws, with missing rows for failures, are stored under each PCA bundle's parallel$draws.

Refitting includes variation from parameter estimation under the fitted null; generating parameters themselves are plug-in estimates, not posterior draws. Cutoffs are componentwise reference quantiles, not multiplicity-adjusted tests. Scanning every component does not preserve the nominal false-flag probability of one component. Absence of a flag is not evidence that all model assumptions hold. Few replicates give unstable tail quantiles. Neither a fixed number of replicates nor this algorithm establishes calibrated false-flag rates or proves unidimensionality. Inspect convergence, quadrature sensitivity, overlap and competing dependence explanations before substantive use.

Dimensionality and network follow-up

Checking a one-dimensional model does not require fitting a multidimensional alternative first. Residual structure can also reflect testlet, rater or assignment effects; neither the component count nor the count plus one is an estimated number of substantive abilities. Combine the screen with q3_statistic(), marginal-fit review and substantive task/criterion evidence. The current permutation reference does not replace a fitted-model parametric bootstrap. No native residual-network/EGA or calibrated dimensionality decision is supplied here. Residual communities would describe remaining associations, not automatically latent dimensions. See vignette("mfrmr-visual-diagnostics") for alternatives and their scope.

Typical workflow

1. Fit model and run diagnose_mfrm() with residual_pca = "none" or "both".
2. Call analyze_residual_pca(..., mode = "both").
3. Review summary(pca), then plot scree/loadings.
4. Cross-check with fit/misfit diagnostics before conclusions.

Interpretation

For plot():

- "scree" shows component eigenvalues. A dominant early component is exploratory evidence of residual structure, not a standalone test of unidimensionality.
- "parallel_scree" shows the observed eigenvalues and saved reference mean/cutoff; "parallel_excess" shows the observed-minus-cutoff difference. Compute these references with parallel = TRUE in analyze_residual_pca() first. Exceeding a cutoff suggests follow-up, not proof of multidimensionality. Inspect parallel_status and bootstrap_trials for unavailable comparisons and distinguish residual permutation from refitted model bootstrap.
- "loadings" shows the largest absolute loadings; inspect both signs and magnitudes to identify the contributing variables or elements.

Use overall mode before locating patterns within individual facets. Repeated plots of a saved result reuse the same PCA and do not provide independent evidence.

References

The residual-PCA idea follows the Rasch residual-structure literature, especially Linacre's discussions of principal components of Rasch residuals. The current `mfrmr` implementation should be interpreted as an exploratory extension for many-facet workflows rather than as a direct reproduction of a single FACETS/Winsteps output table.

The optional parallel analysis follows Horn's data-driven eigenvalue comparison logic and later recommendations to compare observed eigenvalues with high quantiles of a reference distribution. Here that reference is generated by within-column permutation of standardized residuals by default. The optional model bootstrap instead regenerates ratings and refits the supported model. The cited factor-retention literature does not calibrate either many-facet residual comparison as a fitted-model test.

- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30, 179-185. doi:10.1007/BF02289447.
- Glorfeld, L. W. (1995). An improvement on Horn's parallel analysis methodology for selecting the correct number of factors to retain. *Educational and Psychological Measurement*, 55, 377-393. doi:10.1177/0013164495055003002.
- Hayton, J. C., Allen, D. G., & Scarpello, V. (2004). Factor retention decisions in exploratory factor analysis: A tutorial on parallel analysis. *Organizational Research Methods*, 7, 191-205.
- Timmerman, M. E., & Lorenzo-Seva, U. (2011). Dimensionality assessment of ordered polytomous items with parallel analysis. *Psychological Methods*, 16, 209-220.
- Linacre, J. M. (1998). *Structure in Rasch residuals: Why principal components analysis (PCA)?* Rasch Measurement Transactions, 12(2), 636. https://www.rasch.org/rmt/rmt122m.htm
- Linacre, J. M. (1998). *Detecting multidimensionality: Which residual data-type works best?* Journal of Outcome Measurement, 2(3), 266-283.

Applications and broader measurement discussions below provide context; they do not validate this implementation's residual aggregation, reference cutoffs or automatic dimensionality decisions:

- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly*, 2(3), 197-221.
- Yamashita, T. (2024). An application of many-facet Rasch measurement to evaluate automated essay scoring: A case of ChatGPT-4.0. *Research Methods in Applied Linguistics*, 3(3), 100133.
- Uto, M. (2021). A multidimensional generalized many-facet Rasch model for rubric-based performance assessment. *Behaviormetrika*, 48(2), 425-457.
- Aryadoust, V., Ng, L. Y., & Sayama, H. (2021). A comprehensive review of Rasch measurement in language assessment: Recommendations and guidelines for research. *Language Testing*, 38(1), 6-40.
- Tseng, W.-T. (2016). Measuring English vocabulary size via computerized adaptive testing. *Computers & Education*, 97, 69-85.

See Also

`diagnose_mfrm()`, plot(x, type = "residual_pca"), mfrmr_visual_diagnostics

`analyze_residual_pca()`, mfrmr_plot_migration

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- diagnose_mfrm(fit, residual_pca = "both")
pca <- analyze_residual_pca(diag, mode = "both")
pca2 <- analyze_residual_pca(fit, mode = "both")
summary(pca)
p <- plot(pca, type = "residual_pca", mode = "overall", view = "scree", draw = FALSE)
p$data$plot
head(p$data)
# Optional permutation reference repeats the PCA calculation.
# example("analyze_residual_pca", run.dontrun = TRUE)
## Not run:
pca_pa <- analyze_residual_pca(diag, mode = "overall", parallel = TRUE, parallel_reps = 10)
head(pca_pa$overall_table)

## End(Not run)
head(pca$overall_table)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_residual_pca", run.dontrun = TRUE)
## Not run:
ratings_all <- load_mfrmr_data("example_core")
selected_people <- unique(ratings_all$Person)[1:24]
ratings <- ratings_all[
  ratings_all$Person %in% selected_people, , drop = FALSE]
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "JML")
diag <- diagnose_mfrm(fit, residual_pca = "overall")
pca <- analyze_residual_pca(diag, mode = "overall")
plt <- plot(pca, type = "scree", draw = FALSE)
head(plt$data)
# Ten replicates illustrate the display, not a stable reference cutoff.
pca_pa <- analyze_residual_pca(diag, mode = "overall", parallel = TRUE,
                               parallel_reps = 10, seed = 2026)
pa <- plot(pca_pa, type = "parallel_scree", draw = FALSE)
head(pa$data)
plt_load <- plot(pca, type = "loadings", component = 1, draw = FALSE)
head(plt_load$data)
if (interactive()) {
  plot(pca, type = "scree", preset = "publication")
}

## End(Not run)
```

---

anchor_to_baseline *Fit new data anchored to a baseline calibration*

Description

Re-estimates a fitted many-facet model on new data while holding selected facet parameters fixed at the values from a previous (baseline) calibration. This transfers baseline coordinates into a new fit through direct equality constraints. Common-scale interpretation is conditional on a defensible baseline and invariant cross-run element identity. For GPCM, treat this as direct exploratory anchor/drift support rather than as the package's formal linking-synthesis route.

Usage

```
anchor_to_baseline(
  new_data,
  baseline_fit,
  person,
  facets,
  score,
  anchor_facets = NULL,
  include_person = FALSE,
  weight = NULL,
  model = NULL,
  method = NULL,
  anchor_policy = "warn",
  mml_integration = NULL,
  centering_scope = NULL,
  ...
)

## S3 method for class 'mfrm_anchored_fit'
print(x, ...)

## S3 method for class 'mfrm_anchored_fit'
summary(object, ...)

## S3 method for class 'summary.mfrm_anchored_fit'
print(x, ...)
```

Arguments

new_data	Data frame in long format (one row per rating).
baseline_fit	An inference-ready mfrm_fit object from a previous calibration under the current readiness contract.
person	Character column name for person/examinee.
facets	Character vector of facet column names.
score	Character column name for the rating score.
anchor_facets	Character vector of facets to anchor (default: all non-Person facets).
include_person	If TRUE, also export Person values for a receiving JML fit. MML rejects Person anchors. Use only when constant ability for the same Persons is substantively intended; shared Persons need not be fixed.

weight	Optional character column name for observation weights.
model	Scale model override; defaults to baseline model.
method	Estimation method override; defaults to baseline method.
anchor_policy	How to handle anchor issues: "warn", "error", "silent".
mml_integration	MML integration mode passed to fit_mfrm(). NULL inherits the baseline fit's mode for MML; JML uses "fixed".
centering_scope	NULL inherits the baseline's convention. Older fits without this setting retain "remainder". Supply "facet" or "remainder" explicitly only when that constraint system is intended; see fit_mfrm().
...	For anchor_to_baseline(), additional arguments passed to fit_mfrm(). Ignored by the S3 print and summary methods on this page.
object	An mfrm_anchored_fit object (for summary).
x	An mfrm_anchored_fit object.

Details

This function automates the baseline-anchored calibration workflow:

1. Extracts anchor values from the baseline fit using make_anchor_table().
2. Re-estimates the model on new_data with those anchors fixed via fit_mfrm(..., anchors = anchor_table).
3. Runs diagnose_mfrm() on the anchored fit.
4. Compares the fixed non-Person anchor values in the new fit with their supplied values. This checks constraint consistency, not invariance.

The helper refuses a baseline unless its current readiness record has InferenceReady = TRUE. Also confirm compatible model, score coding, orientation, and population conventions, and document that anchor labels denote the same elements with invariant meaning. The function checks object shape and receiving-data anchor compatibility, but does not establish those substantive conditions. Fixing parameters can identify coordinates; it does not create observed common-rating links.

Only model, method, the centering convention and MML integration mode inherit baseline settings automatically. Supply other settings explicitly through ..., including rubric bounds, category policy, facet orientation, PCM/GPCM facet roles and any population model. RSM/PCM MML without a population model uses a fixed standard-normal ability distribution; anchors do not automatically estimate a new cohort's mean or variance.

The table fixes locations only: thresholds and GPCM slopes are re-estimated. Use extract_mfrm_calibration() and score_mfrm_calibration() when the question is scoring against a complete fixed calibration. Partial anchors retain the centering restrictions described in fit_mfrm(); this is not an unrestricted fixed-common-parameter linking implementation for arbitrary new levels. Elements present in the anchor table but absent from the new data are handled according to anchor_policy: "warn" (default) emits a message, "error" stops execution, and "silent" ignores silently.

The returned drift table contains only non-Person anchor rows that match the new data. Unanchored common levels and Person levels are absent. The compared values are fixed by construction, so zero differences do not show that raters or tasks remained stable. For separate cross-wave

screening, fit the waves independently and use `detect_anchor_drift()`, retaining its offset and uncertainty limitations.

Differences compare each retained fixed value to the supplied anchor:

$$\Delta_e = \hat{\delta}_{e,\text{new}} - \hat{\delta}_{e,\text{base}}$$

An element is **flagged** when $|\Delta_e| > 0.5$ logits or $|\Delta_e / SE_{\Delta_e}| > 2.0$, where $SE_{\Delta_e} = \sqrt{SE^2_{\text{base}} + SE^2_{\text{new}}}$. `SE_Diff` is a plug-in independence calculation. It does not propagate baseline-anchor uncertainty or cross-fit covariance, and anchored estimates in the new fit are constrained rather than independently re-estimated. Consequently, `Drift_SE_Ratio` and `Flag` are descriptive consistency screens, not formal z tests or calibrated error-rate decisions.

Value

Object of class `mfrm_anchored_fit` with components:

fit The anchored `mfrm_fit` object.

diagnostics Output of `diagnose_mfrm()` on the anchored fit.

baseline_anchors Anchor table extracted from the baseline.

drift Tibble of element-level drift statistics.

Which function should I use?

- Use `anchor_to_baseline()` when you have one new dataset and want to place it on baseline coordinates within the documented constraint system.
- Use `detect_anchor_drift()` when you already have multiple fitted waves and want to compare their stability.
- Use `build_equating_chain()` when you need cumulative offsets across an ordered series of waves.

Interpreting output

- `$drift`: one row per retained non-Person anchor with columns `Facet`, `Level`, `Baseline`, `New`, `Drift`, `SE_Baseline`, `SE_New`, `SE_Diff`, `Drift_SE_Ratio`, and `Flag`. Read this as a constraint-consistency table. Zero differences are expected by construction and provide no independent evidence of stability.
- `$fit`: the full anchored `mfrm_fit` object, usable with `diagnose_mfrm()`, `measurable_summary_table()`, etc.
- `$diagnostics`: pre-computed diagnostics for the anchored calibration.
- `$baseline_anchors`: the candidate table fed to `fit_mfrm()`. Some rows may be excluded when `anchor_policy` permits unmatched IDs. Inspect `$fit$config$anchor_review$anchors` for the retained direct constraints.

Typical workflow

1. Fit the baseline model: `fit1 <- fit_mfrm(...)`.
2. Collect new data (e.g., a later administration).

3. Call `res <- anchor_to_baseline(new_data, fit1, ...)`.
4. Inspect `summary(res)` for numerical readiness and fixed-value consistency; neither establishes anchor invariance or a successful substantive link.
5. For multi-wave drift monitoring, fit waves separately and pass the fits to `detect_anchor_drift()` or `build_equating_chain()`.

See Also

`fit_mfrm()`, `make_anchor_table()`, `detect_anchor_drift()`, `diagnose_mfrm()`, `build_equating_chain()`, mfrmr_linking_and_dff

Examples

```
# Long-running workflow; run explicitly when needed.
# example("anchor_to_baseline", run.dontrun = TRUE)
## Not run:
# Packaged synthetic teaching data; no data-generation code is needed.
ratings <- load_mfrmr_data("example_core")
people <- unique(ratings$Person)
# Two balanced 24-Person waves retain every Rater and Criterion.
d1 <- ratings[ratings$Person %in% people[1:24], , drop = FALSE]
d2 <- ratings[ratings$Person %in% people[25:48], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "MML", rating_min = 1, rating_max = 4,
                 category_policy = "preserve")
res <- anchor_to_baseline(d2, fit1, "Person",
                          c("Rater", "Criterion"), "Score",
                          anchor_facets = "Criterion", anchor_policy = "error",
                          rating_min = 1, rating_max = 4,
                          category_policy = "preserve")
summary(res)
# These fixed-value differences are not a test of Criterion stability.
head(res$drift[, c("Facet", "Level", "Drift", "Flag")])
res$baseline_anchors[1:3, ]

## End(Not run)
```

`apa_table` *Build an APA-oriented table handoff using base R structures*

Description

Build an APA-oriented table handoff using base R structures

Usage

```
apa_table(
  x,
  which = NULL,
  diagnostics = NULL,
  digits = 2,
  caption = NULL,
  note = NULL,
  bias_results = NULL,
  context = list(),
  whexact = FALSE,
  branch = c("apa", "facets")
)

## S3 method for class 'apa_table'
summary(object, digits = 3, top_n = 8, ...)

## S3 method for class 'apa_table'
plot(
  x,
  y = NULL,
  type = c("numeric_profile", "first_numeric"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	For apa_table(): A data.frame, mfrm_fit, summary() output supported by build_summary_table_bundle(), an mfrm_summary_table_bundle, diagnostics list, bias-result list, saved RSM/PCM fixed-facet intervals, or saved GPCM slope/curve/bootstrap inference. Ordinary Person-score results also support which = "decisions" when cut scores were requested. For plot(): Output from apa_table().
which	Optional table selector when x has multiple tables.
diagnostics	Optional diagnostics from diagnose_mfrm() (used when x is mfrm_fit and which targets diagnostics tables).
digits	For apa_table(): Uniform number of rounding digits for numeric columns, except saved decision probabilities, boundaries and priors, which are retained at full precision in table. For summary(): Number of digits used for numeric summaries.
caption	Optional caption text.
note	Optional note text.

bias_results	Optional output from estimate_bias() used when auto-generating APA metadata for fit-based tables.
context	Optional context list forwarded when auto-generating APA metadata for fit-based tables.
whexact	Logical forwarded to APA metadata helpers.
branch	Output branch: "apa" for manuscript-oriented labels, "facets" for FACETS-aligned labels.
object	Output from apa_table().
top_n	Maximum numeric columns shown in numeric_profile.
...	Reserved for generic compatibility.
y	Reserved for generic compatibility.
type	Plot type: "numeric_profile" (column means) or "first_numeric" (distribution of the first numeric column).
main	Optional title override.
palette	Optional named color overrides.
label_angle	Axis-label rotation angle for bar-type plots.
draw	If TRUE, draw using base graphics.

Details

This helper avoids styling dependencies and returns a reproducible base data.frame plus manuscript-oriented metadata. It does not claim complete APA 7 or JARS compliance: outside the saved decisions table, digits applies the same rounding rule to every numeric column, so statistic-specific formatting (for example, exact *p*-value, confidence-interval, and effect-size conventions) and the target journal's final typography still require human review.

Supported which values:

- For mfrm_fit: "summary", "person", "facets", "steps"
- For summary() outputs or mfrm_summary_table_bundle: names listed in build_summary_table_bundle(x)$table_
- For diagnostics list: "overall_fit", "measures", "fit", "reliability", "facets_chisq", "bias", "interactions", "interrater_summary", "interrater_pairs", "obs"
- For bias-result list: "table", "summary", "chi_sq"
- For ordinary score_mfrm_persons() output: "decisions". predict_mfrm_units() and score_mfrm_calibration() results and their summaries also accept this selector. Cuts must have been requested.
- For RSM/PCM fixed-facet intervals: "intervals" (default), "settings", "contrasts", and "clusters" when present. Method, confidence level and unavailable reasons remain with the selected estimates and bounds.
- For GPCM inference: "intervals" or "curves"; bootstrap results also retain "trials", "checks" and "source_checks" when recorded, "sampling", and "availability" for slope intervals or "test" for a null-model LRT. Extended results also expose "settings", and "clusters"/"contrasts" when present. Saved profile intervals also expose "profile", "profile_endpoints", "profile_checks" and "wald". Target and method columns are preserved.

summary() method: Summarize an APA/FACETS table object

For summary(): Compact summary helper for QA of table data before manuscript export.

plot() method: Plot an APA/FACETS table object using base R

For plot(): Quick visualization helper for numeric columns in apa_table() output. It is intended for table QA and exploratory checks, not final publication graphics.

Value

A list of class apa_table with fields:

- table (data.frame)
- which
- caption
- note
- digits
- branch, style

For summary(): An object of class summary.apa_table.

For plot(): A plotting-data object of class mfrm_plot_data.

Interpreting output

Saved ordinary Person-score, unit-prediction and portable-score results with cut probabilities support which = "decisions". Numeric probabilities, cut boundaries and priors are retained without decimal rounding in the table. They condition on calibration and the scoring prior; the cut labels do not establish CEFR alignment or a decision rule.

- table: plain data.frame ready for export or further formatting.
- which: source component that produced the table.
- caption/note: manuscript-oriented metadata stored with the table.

For summary():

- overview: table size/composition and missingness.
- numeric_profile: quick distribution summary of numeric columns.
- caption/note: text metadata readiness.

For plot():

- "numeric_profile": compares column means to spot scale/centering mismatches.
- "first_numeric": checks distribution shape of the first numeric column.

Typical workflow

1. Build table object with `apa_table(...)`.
2. Inspect quickly with `summary(tbl)`.
3. Render base preview via `plot(tbl, ...)` or export `tbl$table`.

For `summary()`:

1. Build table with `apa_table()`.
2. Run `summary(tbl)` and inspect `overview`.
3. Use `plot.apa_table()` for quick numeric checks if needed.

For `plot()`:

1. Build table with `apa_table()`.
2. Run `summary(tbl)` for metadata.
3. Use `plot(tbl, type = "numeric_profile")` for quick numeric QC.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `build_apa_outputs()`, `reporting_checklist()`, mfrmr_reporting_and_apa, `predict_mfrm_units()`

`apa_table()`, `plot()`

`apa_table()`, `summary()`

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Turn one summary table into a table with a caption and note
results <- summary(fit)
tbl <- apa_table(results, which = "facet_overview",
                 caption = "Distribution of estimates within each facet")
tbl # Prints the table, caption, and note

# Extract the ordinary data frame for further formatting or export
tbl$table
# This table summarizes facets; use as.data.frame(fit) for individual estimates
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.apa_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
tbl <- apa_table(fit, which = "summary")
```

```
summary(tbl)

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.apa_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
tbl <- apa_table(fit, which = "summary")
p <- plot(tbl, draw = FALSE)
p2 <- plot(tbl, type = "first_numeric", draw = FALSE)
if (interactive()) {
  plot(
    tbl,
    type = "numeric_profile",
    main = "APA Numeric Profile (Customized)",
    palette = c(numeric_profile = "#2b8cbe", grid = "#d9d9d9"),
    label_angle = 45
  )
}

## End(Not run)
```

`apply_empirical_bayes_shrinkage` *Apply empirical-Bayes shrinkage to fitted non-person facet estimates*

Description

Post-hoc normal-model shrinkage helper that augments an `mfrm_fit` with descriptive empirical-Bayes adjustments for each non-person facet. The shrinkage variance $\hat{\tau}^2$ is estimated by method of moments from the facet-level point estimates and their standard errors:

$$\hat{\tau}^2 = \max\left(0, \frac{1}{K}\sum_{j=1}^{K}\hat{\delta}_j^2 - \overline{\mathrm{SE}^2}\right),$$

where zero is the chosen shrinkage target and the moment is calculated on complete finite estimate/positive-SE pairs. A sum-to-zero identification constraint does not establish a known population mean or independent errors. The shrinkage factor is $B_j = \mathrm{SE}_j^2/(\hat{\tau}^2 + \mathrm{SE}_j^2)$, with $\hat{\delta}_j^{EB} = (1 - B_j)\hat{\delta}_j$ and $\mathrm{ShrunkSE}_j = \sqrt{(1 - B_j)\mathrm{SE}_j^2}$.

Usage

```
apply_empirical_bayes_shrinkage(
  fit,
  facet_prior_sd = NULL,
  shrink_person = FALSE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm() with a non-empty facets$others table.
facet_prior_sd	Optional numeric scalar. When supplied, the shrinkage variance is fixed at facet_prior_sd^2 instead of being estimated from the data. Its square must be representable, and positive when the supplied SD is positive. Useful when a prior is elicited from expert knowledge or a previous fit.
shrink_person	Logical. When TRUE, the same empirical-Bayes adjustment is also applied to ordinary-JML Person estimates with available sampling SEs. Default FALSE. Other estimators are refused: MML EAP estimates already use a population prior, and their posterior SDs are not sampling SEs for a second normal-model update.

Details

ShrunkSE is a plug-in normal-model quantity conditional on the selected prior variance. It omits uncertainty in that variance, cross-level covariance and the fitted identification/anchor constraints. It is neither a calibrated SE nor a guaranteed lower bound. Unequal shrinkage can break the original sum-to-zero constraint. Original estimates, anchors and predictions are unchanged; this helper does not refit a hierarchical MFRM. The normal-normal formula is exact for an independent Gaussian observation with a known Gaussian prior variance. Applying it to jointly calibrated MFRM estimates is a descriptive approximation; a James-Stein improvement in risk is not established for this adjustment.

A zero estimated prior variance fully pools eligible estimates at zero and gives zero plug-in SE; this does not establish perfect precision. Optional plot whiskers are descriptive normal bands. Their legacy CI column names do not confer confidence-interval coverage; SupportsFormalInference is false. Invalid estimate/SE pairs retain their original values with unavailable shrinkage factors. With fewer than three valid pairs no shrinkage is applied. Unrepresentable moment variances also leave the original values unchanged, with unavailable factors and an explanatory Note in the report.

The report records complete-pair counts and mean shrinkage over those pairs. EffectiveDF sums retained weights over eligible levels, conditional on the chosen prior variance; it is not model degrees of freedom for testing or IC. Reapply this helper to existing fits to refresh old reports; no MFRM refit is needed. Regenerate previously saved plots and exports as well. Updated reports carry CalculationVersion = 2: positive prior variances no longer produce zero plug-in SE merely because a shrinkage factor rounded to one. Reapplication replaces the previous adjustment; shrink_person = FALSE removes any previous Person shrinkage columns. The original fitting inputs remain recorded separately from the latest post-fit shrinkage settings. build_mfrm_replay_script() replays the adjustment after fitting, preserving whether diagnostic SEs were attached by fit_mfrm(). Manually edited SEs or other table edits require their own reproducible editing steps.

Value

The same mfrm_fit, with augmented columns and a new shrinkage_report list entry, and with fit$config$facet_shrinkage set to "empirical_bayes".

Typical workflow

1. Fit the model as usual with fit_mfrm().

2. Call `apply_empirical_bayes_shrinkage(fit)` when small-N facets are present and zero-centered pooling is substantively defensible (see `facet_small_sample_review()`); small counts alone do not require it.
3. Report both the original and shrunk estimates in the manuscript, citing Efron & Morris (1973). `build_apa_outputs()` will add the sentence automatically when `fit$config$facet_shrinkage` is set.

References

Efron, B., & Morris, C. (1973). Combining possibly related estimation problems. *Journal of the Royal Statistical Society: Series B, 35*(3), 379-402.

Efron, B. (2021). *Empirical Bayes: Concepts and methods* (Technical report). Department of Statistics, Stanford University. https://efron.ckirby.su.domains/papers/2021EB-concepts-methods.pdf

Morris, C. N. (1983). Parametric empirical Bayes inference: Theory and applications. *Journal of the American Statistical Association, 78*(381), 47-55.

See Also

`fit_mfrm()` (which accepts `facet_shrinkage` directly), `facet_small_sample_review()`, `compute_facet_icc()`.

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
fit_eb <- apply_empirical_bayes_shrinkage(fit)
fit_eb$shrinkage_report
# Look for:
# - `Tau2` is the estimated between-level prior variance per facet.
#   `Tau2 = 0` fully pools eligible estimates at zero
#   (`MeanShrinkage = 1`). Zero plug-in SE is not perfect precision.
# - `MeanShrinkage` near 0 = little movement, near 1 = heavy pooling
#   toward the chosen zero target; inspect the SEs and prior variance.
# - `EffectiveDF` sums retained weights over eligible levels, conditional
#   on the chosen prior variance; it is not degrees of freedom for testing.
head(fit_eb$facets$others[, c("Facet", "Level", "Estimate",
                              "ShrunkEstimate", "ShrinkageFactor")])
# Look for: rows where `ShrinkageFactor` is large (close to 1) had
#   their estimates pulled most strongly toward the facet mean (0).
```

`as.data.frame.mfrm_fit` *Convert mfrm_fit to a tidy data.frame*

Description

Returns all facet-level estimates (person and others) in a single tidy data.frame. Person rows retain their original identifiers; review or transform them before writing the result outside a controlled analysis environment.

Usage

```
## S3 method for class 'mfrm_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An mfrm_fit object from fit_mfrm.
row.names	Ignored (included for S3 generic compatibility).
optional	Ignored (included for S3 generic compatibility).
...	Additional arguments (ignored).

Details

This method returns four columns (Facet, Level, Estimate, Extreme) so that the result is easy to inspect, join, or write to disk.

Value

A data.frame with columns Facet, Level, Estimate, and Extreme. The Extreme column is populated for person rows from the extreme-score flag ("Min" / "Max" / NA); non-person facet rows carry NA in that column by design. Corrected-JML fits additionally retain estimator, correction order, point/covariance status, uncertainty target and structural RootSE columns. Person RootSE is missing; extreme Person flags are "low" / "high" / "none". The estimation_note attribute explains the local-root uncertainty and remaining bias. Ordinary diagnostic attachments do not apply to this estimator; see the Corrected JML section of fit_mfrm.

Interpreting output

Person estimates are returned with Facet = "Person". All non-person facets are stacked underneath in the same schema.

Typical workflow

1. Fit a model with fit_mfrm().
2. Convert with as.data.frame(fit) for a compact long-format export.
3. Join additional diagnostics later if you need SE or fit statistics.

See Also

fit_mfrm, export_mfrm_tables

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("as.data.frame.mfrm_fit", run.dontrun = TRUE)
## Not run:
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Extract estimates and select the rows to display
estimates <- as.data.frame(fit)
head(subset(estimates, Facet == "Person")) # First six persons
subset(estimates, Facet == "Rater")       # All raters
subset(estimates, Facet == "Criterion")   # All criteria

## End(Not run)
```

---

assess_mfrm_recovery *Assess whether recovery-simulation results are ready to use*

Description

Converts the numerical output from simulate_mfrm_recovery() into a reviewer-facing adequacy checklist. The goal is not to impose one universal pass/fail rule; it is to make the main user questions explicit: Did the runs finish? Did the fitted models converge? Are uncertainty summaries available? Are coverage and Monte Carlo precision plausible? If practical RMSE or bias limits are supplied, which parameter groups need follow-up? For GPCM, which slope-regime generator condition frames the recovery evidence? This only reads an existing simulation result. It does not generate data, refit models, add replications or repair unavailable uncertainty estimates.

Usage

```
assess_mfrm_recovery(
  x,
  min_reps = 30,
  min_success_rate = 0.95,
  min_convergence_rate = 0.95,
  min_se_available = 0.8,
  coverage_target = 0.95,
  coverage_tolerance = 0.05,
  max_mcse_rmse_ratio = 0.25,
  max_rmse = NULL,
  max_abs_bias = NULL,
  top_n = 6,
  digits = 3,
  ...
```

```
)

## S3 method for class 'mfrm_recovery_assessment'
plot(
  x,
  y = NULL,
  type = c("status", "metrics"),
  metric = c("rmse", "bias", "coverage", "se_available", "mcse_rmse"),
  draw = TRUE,
  ...
)
```

Arguments

x	For `assess_mfrm_recovery()`, output from `simulate_mfrm_recovery()`. For `plot.mfrm_recovery_assessment()`, output from `assess_mfrm_recovery()`.
min_reps	Minimum replication count expected before treating the simulation for inferential use.
min_success_rate	Minimum acceptable proportion of replications that generated data and produced a fitted model.
min_convergence_rate	Minimum acceptable proportion of replications whose fitted model reported convergence.
min_se_available	Minimum acceptable proportion of all expected targets with matching structural standard errors in each parameter group, including targets in failed fits. Unknown target counts give `not_available`. Set to `NULL` to skip this check.
coverage_target	Nominal coverage target, usually `0.95`.
coverage_tolerance	Absolute tolerance around `coverage_target`.
max_mcse_rmse_ratio	Maximum acceptable Monte Carlo SE of RMSE divided by RMSE. Set to `NULL` to skip this precision check.
max_rmse	Optional practical RMSE limit. Use a scalar for all parameter groups or a named vector/list with names such as `"facet"`, `"step"`, `"slope"`, `"Rater"`, or `"facet:Rater:logit"`.
max_abs_bias	Optional practical absolute-bias limit. Naming follows `max_rmse`. Mean-aligned location groups and complete identified log-slope blocks have zero average bias by construction and remain `not_assessed`.
top_n	Number of next-action lines retained in the compact output.
digits	Digits used by the print method.
...	For `assess_mfrm_recovery()`, unused arguments must be empty; unknown assessment options are rejected rather than silently ignored. For the plot method, reserved for S3 generic compatibility.

y	Reserved for S3 generic compatibility.
type	Assessment plot route. "status" summarizes checklist status counts; "metrics" plots a parameter-group assessment metric colored by its status.
metric	Metric used when type = "metrics". Supported values are "rmse", "bias", "coverage", "se_available", and "mcse_rmse".
draw	If TRUE, draw with base graphics. If FALSE, return an mfrm_plot_data object with reusable plot tables and metadata.

Details

RMSE and bias adequacy depends on the substantive scale and the use case, so the function does not mark them as failed unless the user supplies max_rmse or max_abs_bias. Without those limits, the corresponding rows are marked not_assessed and the next action asks the user to set practical thresholds when a decision depends on the metric. BiasBasis explains groups whose mean bias is fixed at zero by identification; even a supplied bias limit cannot turn that identity into recovery evidence. Threshold names are matched from the most specific group to its type or default; see max_rmse for examples.

Metrics are recomputed from row-level evidence. Coverage is conditional on the available centered facet intervals, not all planned targets. uncertainty_review retains counts, CoverageAvailabilityRate, McseCoverage95 and CIReasons alongside its descriptive checklist. See simulate_mfrm_recovery() for the estimands, independent-trial MCSE, unsupported interval groups and handling of legacy saved results. Checklist tolerances do not replace the Monte Carlo uncertainty or qualify an estimator.

The condition_review table is generator metadata for interpreting the recovery run. For GPCM, GPCMSlopeRegime, StressLevel, and generated score-category support describe the data-generating condition; they are not model-fit tests and they are not literature-derived adequacy cut points. condition_reporting_notes turns those generator conditions into reporter-facing caveats, such as high-dispersion slope stress or sparse generated score support.

The optional diagnostic_review table is available when simulate_mfrm_recovery() was called with include_diagnostics = TRUE. It summarizes fit and separation operating characteristics as diagnostic context only. Its availability fields do not mean that fit or separation values are adequate, and those rows do not enter the recovery adequacy status. diagnostic_reporting_notes should be read first when drafting fit/separation language because it separates zero separation/reliability, absolute fit-ZSTD flags, and df-sensitive ZSTD flags from recovery criteria.

plot.mfrm_recovery_assessment() is a user-facing review aid. Use type = "status" first to see where checklist attention is needed, then type = "metrics" to inspect the parameter groups behind RMSE, bias, coverage, standard-error availability, or Monte Carlo precision statuses. The intended reading order is summary(recovery_review), then condition_reporting_notes and condition_review, then diagnostic_reporting_notes and diagnostic_review, then the status plot, then the metric plot, then the row-level recovery table for the parameter groups that need follow-up. When draw = FALSE, the plot data also include reading_order, guidance, condition/diagnostic handoff tables, and user-facing plot tables such as section_status for status plots and metric_review for metric plots.

Value

An object of class mfrm_recovery_assessment with:

- `overview`: compact run-level status.
- `checklist`: reviewer-facing adequacy checks.
- `condition_review`: generator-condition metadata, including `GPCM` slope-regime interpretation and generated score-category support when available.
- `condition_reporting_notes`: reporter-facing generator-condition caveats separated from parameter-recovery conclusions.
- `diagnostic_reporting_notes`: reporter-facing fit/separation caveats retained as diagnostic context rather than recovery criteria.
- `diagnostic_review`: optional fit/separation operating-characteristic context when retained by `simulate_mfrm_recovery()`.
- `metric_review`: parameter-group metric checks.
- `uncertainty_review`: coverage / SE availability interpretation with expected/available counts, coverage MCSE and unavailable reasons.
- `reading_order`: recommended first-read order for the summary, condition, plot, and row-level recovery outputs.
- `next_actions`: short action list sorted by severity.
- `thresholds`: thresholds used for the assessment.

See Also

`simulate_mfrm_recovery()`, `plot.mfrm_recovery_simulation()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("assess_mfrm_recovery", run.dontrun = TRUE)
## Not run:
rec <- simulate_mfrm_recovery(
  n_person = 12,
  n_rater = 2,
  n_criterion = 2,
  reps = 1,
  maxit = 30,
  seed = 123
)
assess_mfrm_recovery(rec, min_reps = 1, max_rmse = 1)

# Read the GPCM generator condition separately from recovery adequacy.
gpcm_spec <- build_mfrm_sim_spec(
  n_person = 14,
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 2,
  model = "GPCM",
  step_facet = "Criterion",
  slope_facet = "Criterion",
  slopes = c(0.85, 1.15),
  assignment = "crossed"
```

```
)
gpcm_rec <- suppressWarnings(simulate_mfrm_recovery(
  sim_spec = gpcm_spec,
  reps = 1,
  fit_method = "MML",
  quad_points = 5,
  maxit = 12,
  include_diagnostics = TRUE,
  include_person = FALSE,
  seed = 456
))
gpcm_review <- assess_mfrm_recovery(
  gpcm_rec,
  min_reps = 1,
  max_rmse = c(slope = 2),
  max_abs_bias = c(slope = 1),
  min_se_available = NULL,
  max_mcse_rmse_ratio = NULL
)
gpcm_review$condition_reporting_notes[, c(
  "ConditionArea", "ReportingAttention", "ConditionFinding"
)]
gpcm_review$condition_review[, c(
  "Model", "GPCMSlopeRegime", "StressLevel", "ScoreSupportStatus"
)]
gpcm_review$diagnostic_reporting_notes[, c(
  "Facet", "ReportingAttention", "DiagnosticFinding"
)]
summary(gpcm_review)$reading_order

## End(Not run)
```

`as_flextable` *Generic for converting objects to a* `flextable`

Description

Generic for converting objects to a `flextable`

Usage

```
as_flextable(x, ...)
```

Arguments

`x`	Object to convert.
`...`	Passed to methods.

Value

A `flextable` object (concrete return type from the underlying method, e.g. `[as_flextable.apa_table()]` returns a `flextable` ready for `flextable::save_as_docx()`).

See Also

`as_flextable.apa_table()` for the `apa_table` method; `as_kable()` for a `knitr::kable`-targeted alternative; `apa_table()` for constructing an `apa_table` in the first place.

Examples

```
# Optional integration example (loads additional modelling or rendering packages).
# example("as_flextable", run.dontrun = TRUE)
## Not run:
tbl <- structure(
  list(
    table = data.frame(Term = c("Rater A", "Rater B"), Estimate = c(-0.12, 0.18)),
    caption = "Facet estimates",
    note = "Toy values for formatting only."
  ),
  class = "apa_table"
)
if (requireNamespace("flextable", quietly = TRUE)) {
  invisible(as_flextable(tbl))
}

## End(Not run)
```

---

as_flextable.apa_table

Convert an `apa_table` *to a* `flextable`

Description

Produces a Word / PowerPoint-friendly `flextable` with the caption and note wired in. Requires `flextable` (in Suggests).

Usage

```
## S3 method for class 'apa_table'
as_flextable(x, ...)
```

Arguments

`x`	An `apa_table` object from `apa_table()`.
`...`	Reserved for generic compatibility.

Value

A `flextable` object, or a message when `flextable` is unavailable.

See Also

`as_kable.apa_table()`, `apa_table()`.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("as_flextable.apa_table", run.dontrun = TRUE)
## Not run:
tbl <- structure(
  list(
    table = data.frame(Term = c("Rater A", "Rater B"), Estimate = c(-0.12, 0.18)),
    caption = "Facet estimates",
    note = "Toy values for formatting only."
  ),
  class = "apa_table"
)
if (requireNamespace("flextable", quietly = TRUE)) {
  invisible(as_flextable(tbl))
}

## End(Not run)
```

---

as_ggplot *Convert draw-free mfrmr plot data to ggplot2*

Description

`as_ggplot()` is an optional renderer for an `mfrm_plot_data` object or an object whose `plot()` method supports `draw = FALSE`. Base graphics remain the default; the returned `ggplot` can be restyled or composed downstream.

Usage

```
as_ggplot(x, type = NULL, component = NULL, ...)

## Default S3 method:
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_design_evaluation'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_design_evaluation_plot_data'
as_ggplot(x, type = NULL, component = NULL, ...)
```

```
## S3 method for class 'mfrm_signal_detection'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_signal_detection_plot_data'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_plot_data'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_slope_intervals'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_curve_intervals'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_gpcm_bootstrap'
as_ggplot(x, type = NULL, component = NULL, ...)

## S3 method for class 'mfrm_results'
as_ggplot(x, type = NULL, component = NULL, ...)
```

Arguments

x	An mfrm_plot_data object, or an mfrmr object with a draw-free plot method. Corrected-JML slope, location and step plots retain their point or cumulative-distribution view, labels, reference and interpretation note; conversion does not add confidence intervals.
type	Optional plot type passed to plot() for a non-plot-data input. A saved plot-data input already selects a view and rejects type.
component	Optional tabular payload component to convert.
...	Arguments passed to the draw-free plot method. CCC conversion additionally accepts slope_aes, facet_by, and show_overlay. A saved mfrm_plot_data object already contains its view and display settings: it rejects extra arguments (except those three CCC controls for a complete CCC view). To change the level or source plot settings, create a new payload from the fitted model or statistical result. For appearance changes after conversion, use ggplot tools such as ggplot2::labs(title = NULL).

Details

Bubble conversion uses the saved radii, facet colours, facet order and reference lines. It preserves relative circle radii, rather than replacing them with equal-sized points. Base circles use plot units; ggplot point sizes use physical units, so absolute sizes need not match across devices. Payloads lacking matching saved radii or facet colours must be recreated with plot(x, type = "bubble").

Dedicated conversions are provided for Wright maps, theta-to-expected-score pathways, fit-statistic-to-measure pathways, category characteristic curves, bubble charts, DIF/DFF summaries and heatmaps, portable-calibration score review plots, testlet calibration, shared-rater severity and bootstrap intervals, conditional Person scores, and multivariate D-study comparisons. Matched ordinary/extended

facet comparisons preserve centered effects, paired/difference views, excluded rows, display controls and alternative text; see plot.mfrm_extended_comparison(). Portable-calibration score views preserve their selected axes, stored endpoints, review symbols and unplotted dispositions with either the default or component = "data". Use plot_data() for other components; conversion does not rescore Persons or change their interval level. Agreement conversions retain the selected exact/correlation/difference metric, saved pair ordering and review thresholds, model-expected overlay, and missing comparisons. Agreement heatmaps use the saved facet labels and palette with a fixed metric scale. Use plot_data() for other tables in an agreement payload; automatic conversion does not guess a different statistic from the first numeric column. Linking conversions preserve the selected drift, wave estimates and stored interval endpoints, cumulative offsets, or common-element counts. Missing offsets do not become zero or connect a line across an unavailable link. Drift heatmaps use their saved residual matrix. Use plot_data() for other tables; conversion does not recompute offsets or their uncertainty. Threshold-sensitivity tile and curve payloads also preserve their selected view, display controls and text alternatives; see plot.mfrm_screening_sensitivity(). Response residual displays preserve their posterior or conditional plug-in probability definition and lack of reference cutoffs. Paired views retain available Infit when Outfit is undefined; see plot.mfrm_response_diagnostics(). Testlet conversions retain unavailable rows, prior-only symbols and the conditional-interval note; they do not estimate diagnostics or add calibration uncertainty. Extended-model interval, precision and empirical-distribution views preserve display settings, data exclusions and text alternatives; see plot.mfrm_testlet_scores(). Bootstrap conversion retains infinite endpoints as arrows and ordinary intervals as a dashed comparison. D-study conversions preserve G/Phi or SEM panels, fixed-count groups, score units, and unavailable estimates. They retain supplied D-study coefficient ribbons and interval gaps. Converting a saved plot object does not refit models or calculate intervals. G-study component conversions retain raw negative points, component intervals and square outlines for unavailable intervals. For a G-study or D-study input, intervals = TRUE is forwarded to its plot method and computes intervals before conversion; also supply assumption = "normal". For multivariate D-study plots, an explicit component = "series" keeps this dedicated conversion; use plot_data() for other tables. Difference-interval plots from mfrm_multivariate_d_compare() use their base plot() method or plot_data(); automatic conversion is not supported. Severity profiles preserve their saved normal bounds, ordered levels, guide bands, fixed points, unavailable states and precision notes with the default or component = "data". The word "profile" does not imply a likelihood-profile interval. Conversion does not recalculate bounds or qualify exploratory JML or fallback/regularized uncertainty for inference. Residual-PCA scree, reference, excess and loadings plots have dedicated conversions with the default or component = "data". They preserve saved eigenvalues, reference cutoffs, signed loadings and display encodings; method identifiers are not inferred as plotted variables. No PCA is recomputed. External-feature PCA scree, scores and loadings views have dedicated conversions. They preserve selected axes, retained-component symbols, saved group colours and shapes, labels and transformation metadata. Scores retain equal axis units; loading coefficients are not correlations. The default and component = "table" use the same complete PCA view. Use ggplot2::labs(title = NULL, subtitle = NULL) to remove headings from the returned ggplot. No PCA or clustering is refitted. External-feature dendrogram conversion retains the stored merges, heights, leaf order and group boxes, including tied-height cuts. The default and component = "tree" use the same complete view. Labels follow the saved setting; no tree is refitted or partition selected. Source metadata remain available with plot_data(). Heights are not significance or branch support. Imputation co-membership heatmaps have dedicated conversion with the default or component = "matrix". Values retain their all-imputation denominator, selected ID order and fixed zero-to-one scale. Unavailable cells use grey fill and crosses; zero is

a valid fraction. The saved legend, labels and imputation count are reused, without recomputing or pooling partitions. Silhouette conversion preserves widths on a fixed minus-one-to-one scale, their saved order and overall-mean line. Numeric profiles preserve original-unit means, medians and counts, using separate symbols with slight vertical offsets. Categorical profiles preserve the category order, unused levels, group counts and within-group proportions on a zero-to-one scale. The default and `component = "table"` keep the full selected view; categorical profiles also accept `component = "matrix"`. These summaries do not estimate uncertainty, rater quality or sampling stability. For main-effects `mfrm_d_study` results, use the base `plot()` method or `plot_data()` for custom graphics; automatic conversion is refused because generic column selection does not preserve those design comparisons. Fit-measure `measure_ci` plots retain each facet/level row and its saved normal-band endpoints, including exploratory JML bands. Open diamonds mark fixed values and crosses mark estimates without intervals. Conversion does not recompute bounds or change their inferential qualification. The default and `component = "table"` preserve the complete view and its returned notes. Pooled fixed-facet MI intervals have dedicated conversion with the default or `component = "table"`. Saved t-interval endpoints, degrees of freedom, contrasts and complete-data information cautions are retained, with no repooling or interval recalculation. Open diamonds mark fixed targets without intervals, crosses mark missing intervals, and arrows mark infinite bounds. Arrow tips are plotting limits, not replacement finite confidence limits. Selecting `component` does not enable unsupported conversions: it cannot preserve D-study differences through generic column selection. Use `plot_data()` to extract the table and specify the axes, intervals and grouping explicitly in custom graphics. Residual Q-Q plots require the base renderer or explicitly specified custom quantile axes; automatic conversion is unavailable, even with `component`. Residual-correlation heatmaps retain the selected facet on both axes, a fixed -1 to 1 colour scale, and grey unavailable cells; their pair table and notes retain overlap counts and interpretation qualifications. Saved rating-scale counts and thresholds, and category-structure counts, retain the selected values, missing outcomes, expected-count overlay and saved palette. Use the default or the matching `category_table` / `threshold_table` component. Threshold figures show point estimates; stored SEs and CI-use flags remain in the payload, not as curve or gap intervals. Category-curve bundles (including their overview), category-structure boundary views and threshold ladders require their base renderer or custom graphics from `plot_data()`; automatic conversion is unavailable even with `component`. Generic column selection would mix distinct curve targets or lose the selected view. Fitted CCC plots from plot(fit, type = "ccc") retain their dedicated conversion.

Bias scatter/ranked views preserve their axes, saved review rules and conditional interval bounds, including dashed search-limited bounds. Bias heatmaps, histograms and facet profiles require base/custom graphics; the generic converter cannot combine their distinct layouts safely. Halo distributions, selected per-rater summaries and signed retained-edge matrices preserve the selected metric, missing comparisons and caveats. Halo network conversion is unavailable. Save the complete analysis result and recreate its plot to obtain current interpretation notes.

Shrinkage plots from plot(fit, type = "shrinkage") preserve both estimates for each facet level, their adjustment direction and saved descriptive bands. Use the default or `component = "data"`. Conversion does not turn plug-in bands into calibrated confidence intervals. Recreate older plots from a fit with a refreshed shrinkage report first.

Other draw-free payloads use a conservative tabular fallback; inspect `plot_data_components()` when automatic inference is not appropriate. Titles, subtitles and captions are omitted by default and retained as metadata. Use `plot_data(p, "notes")` to retrieve descriptions and cautions. Explicitly enabled text is wrapped at 72 columns; use `ggplot2::labs()` for custom labels or narrower exports. Fit plots created with `show_title = FALSE` or `show_notes = FALSE` retain those settings on conversion. To set them when converting a fit directly, pass the flags through `...`. Interpre-

tation and display notes remain available in `attr(plot, "mfrmr_notes")`, when present in the source payload. Wright, pathway, and CCC conversions share the fit-family series palette, honour monochrome presets and supplied palette overrides, and preserve line-type or point-shape distinctions. CCC categories retain their source order. When `slope_aes = "colour"`, slopes instead use a continuous viridis scale (a grey gradient in monochrome); categories still have line types.

Fixed-facet RSM/PCM intervals retain the selected covariance method, level, status symbols and optional ordinary comparison. Titles, subtitles, captions, reference lines and legends can be changed through the source plot arguments. Methods use both line types and vertical offsets, including in monochrome.

Value

A `ggplot2` plot object, with a `mfrmr_notes` attribute when the source plot payload contains a `notes` table. Paired Wright/CCC payloads from plot(reference, comparison = comparison) retain their selected comparison or difference view, group selection and monochrome panel policy.

Interactive views

The visual gallery shows how to pass a supported ggplot to `plotly::ggplotly()` for browser-based hovering, zooming and legend filtering. Plotly is an optional downstream renderer. Keep the original mfrmr plot and its `plot_data()` notes alongside the widget; conversion does not recalculate estimates or intervals. Check labels, custom geoms and interval layers after conversion. The gallery includes static figures, numeric tables and an HTML export recipe.

See Also

`mfrmr_output_guide()` with `scope = "plots"` for selected purpose-based routes, conversion status and alternatives.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("as_ggplot", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
# A balanced slice retains every Rater and Criterion while running quickly.
toy <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
fit <- fit_mfrm(toy, "Person",
                c("Rater", "Criterion"), "Score", maxit = 30)
as_ggplot(fit, type = "wright")
as_ggplot(fit, type = "fit_pathway", include_person = TRUE)
as_ggplot(plot(fit, type = "ccc", draw = FALSE))

## End(Not run)
```

as_kable *Generic for converting objects to a* knitr::kable

Description

Generic for converting objects to a knitr::kable

Usage

```
as_kable(x, ...)
```

Arguments

x	Object to convert.
...	Passed to methods.

Value

A knitr::kable object (concrete return type from the underlying method, e.g. [as_kable.apa_table()] returns a kableExtra object when the package is installed).

See Also

as_kable.apa_table() for the apa_table method; as_flextable() for a flextable-targeted alternative; apa_table() for constructing an apa_table in the first place.

Examples

```
tbl <- structure(
  list(
    table = data.frame(Term = c("Rater A", "Rater B"), Estimate = c(-0.12, 0.18)),
    caption = "Facet estimates",
    note = "Toy values for formatting only."
  ),
  class = "apa_table"
)
if (requireNamespace("knitr", quietly = TRUE)) {
  invisible(as_kable(tbl))
}
```

as_kable.apa_table *Convert an* apa_table *to a* knitr::kable() *object*

Description

Renders the table data for direct inclusion in RMarkdown, Quarto, or HTML reports, wiring the caption and note slots into the standard APA placement (caption above, note below). When kableExtra is installed the note is attached as a footer for HTML or LaTeX. Otherwise the complete rendered table is collapsed to one knitr_kable string before a single note is appended.

Usage

```
## S3 method for class 'apa_table'
as_kable(x, format = c("pipe", "html", "latex"), digits = 3L, ...)
```

Arguments

x	An apa_table object from apa_table().
format	One of "pipe" (default, Markdown), "html", or "latex", passed through to knitr::kable().
digits	Numeric; passed to knitr::kable().
...	Additional arguments forwarded to knitr::kable().

Value

A knitr_kable object ready to be printed inline in a report, or a message when knitr is unavailable.

See Also

as_flextable.apa_table(), apa_table().

Examples

```
tbl <- structure(
  list(
    table = data.frame(Term = c("Rater A", "Rater B"), Estimate = c(-0.12, 0.18)),
    caption = "Facet estimates",
    note = "Toy values for formatting only."
  ),
  class = "apa_table"
)
if (requireNamespace("knitr", quietly = TRUE)) {
  invisible(as_kable(tbl))
}
```

automated_scoring_table *Automated-scoring compatibility interface for score comparisons*

Description

Prefer `score_comparison_table()` for new analyses. This earlier interface retains its argument names, automated/human output columns and default Williamson (2012) numerical screen. The generic interface uses score/reference names and applies no thresholds unless requested.

Usage

```
automated_scoring_table(
  automated,
  human,
  human_reference = NULL,
  categories,
  automated_categories = automated,
  smd = c("pooled", "human"),
  qwk_min = 0.7,
  correlation_min = 0.7,
  max_degradation = 0.1,
  max_abs_smd = 0.15,
  fit = NULL,
  diagnostics = NULL,
  rater_facet = NULL,
  automated_rater = NULL
)
```

Arguments

`automated`, `human`, `human_reference`
: Earlier names for `scores`, `reference`, and `reference2` in `score_comparison_table()`.

`categories`
: Declared numeric score categories in increasing order.

`automated_categories`
: Earlier name for `score_categories`.

`smd`
: `"pooled"` or `"human"`; the latter maps to `"reference"`.

`qwk_min`, `correlation_min`, `max_degradation`, `max_abs_smd`
: Numerical screen limits; see `score_comparison_table()`. These defaults retain the earlier automated-scoring examples and are not universal acceptance criteria.

`fit`, `diagnostics`, `rater_facet`
: Optional saved MFRM source and rater facet; see `score_comparison_table()`.

`automated_rater`
: Earlier name for `rater`.

Value

The same list structure as score_comparison_table(), retaining the earlier automated/human column and category-table names. Saved results created through the earlier interface can still be read directly.

See Also

score_comparison_table()

---

bias_count_table *Build a bias-cell count report*

Description

Build a bias-cell count report

Usage

```
bias_count_table(
  bias_results,
  min_count_warn = 10,
  branch = c("original", "facets"),
  fit = NULL
)
```

Arguments

bias_results	Output from estimate_bias().
min_count_warn	Minimum count threshold for flagging sparse bias cells.
branch	Output branch: "facets" keeps legacy manual-aligned naming, "original" returns compact QC-oriented names.
fit	Optional fit_mfrm() result used to attach run context metadata.

Details

This helper summarizes how many observations contribute to each bias-cell estimate and flags sparse cells.

Branch behavior:

- "facets": keeps legacy manual-aligned column labels (Sq, Observd Count, Obs-Exp Average, Model S.E.) for side-by-side comparison with external workflows.
- "original": keeps compact field names (Count, BiasSize, SE) for custom QC workflows and scripting.

Value

A named list with:

- `table`: cell-level counts with low-count flags
- `by_facet`: named list of counts aggregated by each interaction facet
- `by_facet_a`, `by_facet_b`: first two facet summaries (legacy compatibility)
- `summary`: one-row summary
- `thresholds`: applied thresholds
- `branch`, `style`: output branch metadata
- `fit_overview`: optional one-row fit metadata when `fit` is supplied

Interpreting output

- `table`: cell-level contribution counts and low-count flags.
- `by_facet`: sparse-cell structure by each interaction facet.
- `summary`: overall low-count prevalence.
- `fit_overview`: optional run context (when `fit` is supplied).

Low-count cells should be interpreted cautiously because bias-size estimates can become unstable with sparse support.

Typical workflow

1. Estimate bias with `estimate_bias()`.
2. Build `bias_count_table(...)` in desired branch.
3. Review low-count flags before interpreting bias magnitudes.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `table` data.frame contains, in the legacy-compatible branch:

FacetA, FacetB Interaction facet level identifiers; generic names for the two interaction facets.

Sq Sequential row number.

Observd Count Number of observations for this cell.

Obs-Exp Average Observed minus expected average for this cell.

Model S.E. Standard error of the bias estimate.

Infit, Outfit Fit statistics for this cell.

LowCountFlag Logical; `TRUE` when count < `min_count_warn`.

The `summary` data.frame contains:

InteractionFacets Names of the interaction facets.

Cells, TotalCount Number of cells and total observations.

LowCountCells, LowCountPercent Number and share of low-count cells.

See Also

estimate_bias(), unexpected_after_bias_table(), build_fixed_reports(), mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("bias_count_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t11 <- bias_count_table(bias)
t11_facets <- bias_count_table(bias, branch = "facets", fit = fit)
summary(t11)
p <- plot(t11, draw = FALSE)
p2 <- plot(t11, type = "lowcount_by_facet", draw = FALSE)
if (interactive()) {
  plot(
    t11,
    type = "cell_counts",
    draw = TRUE,
    main = "Bias Cell Counts (Customized)",
    palette = c(count = "#2b8cbe", low = "#cb181d"),
    label_angle = 45
  )
}

## End(Not run)
```

---

bias_interaction_report

Build a bias-interaction plot-data bundle (FACETS Table 13: ranked bias list)

Description

Bundles the **ranked flagged-cells** view of a bias-interaction run for downstream printing and plotting. The three sibling reports in this family are intentionally distinct:

- bias_interaction_report() (this one) = FACETS Table 13: a ranked list of interaction cells with `t`, `bias size`, and screening tail area – use when reviewing which `(facet_a, facet_b)` cells deserve follow-up.
- bias_iteration_report() = iteration history / convergence trace for the bias recalibration (FACETS Table 9 territory) – use when diagnosing whether the bias run itself stabilised.
- bias_pairwise_report() = pairwise contrast table for a target facet (FACETS Table 14 territory) – use when comparing levels within a facet while controlling for the other.

Usage

```
bias_interaction_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  top_n = 50,
  abs_t_warn = 2,
  abs_bias_warn = 0.5,
  p_max = 0.05,
  sort_by = c("abs_t", "abs_bias", "prob")
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
max_abs	Bound for absolute bias size when estimating from fit.
omit_extreme	Omit extreme-only elements when estimating from fit.
max_iter	Iteration cap for bias estimation when x is fit.
tol	Convergence tolerance for bias estimation when x is fit.
top_n	Maximum number of ranked rows to keep.
abs_t_warn	Warning cutoff for absolute t statistics.
abs_bias_warn	Warning cutoff for absolute bias size.
p_max	Warning cutoff for p-values.
sort_by	Ranking key: "abs_t", "abs_bias", or "prob".

Details

Preferred bundle API for interaction-bias diagnostics. The function can:

- use a precomputed bias object from estimate_bias(), or
- estimate internally from mfrm_fit + facet specification.

Save the complete result with base::saveRDS(); plot(out, type = "scatter") and type = "ranked" reuse its flags and thresholds. Rebuild the report from the saved bias estimates to change screening rules; conflicting plotting-time thresholds are rejected. Both views support as_ggplot(), including conditional bounds requested with show_ci = TRUE. Heatmaps, histograms and facet profiles use base graphics or custom plot_data() graphics. Export the bundle's data frames with utils::write.csv().

Value

A named list with bias-interaction plotting/report components. Class: mfrm_bias_interaction.

Interpreting output

Focus on ranked rows where multiple screening criteria converge:

- large absolute t statistic
- large absolute bias size
- small screening tail area

The bundle is optimized for downstream summary() and plot(x, type = "bias") views.

Typical workflow

1. Run estimate_bias() (or provide mfrm_fit here).
2. Build bias_interaction_report(...).
3. Review summary(out) and visualize with plot(x, type = "bias").

See Also

estimate_bias(), build_fixed_reports(), plot(x, type = "bias")

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("bias_interaction_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
out <- bias_interaction_report(bias, top_n = 10)
summary(out)
p_bi <- plot(out, draw = FALSE)
p_bi$data$plot

## End(Not run)
```

bias_iteration_report *Build a bias-iteration report (FACETS Table 9: iteration / convergence trace)*

Description

This report is NOT an alias of `bias_interaction_report()` despite the similar name. It focuses on the **recalibration path** of a bias run: iteration table, convergence summary, and orientation review. Use this to confirm that the bias recalibration itself converged; use `bias_interaction_report()` to review the ranked flagged cells from the converged run.

Usage

```
bias_iteration_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  top_n = 10
)
```

Arguments

`x`	Output from `estimate_bias()` or `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()` (used when `x` is fit).
`facet_a`	First facet name (required when `x` is fit and `interaction_facets` is not supplied).
`facet_b`	Second facet name (required when `x` is fit and `interaction_facets` is not supplied).
`interaction_facets`	Character vector of two or more facets.
`max_abs`	Bound for absolute bias size when estimating from fit.
`omit_extreme`	Omit extreme-only elements when estimating from fit.
`max_iter`	Iteration cap for bias estimation when `x` is fit.
`tol`	Convergence tolerance for bias estimation when `x` is fit.
`top_n`	Maximum number of iteration rows to keep in preview-oriented summaries. The full iteration table is always returned.

Details

This report focuses on the recalibration path used by `estimate_bias()`. It provides a package-native counterpart to legacy iteration printouts by exposing the iteration table, convergence summary, and orientation review in one bundle.

Value

A named list with:

- `table`: iteration history
- `summary`: one-row convergence summary
- `orientation_review`: interaction-facet sign review
- `settings`: resolved reporting options
- `direction_note`: one-line interpretive note describing which direction the iteration moved (carried from the bias estimator; empty string when the underlying estimator does not emit one)
- `recommended_action`: one-line recommended action label (e.g. `"converged"`, `"increase max_iter"`); empty string when the underlying estimator does not emit one

See Also

`estimate_bias()`, `bias_interaction_report()`, `build_fixed_reports()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("bias_iteration_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
out <- bias_iteration_report(fit, diagnostics = diag, facet_a = "Rater", facet_b = "Criterion")
summary(out)

## End(Not run)
```

---

bias_pairwise_report *Build a conditional bias pairwise-contrast report (Table 14 style)*

Description

Compare each target element's local measure across pairs of contexts. With target `Rater` and context `Criterion`, a row compares the same rater's local severity on two criteria. With target `Criterion` and context `Rater`, it compares the same criterion's local difficulty across two raters. The calculation uses conditional bias cells and does not establish numerical equivalence to FACETS Table 14.

Usage

```
bias_pairwise_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  target_facet = NULL,
  context_facet = NULL,
  top_n = 50,
  p_max = 0.05,
  sort_by = c("abs_t", "abs_contrast", "prob"),
  df_method = c("facets", "classical")
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
max_abs	Bound for absolute bias size when estimating from fit.
omit_extreme	Omit extreme-only elements when estimating from fit.
max_iter	Iteration cap for bias estimation when x is fit.
tol	Convergence tolerance for bias estimation when x is fit.
target_facet	Facet whose local contrasts should be compared across the paired context facet. Defaults to the first interaction facet.
context_facet	Facet whose levels supply the two contexts. Must be the other facet in a 2-way interaction.
top_n	Maximum number of ranked rows to keep.
p_max	Screening tail-area cutoff. An available row is flagged when abs(t) >= 2 OR Prob. <= p_max; this is not a p-value-only test.
sort_by	Ranking key: "abs_t", "abs_contrast", or "prob".
df_method	Pairwise df convention: "facets" (modified Rasch-Welch, integer part, matching FACETS 4.5.0) or "classical" (former continuous df).

Details

This helper exposes the pairwise contrast table that was previously only reachable through fixed-width output generation. It is available only for 2-way interactions. Rows without available orientation or precision are retained with missing screening results. `top_n` limits displayed rows; the summary always describes the complete pairwise family.

Value

A named list with:

- `table`: pairwise contrast rows
- `summary`: one-row contrast summary
- `orientation_review`: interaction-facet sign review
- `settings`: resolved reporting options
- `direction_note`: one-line interpretive note describing the dominant pairwise-contrast direction (carried from the underlying bias estimator; empty string when not applicable)
- `recommended_action`: one-line recommended-action label (e.g. routing the user to follow-up review of the largest flagged pairs); empty string when the underlying estimator does not emit one

Interpreting output

- `table`: one row per target element and context pair, with `Target`, `Context1`, `Context2`, `Local Measure1`, `Local Measure2`, `Contrast`, `SE`, `t`, `d.f.`, `Prob.`, `Flag`, and calculation/status metadata. The default sort places the largest absolute t first and unavailable values last.
- `summary`: full-family `Contrasts`, `ScreenAvailable`, `ScreenUnavailable`, `Flagged`, `MeanAbsContrast`, and `MeanAbsT`. `Flagged` counts known screen-positive rows and is missing when none can be screened. A zero count does not clear unavailable comparisons.
- `orientation_review` carries the same facet-orientation sign review as the parent `estimate_bias()` run.
- Flags are unadjusted exploratory screens. This report does not apply multiplicity correction or establish the absence of differential bias.

Typical workflow

1. Fit and diagnose the model.
2. Run `estimate_bias()` to get the underlying interaction effects.
3. Pass that result to `bias_pairwise_report()` with the intended target and context facets.
4. Read `out$summary` for availability and `out$table` for direction, magnitude and uncertainty of the local comparisons.
5. For the ranked flagged-cells view (which (rater, item) pairs have the largest local bias), use `bias_interaction_report()` on the same `estimate_bias()` output.

Standard-error caveat

The contrast standard error is computed as `SE(b_i - b_j) = sqrt(SE_i^2 + SE_j^2)` – the independence approximation conditional on the calibration. The general variance of a difference also subtracts `2 * Cov(b_i, b_j)`: a negative covariance increases that variance, while a positive covariance decreases it. Neither its sign nor its magnitude is established here. There is no general claim that these screens are conservative. The cell shifts are estimated separately with fixed baseline parameters, not by a jointly estimated sum-to-zero bias model.

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press.
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly, 2*(3), 197-221.
- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement, 4*(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement, 5*(2), 189-227.

Pairwise calculation

For each target element, compare its local measure in two contexts. If the target has predictor sign `s`, its local measure is `target + bias/s`; the contrast is Context1 minus Context2. A negatively oriented rater therefore has local severity `severity - bias`. `TargetSign` comes from the saved bias result's `orientation_review`. Missing orientation withholds local measures, contrasts and tail areas; it is not inferred from a facet name.

`SE1` and `SE2` are conditional cell-information SEs with baseline calibration held fixed. The contrast SE is `sqrt(SE1^2 + SE2^2)` under conditional independence of the two cells. Calibration uncertainty and cross-cell covariance are not propagated. `Target S.E.` describes the original target estimate separately and is not added to the conditional local SE; a fixed target can still have uncertain local bias.

The default `df_method = "facets"` uses the modified Rasch-Welch formula described in the FACETS t-statistics manual (von Davier, 2025). Writing `v1 = SE1^2`, `v2 = SE2^2` and `n1`, `n2` for weighted counts, its df is `(v1+v2)^2 * (n1+n2-2) / ((n1+n2+2) * (v1^2/(n1+1)+v2^2/(n2+1)))`. The tail area uses its integer part, matching executed FACETS 4.5.0 unequal-count examples. Values are stabilized to ten decimal places before truncation to avoid numerical noise at integer boundaries. `DFUnrounded` retains the original continuous value. `df_method = "classical"` retains the former Welch-Satterthwaite reference with `n1-1`, `n2-1` and no integer truncation. `DFMethod` and `DFBasis` identify the choice; equal counts alone cannot distinguish the formulas. `ObsN1` and `ObsN2` are raw row counts. Both weighted counts must exceed one, both SEs must be positive and finite, and the final df must be positive. Otherwise the screen remains unavailable. Fractional weights follow the same rule but are outside the executed comparison. These are conditional screens, not calibrated fairness tests or general FACETS equivalence.

Saved pairwise results

Current tables carry `CalculationVersion = 3`. Recreate older fixed or pairwise reports using `build_fixed_reports()` or `bias_pairwise_report()`. If the bias bundle predates baseline-score

corrections or lacks orientation, recreate estimate_bias() from the saved fit and current diagnostics first. No main calibration refit is needed. Keep historical values separately; summaries and fixed-report plots reject older pairwise calculations.

See Also

estimate_bias(), bias_interaction_report(), build_fixed_reports()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("bias_pairwise_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
out <- bias_pairwise_report(fit, diagnostics = diag, facet_a = "Rater", facet_b = "Criterion")
s <- summary(out)
s$summary
# Check ScreenUnavailable before reading the known Flagged count.
# Zero flags do not establish equivalence or absence of bias.
head(out$table)
# For target Rater, compare that rater's local severity across criteria.
# Read Context1 minus Context2 and the conditional SE before follow-up.

## End(Not run)
```

---

bootstrap_mfrm_gpcm *Parametric bootstrap for GPCM slopes or a matched PCM comparison*

Description

Simulate Persons and scores on the analyzed rating assignment and refit the model. With no null_fit, saved draws support slope intervals. With a matched PCM null_fit, simulate under PCM and refit both models for a bootstrap LRT.

Usage

```
bootstrap_mfrm_gpcm(fit, nsim = 499L, seed, null_fit = NULL)

## S3 method for class 'mfrm_gpcm_bootstrap'
confint(
  object,
  parm = "slopes",
  level = 0.95,
  scale = c("relative", "standardized"),
```

```
  contrasts = NULL,
  contrast_scale = c("ratio", "difference"),
  simultaneous = c("none", "bonferroni"),
  ...
)

## S3 method for class 'mfrm_gpcm_bootstrap'
summary(object, ...)

## S3 method for class 'mfrm_gpcm_bootstrap'
print(x, ...)
```

Arguments

`fit`	A native GPCM MML fit with eligible local information.
`nsim`	Planned datasets (default 499, minimum 2). Each needs one or two refits. Small values demonstrate mechanics, not accurate tail probabilities.
`seed`	Required nonnegative integer. The caller's RNG state is restored.
`null_fit`	Optional PCM MML fit accepted by `compare_mfrm()` with `nested = TRUE`. This changes the target to an equal-slope LRT; its draws must not be used as confidence intervals around the alternative GPCM fit.
`object, x`	A saved `mfrm_gpcm_bootstrap` result.
`parm`	Must be `"slopes"`.
`level`	Nominal confidence level.
`scale, contrasts, contrast_scale, simultaneous`	As in `confint.mfrm_fit()`.
`...`	Unused.

Details

One ability is drawn per Person from the fitted conditional-normal population, shared by all that Person's rows. Facets and covariates are fixed. The observed assignment is preserved; omitted ratings remain omitted. No assignment or missingness mechanism is simulated. Population coefficients and variance are reestimated if estimated in the source model. Retained formula, factor coding and person data must reproduce the population design.

`confint()` uses basic bootstrap errors on the log scale for slopes/ratios, and the identity scale for differences. The point estimate minus reversed empirical error quantiles (type 1) supplies the limits. Failed refits and rejected returned estimates are never removed or replaced: their unknown errors are placed at both extremes to enclose the empirical limits for every completion. Limits can be zero, infinite or unbounded. `availability` and `expected_tail_draws` describe this uncertainty. Bonferroni covers only the requested finite family, assuming adequate marginal bootstrap approximations; it is not a coverage guarantee.

For an LRT, `test` reports `(1 + exceedances)/(nsim + 1)`. If any replicate test is unresolved, `PValue` is missing and `PValueLower`/`PValueUpper` bound all completions. Monte Carlo binomial bounds and resolution are also retained. This is fitted-model calibration, not an exact finite-sample test or a remedy for misspecification, dependence between Persons, or informative assignment.

Increasing nsim improves simulation precision, not the underlying model. A profile-likelihood method is not used by this function.

Value

A serializable result containing source, optional null_fit, all trials with errors/warnings and seeds, draws on the free parameter scale, and settings. refit_draws additionally retains returned alternative-model parameter vectors before eligibility checks; rejected rows are diagnostic values, never inputs to confint(). trials identifies the last stage and whether each model refit returned. checks and source_checks retain category counts/states, numerical status and the information diagnostics already computed; missing fields mean not recorded, not a successful check. No extra information calculation is performed for recording. Older saved results may lack these fields. LRT results also have comparison and test. Printing or changing the interval level never refits. Failed replicates remain present. summary() returns planned/available/unresolved counts, settings and either the saved likelihood-ratio test or default slope intervals, plus retained numerical cautions. It does not refit or discard unresolved trials. If a saved analysis records selected refit updates in settings$recheck, printing and slope intervals retain a caution that it is not a complete rerun under one procedure. Sampling tables preserve that record and any settings$diagnostic_checks_scope. Missing history means not recorded; it does not certify that all draws used the currently installed estimator. After changing estimation or acceptance rules, run a separate complete bootstrap to evaluate the changed procedure; retain the original result.

Eligibility and unresolved replicates

A returned fit is not the same as an accepted replicate. For slope bootstrapping, a singleton score category (observed once) is a warning, not by itself a reason to exclude the point estimate. This applies to the source fit and refits only when a fresh category audit confirms all score categories are observed in every step scope, with no unsupported step coordinates or category contrasts. Model identity, numerical convergence, reevaluated likelihood/gradient and unregularized joint-information checks must still pass. The saved fit's readiness is not changed.

Basic bootstrap quantiles do not themselves require a variance estimate for every replicate. The retained information check is a conservative solution-quality restriction, not a requirement of the basic formula. Singular information or unstable numerical refinement remains unresolved, with returned estimates saved for diagnosis. Positive ill-conditioned information can be used with a caution after refinement, unregularized inversion and scaled-gradient checks pass. This is distinct from a random-effect variance estimated at zero; neither proves optimization failed by itself. Wald intervals, information-criterion comparisons and LRT eligibility share the numerical information review but retain their other checks.

checks separates BootstrapEligible from WaldEligible and retains BootstrapCaution. Accepted singleton or weak-information cases produce an aggregate warning; cautions also follow confint(), its printed output, default plot subtitle, APA tables and reports. A custom subtitle (including NULL) overrides the plot text, not the saved diagnostics. source_checks records the source decision. Use apa_table(result, which = "checks") to inspect refits. InformationRefinementVerified, InformationRelativeChange, InformationInverseResidual and InformationScaledGradient record a numerical refinement when needed; missing values mean it was not run. Do not drop unresolved replicates or substitute diagnostic refit_draws for accepted draws. A larger nsim does not remove an unresolved estimation problem or guarantee coverage.

PopulationSD, MinimumStandardizedSlope and MaximumStandardizedSlope describe the returned optimizer solution. Standardized slopes multiply relative slopes by the fitted population SD (one for a fixed standard-normal population). With covariates, this is the residual population SD. These diagnostics help distinguish scale changes from a slope approaching zero; they do not certify a boundary solution or label rater quality. A missing field means the required estimate was not retained. A near-zero slope, an unused score category and an ill-conditioned information matrix can have different consequences for different parameters: stable slopes do not establish finite thresholds or valid Wald intervals. Use checks to identify cases needing further numerical review before interpreting an unresolved case. APA check tables display scale, slope, gradient and information diagnostics with significant digits so that small positive values are not rounded to zero. The original checks fields remain unrounded numeric values.

References

Chalmers, R. P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. Journal of Statistical Software, 48(6), 1–29. doi:10.18637/jss.v048.i06.

Davison, A. C., and Hinkley, D. V. (1997). Bootstrap Methods and Their Application. Cambridge University Press, Chapter 5.

See Also

confint.mfrm_fit(), compare_mfrm(), mml_quadrature_sensitivity()

Examples

```
# After fitting compatible MML models:
# boot <- bootstrap_mfrm_gpcm(gpcm_fit, nsim = 499, seed = 92401)
# confint(boot, scale = "standardized")
# test <- bootstrap_mfrm_gpcm(gpcm_fit, nsim = 499, seed = 92402,
#                             null_fit = pcm_fit)
# test$test
```

---

build_apa_outputs *Build APA text outputs from model results*

Description

Build APA text outputs from model results

Usage

```
build_apa_outputs(
  fit,
  diagnostics,
  bias_results = NULL,
  context = list(),
  whexact = FALSE
)
```

```
## S3 method for class 'mfrm_apa_outputs'
summary(object, top_n = 3, preview_chars = 160, ...)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Output from diagnose_mfrm().
bias_results	Optional output from estimate_bias().
context	Optional named list for report context.
whexact	Use exact ZSTD transformation.
object	Output from build_apa_outputs().
top_n	Maximum non-empty lines shown in each component preview.
preview_chars	Maximum characters shown in each preview cell.
...	Reserved for generic compatibility.

Details

context is an optional named list for narrative customization. Frequently used fields include:

- assessment, setting, scale_desc
- rater_training, raters_per_response
- rater_facet (used for targeted reliability note text)
- line_width (optional text wrapping width for report_text; default = 92)

Output text includes residual-PCA screening commentary if PCA diagnostics are available in diagnostics.

For GPCM, this helper returns a caveated partial reporting bundle over supported diagnostics, direct tables, and plots. It also includes a gpcm_boundary table. Treat the output as slope-aware sensitivity-reporting text, not FACETS score-side equivalence, automatic operational scoring, or design-forecasting evidence.

By default, report_text includes:

- model/data design summary (N, facet counts, scale range)
- optimization/convergence metrics (Converged, Iterations, LogLik) plus the eligible canonical MML AIC/Person-BIC/SABIC panel or its explicit ineligibility and legacy-descriptive status; q<31 MML criteria are explicitly labelled screening/review-only rather than automatic ranking
- anchor/constraint summary (noncenter_facet, centering convention, anchored levels, group anchors, dummy facets)
- latent-regression population-model wording when fit has an active population_formula
- category/threshold diagnostics (including disordered-step details when present)
- overall fit, misfit count, and top misfit levels
- facet reliability/separation, residual PCA summary, and bias-screen counts

- stored inter-rater agreement, its matched-opportunity denominator and unavailable outcomes, separately from separation reliability

Agreement wording uses the rater facet in the stored diagnostics summary; `context$rater_facet` does not recalculate or relabel that summary. The model-expected baseline uses fitted category probabilities with calibration fixed, not marginal-frequency chance agreement. Adjacent agreement includes exact matches and differences of at most 1 on the compared score scale. Missing expected values remain explicit, including the stored number of opportunities without valid probabilities. An absent summary does not establish agreement. See `interrater_agreement_table()` for the matching, repeated-rating and denominator conventions.

`summary()` method: Summarize APA report-output bundles

For `summary()`: This summary is a diagnostics layer for APA text products, not a replacement for the full narrative.

It reports component completeness, line/character volume, and a compact preview for quick QA before manuscript insertion.

Value

An object of class `mfrm_apa_outputs` with:

- `report_text`: APA-oriented Method/Results draft prose
- `decision`: plain-language source-fit interpretation plus the separate precision-contract decision; `FormalInference` is `"Yes"` only when both fit readiness and `contract$precision$supports_formal_inferen` support it
- `fit_readiness`, `fit_readiness_components`, and `fit_readiness_parameters`: exact source-fit readiness provenance
- `table_figure_notes`: consolidated draft notes for tables/visuals
- `table_figure_captions`: draft caption candidates without figure numbering
- `section_map`: package-native section table for manuscript assembly
- `contract`: structured APA reporting contract used for downstream checks

For `summary()`: An object of class `summary.mfrm_apa_outputs`.

Interpreting output

- `report_text`: manuscript-draft narrative covering Method (model specification, estimation, convergence) and Results (global fit, facet separation/reliability, misfit triage, category diagnostics, residual-PCA screening, bias screening), organized as an APA-oriented, third-person Method/Results draft. It is intended for human review and is not a claim of complete APA 7 or JARS compliance.
- `table_figure_notes`: reusable draft note blocks for table/figure appendices.
- `table_figure_captions`: draft caption candidates aligned to generated outputs.
- active latent-regression fits add a population-model section and Table 5 notes/captions that distinguish conditional-normal coefficient reporting from post hoc regression on EAP/MLE scores.

When bias results or PCA diagnostics are not supplied, those sections are omitted from the narrative rather than producing placeholder text. Reporting reuses stored residual PCA results and does not compute omitted overall or facet-specific analyses. Request the intended scope with `diagnose_mfrm(..., residual_pca = "overall")`, `"facet"`, or `"both"` first.

For `summary()`:

- `overview`: total coverage across standard text components.
- `decision`: the source-fit decision shown before draft-completeness checks; text completeness cannot promote a review or blocked fit.
- `components`: per-component density and mention checks (including residual-PCA mentions).
- `sections`: package-native section coverage table.
- `content_checks`: contract-based alignment checks for APA drafting readiness.
- `overview$DraftContractPass`: the primary contract-completeness flag for draft text components.
- `overview$ReadyForAPA`: a backward-compatible alias of that contract flag, not a certification of inferential adequacy.
- `preview`: first non-empty lines for fast visual review.

Typical workflow

1. Build diagnostics (and optional bias results). For `RSM` / `PCM` reporting runs, prefer an `MML` fit and `diagnose_mfrm(..., diagnostic_mode = "both")`.
2. Run `build_apa_outputs(...)`.
3. Check `summary(apa)` for completeness.
4. Insert `apa$report_text` and note/caption fields into manuscript drafts after checking the listed cautions.

For `summary()`:

1. Build outputs via `build_apa_outputs()`.
2. Run `summary(apa)` to screen for empty/short components.
3. Use `apa$report_text`, `apa$table_figure_notes`, and `apa$table_figure_captions` as draft components for final-text review.

Context template

A minimal `context` list can include fields such as:

- `assessment`: name of the assessment task
- `setting`: administration context
- `scale_desc`: short description of the score scale
- `rater_facet`: rater facet label used in narrative reliability text

Input validation

`fit` must be an `mfrm_fit` object from `fit_mfrm()`. `diagnostics` must be an `mfrm_diagnostics` object from `diagnose_mfrm()`. `context` must be a list (use `NULL` or `list()` for no extra context). If supplied, `bias_results` must come from `estimate_bias()` or another package-native bias helper that provides a table component.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model.

See Also

`build_visual_summaries()`, `estimate_bias()`, `reporting_checklist()`, mfrmr_reporting_and_apa

`build_apa_outputs()`, `summary()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_apa_outputs", run.dontrun = TRUE)
## Not run:
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Use the diagnostics saved with this fit
diagnostics <- example_fit$diagnostics

# Prepare draft wording from the fitted model and diagnostics
apa <- build_apa_outputs(
  fit,
  diagnostics = diagnostics,
  context = list(
    assessment = "Synthetic writing assessment",
    setting = "Demonstration dataset",
    scale_desc = "1-4 rating scale",
    rater_facet = "Rater"
  )
)

# Check which sections are supported, then read the draft
apa_review <- summary(apa)
apa_review$content_checks
cat(apa$report_text)
# Adapt the text to the study question, design, and evidence before using it

## End(Not run)
```

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_apa_outputs", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
apa <- build_apa_outputs(fit, diag)
summary(apa)

## End(Not run)
```

`build_conquest_overlap_bundle` *Build a scoped ConQuest-overlap bundle*

Description

Build a scoped ConQuest-overlap bundle

Usage

```
build_conquest_overlap_bundle(
  fit = NULL,
  case = c("synthetic_latent_regression"),
  output_dir = NULL,
  prefix = "conquest_overlap",
  overwrite = FALSE,
  quad_points = 7L,
  maxit = 40L,
  reltol = 1e-09,
  conquest_control = list()
)
```

Arguments

`fit`	Optional output from `fit_mfrm()` or `run_mfrm_facets()`. When omitted, the helper builds the package's `"synthetic_latent_regression"` overlap case.
`case`	Overlap case used when `fit = NULL`. Currently only `"synthetic_latent_regression"` is supported.
`output_dir`	Optional directory where the bundle files should be written. When `NULL`, the helper returns the in-memory bundle only.
`prefix`	File-name prefix used when writing the bundle to disk.
`overwrite`	If `FALSE`, refuse to overwrite existing files.
`quad_points`	Quadrature points used when `fit = NULL` and the overlap case is fit on the fly.

`maxit` Maximum optimizer iterations used when `fit = NULL`. The fitted object's actual value is recorded in the returned summary and settings.

`reltol` Relative convergence tolerance used when `fit = NULL`. Default `1e-9`. The fitted object's actual value is recorded in the returned summary and settings.

`conquest_control` Named list of external calculation controls: `nodes = 31L`, `minnode = -6`, `maxnode = 6`, `p_nodes = 2000L`, `seed = 2L`, `convergence = 1e-8`, `deviance_change = 1e-10`, and `iterations = 2000L`. These do not change the native fit. Nodes/range control calibration; `p_nodes` and `seed` control posterior approximation.

Details

This helper prepares a narrow ConQuest comparison bundle for an `RSM` / `PCM` latent-regression `MML` fit and records the `mfrmr`-side tables to compare after an external ConQuest run. The supported overlap is intentionally narrow:

- ordered-response `RSM` / `PCM` only;
- binary responses only;
- exactly one difficulty-oriented non-person facet, treated as the item facet, with unanchored sum-to-zero locations and default binary steps;
- active latent-regression `MML` with estimated residual variance; an explicitly fixed `population_sd` requires a separately matched model;
- exactly one numeric person covariate beyond the intercept;
- complete person-by-item rectangular data with unit weights;
- no group anchors, dummy facets, interactions or shrinkage.

Unsupported settings are rejected before export; they are not silently converted to the template's model. Retained category support must be binary, even if a polytomous fit happens to have only two observed score values.

The returned bundle standardizes the responses to `{0, 1}`, pivots them to a one-row-per-person wide CSV, stores the corresponding person covariates, and records the `mfrmr` estimates that should be compared externally. The wide CSV uses safe person aliases and the covariate alias `X`. `person_map` and `covariate_map` preserve original labels and are exported alongside the data. `PersonDataVariable` and `CaseVariable` record the covariate columns in `person_data` and `mfrmr_case_eap`; `PopulationParameter` records its coefficient label in `mfrmr_population`. Suffixes avoid overwriting identifier, estimate or variance fields. Keep the bundle for reverse mapping of ConQuest exports. Aliases do not deidentify the bundle, which includes original person data. Older bundles without maps retain their original-ID convention. It also records the actual `mfrmr` optimizer controls, MML engine, terminal gradient, convergence status and severity, and inference-readiness decision. A fit that is not inference-ready remains available for descriptive numerical review. Inspect the source fit's readiness checks and supported inferential scope: lack of readiness need not mean optimizer failure, and convergence alone does not establish readiness. The comparison does not validate standard errors or confidence intervals. The generated ConQuest benchmark template defaults its parameter-change criterion at `1e-8`, deviance-change criterion at `1e-10`, and iteration ceiling at `2000`; these controls are written into the command, summary, and settings so a default stopping rule cannot masquerade as an objective discrepancy. External integration controls are independent of native `quad_points`: equal counts do not imply equal quadrature rules or precision. Check calibration refinement and

range sensitivity separately from posterior scoring. Neither 31 nodes nor a fixed seed guarantees accuracy.

The `conquest_command` component is a conservative starting template, not a guaranteed version-invariant automation. The `conquest_output_contract` component records which requested external output should feed each normalized review table. Use `normalize_conquest_overlap_exports()` for the parameter, regression, covariance, and case-EAP CSV files requested by the generated command. The additional matrixout history CSV retains the objective trajectory and an independent estimated-parameter dimension for release verification; it is not parsed as a free-form report. `normalize_conquest_overlap_files()` and `normalize_conquest_overlap_tables()` remain available for already-extracted custom tables. Then use `review_conquest_overlap()` only after the matching ConQuest run has been executed externally. The bundle and command template alone are not external validation evidence.

This is a controlled analysis bundle, not a deidentified or automatically shareable export. Response files contain person identifiers and responses; the person-data file contains identifiers and covariates; and case-EAP files contain identifiers and person-level estimates. When files are written, the helper emits a warning and writes an artifact-level privacy notice. Apply the study's data-handling policy before sharing or moving any bundle file.

Value

A named list with class `mfrm_conquest_overlap_bundle`.

Running ConQuest locally

This function writes inputs; it does not launch or install ConQuest. Open the generated `.cqc` file in your local ConQuest installation and use its containing directory as the working directory, so the relative CSV paths resolve. A native command-line installation can also read that command file as standard input from the same directory. Check the ConQuest console for successful estimation and the requested exports before normalizing them. ConQuest is optional and is not needed for the package's examples or tests.

ConQuest's EAP calculation uses Monte Carlo posterior integration with separate `p_nodes` and `seed` controls. The `estimate` command's quadrature node count does not set this posterior simulation budget. Record both controls when comparing Person scores; close calibration estimates do not imply identical EAPs. See the ConQuest command reference, `set` and `show`. The template defaults to `p_nodes=2000` and `seed=2`; these give reproducibility, not a guarantee of adequate scoring precision. Use `conquest_control` to increase the posterior integration budget when checking score accuracy, and retain the command with its outputs. The command begins with `reset;`, then sets `lconstraints=items` and `exit_on_error=yes`. Run it in a fresh ConQuest session: reset clears prior analysis settings and the output buffer. It fixes response codes to 0/1. The template uses `fit=no` and `stderr=quick`; it does not compare fit or full parameter-estimate covariance. `export covariance` contains the population variance, not parameter-estimate sampling covariance.

Comparison targets

- regression slope: compare after confirming identical covariate coding and population-model parameterization;

- residual variance `sigma2`: compare after confirming the same latent-scale and variance parameterization;
- item estimates: compare after centering because the Rasch location origin remains constraint-dependent;
- case EAP estimates: compare as posterior summaries under the fitted population model.

Output

The returned object has class `mfrm_conquest_overlap_bundle` and includes:

- `summary`: one-row scope summary with posterior-basis, population-model, optimizer-control, and convergence-review fields
- `comparison_targets`: comparison rules for the exported tables
- `conquest_output_contract`: requested ConQuest outputs and review handoff
- `response_long`: long-format binary response data used by the bundle
- `response_wide`: wide CSV-ready response matrix for the ConQuest template
- `person_data`: one-row-per-person covariate table
- `person_map`: original Person labels and safe ConQuest PID aliases
- `covariate_map`: original covariate name and safe ConQuest variable alias
- `item_map`: mapping from exported response columns to original item levels
- `mfrmr_population`: fitted population-model coefficients plus `sigma2`
- `mfrmr_item_estimates`: fitted item estimates with centered values
- `mfrmr_case_eap`: posterior EAP summaries for the fitted persons
- `conquest_command`: conservative ConQuest command template
- `written_files`: file inventory when `output_dir` is supplied
- `privacy_notice`: artifact-level sensitive-data inventory
- `settings`: bundle settings, including the actual `mfrmr` fit controls and convergence state
- `notes`: interpretation notes

See Also

`normalize_conquest_overlap_files()`, `normalize_conquest_overlap_tables()`, `review_conquest_overlap()`, `reference_case_benchmark()`, `build_mfrm_replay_script()`, `export_mfrm_bundle()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_conquest_overlap_bundle", run.dontrun = TRUE)
## Not run:
# Synthetic latent-regression fit; the full regeneration recipe is
# examples/measurement-fits.R in the package (31 nodes, maxit = 2000).
fit <- readRDS(system.file("examples", "measurement-fits.rds",
  package = "mfrmr", mustWork = TRUE))$conquest
bundle <- build_conquest_overlap_bundle(fit = fit)
bundle$summary[, c("Case", "Facet", "Covariate", "Persons", "Items")]
```

```
summary(bundle)$mfrmr_fit_status
summary(bundle)$conquest_command_scope
summary(bundle)$conquest_output_contract
cat(substr(bundle$conquest_command, 1, 120))

## End(Not run)
```

`build_equating_chain` *Build a screened linking chain across ordered calibrations*

Description

Links a series of calibration waves by computing mean offsets between adjacent pairs of fits. Common linking elements (e.g., raters or items that appear in consecutive administrations) are used to estimate the scale shift. Cumulative offsets express all waves relative to the first wave, conditional on the common-element assumptions. The procedure is a practical screened linking aid, not a full general-purpose equating framework or proof of common-scale comparability.

Usage

```
build_equating_chain(
  fits,
  anchor_facets = NULL,
  include_person = FALSE,
  drift_threshold = 0.5
)

## S3 method for class 'mfrm_equating_chain'
print(x, ...)

## S3 method for class 'mfrm_equating_chain'
plot(
  x,
  y = NULL,
  type = c("common_anchors", "graph", "chain", "links", "anchor_removal",
    "offset_sensitivity"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  show_title = FALSE,
  show_notes = FALSE,
  ...
)

## S3 method for class 'mfrm_equating_chain'
summary(object, ...)

## S3 method for class 'summary.mfrm_equating_chain'
print(x, ...)
```

Arguments

fits	Named list of mfrm_fit objects in chain order.
anchor_facets	Character vector of facets to use as linking elements.
include_person	Include person estimates in linking.
drift_threshold	Non-negative scalar threshold for screening and flagging large residuals in links. Inf retains all finite differences and disables magnitude flagging.
x	An mfrm_equating_chain object.
y	Unused (S3 plot signature requirement).
type	One of "graph" (bipartite wave x link-specific common-element graph), "common_anchors" (default; bar chart of common-anchor counts per wave pair), "chain" (cumulative offsets), "links" (recorded wave comparisons), or "anchor_removal" (newly disconnected wave pairs after deleting each common element), or "offset_sensitivity" (conditional rescreened linking-offset changes).
preset	Visual preset.
draw	If TRUE, draw the plot with base graphics.
show_title	Logical; display the main title.
show_notes	Logical; display graph/topology interpretation footers. Returned notes and ordinary R warnings are unaffected. Axes, legends and data labels remain visible; other chain views have no interpretation footer.
...	Ignored.
object	An mfrm_equating_chain object (for summary).

Details

The screened linking chain uses a screened link-offset method. For each pair of adjacent waves (A, B), the function:

1. Identifies common linking elements (facet levels present in both fits).
2. Computes per-element differences:

$$d_e = \hat{\delta}_{e,B} - \hat{\delta}_{e,A}$$

3. Computes a preliminary link offset using the inverse-variance weighted mean of these differences when standard errors are available (otherwise an unweighted mean).
4. Screens out elements whose residual from that preliminary offset exceeds drift_threshold, then recomputes the final offset on the retained set.
5. Records Offset_SD (standard deviation of retained residuals) and Max_Residual (maximum absolute deviation from the mean) as indicators of link quality.
6. Flags links with fewer than 5 retained common elements in any linking facet as having thin support.

The five-element rule is a package screening convention, not a universal adequacy threshold. Matching labels are assumed to identify the same invariant elements, and input fits are assumed to use compatible model, score, orientation, and population conventions. The helper does not verify those assumptions or source-fit readiness. In particular, adjacent label overlap and a finite offset do not by themselves establish a common scale.

Each adjacent `Offset` is one value pooled across all selected facets. When any common rows have finite positive SEs, only those rows contribute to the inverse-variance weighted offset; retained/support counts can still include rows without usable SEs. If preliminary screening would remove every row, the implementation falls back to all finite differences. Prefer one substantively coherent `anchor_facets` block unless a common shift across facets is justified.

Cumulative offsets are computed by chaining link offsets from Wave 1 forward, placing all waves onto the metric of the first wave. `Offset_SD` is retained-element residual spread, not the standard error of `Offset`. No offset SE, confidence interval, cross-fit covariance, or cumulative uncertainty propagation is currently returned, so uncertainty can compound along the chain without appearing in `cumulative`.

Elements whose per-link residual exceeds `drift_threshold` are flagged in `$element_detail$Flag`. A high `Offset_SD`, many flagged elements, or a thin retained anchor set signals an unstable link that may compromise the resulting scale placement.

Value

Object of class `mfrm_equating_chain` with components:

links Tibble of link-level statistics (offset, SD, etc.).

cumulative Tibble of cumulative offsets per wave.

element_detail Tibble of element-level linking details.

common_by_facet Tibble of retained common-element counts by facet.

config List of analysis configuration.

Which function should I use?

- Use `anchor_to_baseline()` for a single new wave anchored to a known baseline.
- Use `detect_anchor_drift()` when you want direct comparison against one reference wave.
- Use `build_equating_chain()` when no single wave should dominate and you want ordered, adjacent links across the series.

Interpreting output

- `$links`: one row per adjacent pair with `From`, `To`, `N_Common`, `N_Retained`, `Offset_Prelim`, `Offset`, `Offset_SD`, and `Max_Residual`. Small `Offset_SD` relative to the offset indicates a consistent shift across elements. `LinkSupportAdequate = FALSE` means at least one linking facet retained fewer than 5 common elements after screening.
- `$cumulative`: one row per wave with its cumulative offset from Wave 1. Wave 1 always has offset 0.
- `$element_detail`: per-element linking statistics (estimate in each wave, difference, residual from mean offset, and flag status). Flagged elements may indicate DIF or rater re-training effects.

- $common_by_facet: retained common-element counts by linking facet for each adjacent link.
- $config: records wave names and analysis parameters.
- OffsetStatus / OffsetReason explain unavailable link offsets. ScreeningFallback records use of all finite differences when no row passed preliminary screening. N_Contributing and the element-level Contributing flag identify the rows actually used in the final average; retained rows without usable SEs may not contribute.
- OffsetSDStatus distinguishes available residual spread from fewer than two retained elements, unavailable offsets, or a numerical failure. A singleton has an undefined sample SD, not an estimated SD of zero. Unavailable Max_Residual and flagged counts stay missing rather than representing no drift. Summaries also give the unavailable count and the flagged count among available elements.
- cumulative$Status / Reason distinguish unavailable preceding links from an unrepresentable cumulative sum. Later offsets do not become zero when a link is unavailable.
- Read links before cumulative: weak adjacent links can make later cumulative offsets less trustworthy.

Common-element graph

plot(chain, type = "graph") displays waves as squares and common elements as circles, including waves without retained connections. Each element node belongs to one reviewed link: [L1] refers to link 1 in the returned links table. The same element can have different screening results in different comparisons, so it is repeated per link. Solid lines denote retained, unflagged elements; dot-dash denotes retained elements with a drift flag or an unavailable flag; dashed denotes excluded elements; dotted denotes unknown retention. These distinctions also apply in monochrome.

This is a graph of screened common-element comparisons, not an inventory of fixed anchors or an all-pairs assessment. Retention does not guarantee positive weight in the offset, invariance, adequate support or precision. New chains store positional FromID/ToID in links and LinkID in element_detail; display names may repeat or contain punctuation. Older chains are matched by their complete link labels; ambiguous references require rebuilding the chain.

With draw = FALSE, the plot returns an mfrm_plot_data object. Its $data$data contains nodes, edges, links, elements, n_waves, n_anchors (unique facet/level identities), and n_element_nodes (link-specific nodes). Edge from/to refer to node NodeId; AnchorId identifies the same facet/level across links. Retained, Flag, Status and Reason preserve screening information. RetainedDegree counts retained incidences, not statistical information. Interpretation notes remain in $data$notes and in print() even when show_notes = FALSE. Full labels remain in the node table if the device is too narrow. Base graphics are sufficient; no optional graph package is needed.

Wave links and common-element removal

plot(chain, type = "links") summarizes the recorded wave-to-wave comparisons. Numbers count common elements with Retained = TRUE, including flagged retained elements. A dotted comparison has no recorded retained element; this is not proof that the original rating design is disconnected. Missing retention is counted separately. Dot-dash connections have flagged retained elements or inadequate/unknown recorded link support.

plot(chain, type = "anchor_removal") removes each facet/level identity, one at a time, from all recorded comparisons. Other retention decisions stay fixed. The horizontal axis counts newly disconnected unordered wave pairs, including indirect paths. Pairs already disconnected at baseline

are not counted. Losing a direct link need not disconnect its endpoints if another path remains. Zero means no new graph disconnection, not negligible impact on estimates or adequate linking support. No screening, offsets, estimates, SEs or confidence intervals are recomputed by either view.

Both views return `$data$data$nodes`, `links`, `elements` and `summary`. The removal view additionally returns `removal` (one row per common-element identity), `lost_links` (recorded comparisons losing their last retained element), `lost_pairs` (newly disconnected wave pairs), and `components_after` (baseline and post-removal membership for every wave). Use `AnchorId` to join the removal table to these details; `FromID`/`ToID` and `WaveID` identify waves independently of display labels. Component numbers are local to each partition: compare membership relations, not numeric labels across Before, After or different removal scenarios. Removal counts include `RetainedOccurrences`, `UnknownOccurrences`, `LostLinks`, `ComponentsAfter` and `NewlyDisconnectedPairs`.

These calculations reuse the recorded common-element graph; they do not search for new links or assess statistical influence. The full lost-pair table can be large for chains with many waves and common elements. Compact plots abbreviate long labels and warn when IDs or a larger device are needed; full identities and interpretation notes remain in the returned tables.

Conditional offset sensitivity

`plot(chain, type = "offset_sensitivity")` removes each common `(Facet, Level)` identity from every adjacent link, then reruns the existing offset calculation with its recorded drift threshold. Preliminary offsets, screening, final weighted/unweighted offsets and cumulative offsets are recomputed. The source wave estimates and SEs remain fixed: this is a conditional linking sensitivity calculation, not a refit of item, rater or person parameters or a new uncertainty estimate.

The view requires an ordered adjacent chain with the recorded method, threshold, support guideline, source estimate/SE columns and offsets. Recomputed baseline offsets must agree with recorded offsets within `1e-8 * max(1, abs(recorded_offset))`; inconsistent or incomplete source records require rebuilding the chain. Numerical overflow/underflow that prevents a finite preliminary offset is reported as a numerical failure.

The figure shows each deletion's maximum absolute cumulative offset change over finite comparisons, excluding the fixed-zero first wave. Shapes distinguish complete comparisons, partial comparisons and no finite comparison. An unavailable comparison remains `NA`, not zero. If one link becomes unavailable, later cumulative offsets also become unavailable. Read the detailed tables when a figure marks an incomplete comparison.

The `$data$data` payload contains `settings`, `baseline_links`, `baseline_cumulative`, `baseline_elements`, `removal`, `links`, `cumulative`, `retention_changes` and `common_by_facet`. Join `RemovedAnchorId` in scenario tables to `removal$AnchorId`. Signed `Change` is recalculated minus baseline offset. `Status` distinguishes `computed`, `no_common`, `no_finite_differences` and `numerical_failure`. `N_Contributing` and element `Contributing` identify positive finite weight contributors (or retained rows in an unweighted offset). `retention_changes` records other elements whose retention, contribution or residual flag changes; the deliberately removed element is omitted. Facets whose last element is removed remain in support tables with zero counts. No support failure is converted into a statement of readiness.

`ScreeningFallback` identifies the existing rule that restores all finite differences if preliminary screening would remove every one. SEs may be missing, in which case the existing unweighted fallback is used. No offset SE, confidence interval, cross-fit covariance, or uncertainty propagation is supplied. A small conditional change is not evidence of anchor invariance or negligible change under a full model refit. Notes remain available when figure titles and footers are omitted.

Typical workflow

1. Fit each administration wave separately: `fit_a <- fit_mfrm(...)`.
2. Combine into an ordered named list: `fits <- list(Spring23 = fit_s, Fall23 = fit_f, Spring24 = fit_s2)`.
3. Call `chain <- build_equating_chain(fits)`.
4. Review `summary(chain)` for link quality.
5. Visualize with `plot(chain, type = "chain")`.
6. For problematic links, investigate flagged elements in `chain$element_detail` and consider removing them from the anchor set.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`detect_anchor_drift()`, `anchor_to_baseline()`, `make_anchor_table()`, `plot(x)`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("build_equating_chain", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
people <- unique(toy$Person)
d1 <- toy[toy$Person %in% people[1:24], , drop = FALSE]
d2 <- toy[toy$Person %in% people[25:48], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML")
```

```
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML")
# Check convergence before using the estimates for linking.
summary(fit1)
summary(fit2)
# The four shared levels per facet trigger a thin-support warning.
# Inspect that warning; matching labels alone do not establish a common scale.
chain <- build_equating_chain(list(Form1 = fit1, Form2 = fit2))
summary(chain)
chain$cumulative

## End(Not run)
```

`build_fixed_reports` *Build legacy-compatible fixed-width text reports*

Description

Build legacy-compatible fixed-width text reports

Usage

```
build_fixed_reports(
  bias_results,
  target_facet = NULL,
  branch = c("facets", "original"),
  df_method = c("facets", "classical")
)
```

Arguments

`bias_results`	Output from `estimate_bias()`.
`target_facet`	Optional target facet for pairwise contrast table.
`branch`	Output branch: `"facets"` keeps the legacy-compatible fixed-width layout; `"original"` returns compact sectioned fixed-width text for report drafts.
`df_method`	Pairwise df convention: `"facets"` (modified Rasch-Welch, integer part, matching FACETS 4.5.0) or `"classical"` (former continuous df).

Details

This function generates plain-text, fixed-width output intended to be read in console/log environments or exported into text reports.

The pairwise section (Table 14 style) is only generated for 2-way bias runs. For higher-order interactions (`interaction_facets` length >= 3), the function returns the bias table text and a note explaining why pairwise contrasts were skipped.

Value

A named list with class mfrm_fixed_reports (and a branch-specific subclass mfrm_fixed_reports_<branch>):

- bias_fixed: fixed-width interaction table text
- pairwise_fixed: fixed-width pairwise contrast text
- pairwise_table: underlying pairwise data.frame
- branch: character scalar "original" or "facets" echoing which fixed-width style was rendered
- style: character scalar carrying the resolved style preset used when building the text artifact
- interaction_label: human-readable label for the interaction that drove the bias run ("Rater x Criterion"-style); NA when no bias rows are available
- target_facet: character scalar identifying which facet was used as the target facet for pairwise contrasts; NA when no pairwise contrasts were requested or available

Pairwise calculation

For each target element, compare its local measure in two contexts. If the target has predictor sign s, its local measure is target + bias/s; the contrast is Context1 minus Context2. A negatively oriented rater therefore has local severity severity - bias. TargetSign comes from the saved bias result's orientation_review. Missing orientation withholds local measures, contrasts and tail areas; it is not inferred from a facet name.

SE1 and SE2 are conditional cell-information SEs with baseline calibration held fixed. The contrast SE is sqrt(SE1^2 + SE2^2) under conditional independence of the two cells. Calibration uncertainty and cross-cell covariance are not propagated. Target S.E. describes the original target estimate separately and is not added to the conditional local SE; a fixed target can still have uncertain local bias.

The default df_method = "facets" uses the modified Rasch-Welch formula described in the FACETS t-statistics manual (von Davier, 2025). Writing v1 = SE1^2, v2 = SE2^2 and n1, n2 for weighted counts, its df is (v1+v2)^2 * (n1+n2-2) / ((n1+n2+2) * (v1^2/(n1+1)+v2^2/(n2+1))). The tail area uses its integer part, matching executed FACETS 4.5.0 unequal-count examples. Values are stabilized to ten decimal places before truncation to avoid numerical noise at integer boundaries. DFUnrounded retains the original continuous value. df_method = "classical" retains the former Welch-Satterthwaite reference with n1-1, n2-1 and no integer truncation. DFMethod and DFBasis identify the choice; equal counts alone cannot distinguish the formulas. ObsN1 and ObsN2 are raw row counts. Both weighted counts must exceed one, both SEs must be positive and finite, and the final df must be positive. Otherwise the screen remains unavailable. Fractional weights follow the same rule but are outside the executed comparison. These are conditional screens, not calibrated fairness tests or general FACETS equivalence.

Saved pairwise results

Current tables carry CalculationVersion = 3. Recreate older fixed or pairwise reports using build_fixed_reports() or bias_pairwise_report(). If the bias bundle predates baseline-score corrections or lacks orientation, recreate estimate_bias() from the saved fit and current diagnostics first. No main calibration refit is needed. Keep historical values separately; summaries and fixed-report plots reject older pairwise calculations.

Interpreting output

- `bias_fixed`: fixed-width table of interaction effects.
- `pairwise_fixed`: pairwise contrast text (2-way only).
- `pairwise_table`: structured contrast table.
- `interaction_label`: facets used for the bias run.

Typical workflow

1. Run `estimate_bias()`.
2. Build text bundle with `build_fixed_reports(...)`.
3. Use `summary()`/`plot()` for quick checks, then export text blocks.

Preferred route for new analyses

For new reporting workflows, prefer `bias_interaction_report()` and `build_apa_outputs()`. Use `build_fixed_reports()` when a fixed-width text artifact is specifically required for a compatibility handoff.

See Also

`estimate_bias()`, `build_apa_outputs()`, `bias_interaction_report()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_fixed_reports", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
fixed <- build_fixed_reports(bias)
fixed_original <- build_fixed_reports(bias, branch = "original")
summary(fixed)
p <- plot(fixed, draw = FALSE)
p2 <- plot(fixed, type = "pvalue", draw = FALSE)
if (interactive()) {
  plot(
    fixed,
    type = "contrast",
    draw = TRUE,
    main = "Pairwise Contrasts (Customized)",
    palette = c(pos = "#1b9e77", neg = "#d95f02"),
    label_angle = 45
  )
}
```

```
## End(Not run)
```

`build_linking_review` *Build a linking-review synthesis object*

Description

Build a linking-review synthesis object

Usage

```
build_linking_review(
  anchor_review = NULL,
  drift = NULL,
  chain = NULL,
  top_n = 10
)

## S3 method for class 'mfrm_linking_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`anchor_review`	Optional output from `review_mfrm_anchors()`.
`drift`	Optional output from `detect_anchor_drift()`.
`chain`	Optional output from `build_equating_chain()`.
`top_n`	For `build_linking_review()`: Maximum number of linking-risk rows to highlight in summary outputs. The full object keeps the full risk tables. For `summary()`: Number of top linking-risk rows to keep in the compact summary.
`object`	Output from `build_linking_review()`.
`digits`	Number of digits for printed numeric values.
`...`	Reserved for generic compatibility.

Details

`build_linking_review()` does not recompute anchor, drift, or chain statistics. It is a synthesis layer that organizes package-native evidence into one operational review surface with:

- a front-door status block,
- ranked linking risks,
- explicit next actions,
- plot routing metadata,
- a reporting/export handoff map.

The helper keeps the current conservative interpretation policy: anchor drift and screened links are operational review tools, not automatic proofs of scale equivalence or score comparability. It also does not verify source-fit readiness, cross-wave element identity, invariance, or the external assumptions behind group anchors; these must be established before promoting the synthesized review.

`summary()` method: Summarize a linking-review object

Value

An object of class `mfrm_linking_review`.

For `summary()`: An object of class `summary.mfrm_linking_review`.

Recommended input route

Use existing package-native outputs in this order:

1. `review_mfrm_anchors()` for pre-fit anchor adequacy.
2. `detect_anchor_drift()` for direct wave-to-reference drift screening.
3. `build_equating_chain()` for adjacent screened-link review across waves.

Interpreting output

- `overview`: which evidence sources were supplied and the current review status.
- `top_linking_risks`: primary operational triage table.
- `group_view_index`: stable wave/link/facet/source-family grouping routes.
- `plot_map`: which plotting function to use for each result.
- `reporting_map`: what is covered here versus which manuscript-oriented helper should be used separately.

GPCM boundary

This helper summarizes the documented `RSM` / `PCM` linking workflow. When supplied drift/chain sources resolve to `GPCM`, it retains an explicitly exploratory review and marks that scope as `supported_with_caveat`. This does not establish score interchangeability, invariance or a common scale across administrations.

See Also

`review_mfrm_anchors()`, `detect_anchor_drift()`, `build_equating_chain()`, plot(x), mfrmr_linking_and_dff

`build_linking_review()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("build_linking_review", run.dontrun = TRUE)
## Not run:
# Deliberately linked teaching waves: common labels below represent the
# same rater and criterion identities by construction.
```

```
toy <- load_mfrmr_data("example_core")
people <- unique(toy$Person)
# Two balanced 24-Person waves retain every Rater and Criterion.
d1 <- toy[toy$Person %in% people[1:24], , drop = FALSE]
d2 <- toy[toy$Person %in% people[25:48], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "MML", quad_points = 7, maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "MML", quad_points = 7, maxit = 30)
anchor_review_obj <- review_mfrm_anchors(d1, "Person", c("Rater", "Criterion"), "Score")
drift <- detect_anchor_drift(list(Wave1 = fit1, Wave2 = fit2))
chain <- build_equating_chain(list(Wave1 = fit1, Wave2 = fit2))
review <- build_linking_review(anchor_review = anchor_review_obj, drift = drift, chain = chain)
summary(review)
review$top_linking_risks
review$group_view_index

## End(Not run)
```

`build_mfrm_manifest` *Build a reproducibility manifest for an MFRM analysis*

Description

Build a reproducibility manifest for an MFRM analysis

Usage

```
build_mfrm_manifest(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  include_person_anchors = FALSE,
  data = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()` or `run_mfrm_facets()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. When `NULL`, diagnostics are computed with `residual_pca = "none"`.
`bias_results`	Optional output from `estimate_bias()` or a named list of bias bundles.
`population_prediction`	Optional output from `predict_mfrm_population()`.

`unit_prediction`
: Optional output from `predict_mfrm_units()`.

`plausible_values`
: Optional output from `sample_mfrm_plausible_values()`.

`include_person_anchors`
: If `TRUE`, include person measures in the exported anchor table.

`data`
: Optional original analysis data frame. When supplied, the manifest's `input_summary` row for `data` describes the user's untouched input rather than the package's internal `prep$data` (which carries synthesised `Weight` / `score_k` columns). The summary records structure, not a byte-level or serialized-object identity claim.

Details

This helper creates a compact package-native analysis record. It summarizes analysis settings, source columns, anchoring information, and which downstream outputs are currently available.

Value

A named list with class `mfrm_manifest`.

When to use this

Use `build_mfrm_manifest()` when you want a compact, machine-readable record of how an analysis was run. Compared with related helpers:

- `export_mfrm_tables()` writes analysis tables only.
- `build_mfrm_manifest()` records settings and available outputs.
- `build_mfrm_replay_script()` creates an executable R script.
- `export_mfrm_bundle()` writes a project-folder reproducibility bundle; review every included artifact for person-level or sensitive data before sharing it.

Output

The returned bundle has class `mfrm_manifest` and includes:

- `summary`: one-row analysis overview
- `readiness`, `readiness_components`, and `readiness_parameters`: the fitted model's recorded numerical checks and restrictions on inference, overall and by component or parameter
- `environment`: package/R/platform metadata
- `model_settings`: key-value model settings table
- `source_columns`: key-value data-column table
- `estimation_control`: key-value optimizer settings table
- `anchor_summary`: facet-level anchor summary. `model_settings` separately records `centering_scope` and `noncenter_facet`; missing convention metadata in a native fit retains the legacy `"remainder"` interpretation
- `anchors`: machine-readable anchor table

- `hierarchical_review`: retained traceability table for hierarchical / small-sample design flags
- `missing_recoding`: retained traceability table for missing-code recoding
- `shrinkage_review`: retained traceability table for shrinkage settings
- `available_outputs`: availability table for diagnostics/bias/PCA/prediction outputs
- `dependencies`, `input_summary`, and `session_info`: reproducibility metadata tables. `input_summary` records object structure and does not claim byte-for-byte identity across platforms or serialization formats
- `settings`: manifest build settings

Interpreting output

The `summary` table is the direct place to confirm that you are looking at the intended analysis. The `model_settings`, `source_columns`, and `estimation_control` tables are designed for reproducibility records and method write-up. `Converged` records the optimizer return-code decision; `InferenceReady` records whether the fit passes the additional numerical checks required for inference. They need not agree; passing these checks does not establish model fit or the statistical adequacy of every interval method. The `readiness` tables preserve these checks for the original fitted model, including the check-format version and reason codes. Re-running the analysis must repeat the checks using the new fit; the saved records do not certify it. Active latent-regression fits also record their population-model provenance there, including the fitted scoring basis, stored `population_formula`, and person-level data requirements used by the fitted population model. When categorical background variables are expanded through `stats::model.matrix()`, `population_xlevel_variables` and `population_contrast_variables` identify the variables whose fitted coding must be preserved for replay/scoring. The `available_outputs` table is especially useful before building bundles, because it tells you whether residual PCA, anchors, bias results, or prediction-side artifacts are already available. A practical reading order is `summary` first, `available_outputs` second, and `anchors` last when reproducibility depends on fixed constraints.

Typical workflow

1. Fit a model with `fit_mfrm()` or `run_mfrm_facets()`.
2. Compute diagnostics once with `diagnose_mfrm()` if you want explicit control over residual PCA.
3. Build a manifest and inspect `summary` plus `available_outputs`.
4. If you need files on disk, pass the same objects to `export_mfrm_bundle()`.

For `GPCM` fits, the manifest is available with an explicit `gpcm_boundary` table. It records supported direct diagnostics/reporting surfaces while keeping full FACETS score-side contract review blocked and routing design forecasting through its separate caveated capability row.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model. Supplied unit scores, plausible values and fit-derived forecasts must also match this calibration. A forecast created with `sim_spec` remains a separate scenario; its source is recorded without asserting a match to the fit.

See Also

export_mfrm_bundle(), build_mfrm_replay_script(), make_anchor_table(), reporting_checklist()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_mfrm_manifest", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
manifest <- build_mfrm_manifest(fit, diagnostics = diag)
manifest$summary[, c("Model", "Method", "Observations", "Facets")]
manifest$available_outputs[, c("Component", "Available")]

## End(Not run)
```

---

build_mfrm_network_review

Build an MFRM network review

Description

Build an MFRM network review

Usage

```
build_mfrm_network_review(
  fit = NULL,
  diagnostics = NULL,
  sparse_design = NULL,
  peer_review_design = NULL,
  top_n_subsets = NULL,
  min_observations = 0,
  top_n = 10,
  include_graph = FALSE,
  data = NULL,
  person = NULL,
  facets = NULL,
  score = NULL,
  basis = c("observed", "planned"),
  weight = NULL
)

## S3 method for class 'mfrm_network_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
sparse_design	Optional sparse-design metadata. Supply either the generated data frame that carries the mfrm_sparse_design attribute, the attribute itself, or a data frame with sparse design columns such as SparseDesignActive, DesignDensity, MinCommonPersonsPerRaterPair, and ZeroCommonRaterPairs.
peer_review_design	Optional peer-review design metadata. Supply either the generated data frame that carries the mfrm_peer_review_design attribute, the attribute itself, or its overview data frame.
top_n_subsets	Optional maximum number of connected-subset rows to retain before constructing the graph; passed to mfrm_network_analysis().
min_observations	Minimum observations required to keep a subset row; passed to mfrm_network_analysis().
top_n	For build_mfrm_network_review(): Number of central/cut/bridge rows to retain in the review. For summary(): Number of central/cut/bridge rows to keep in the compact summary.
include_graph	Logical; if TRUE, keep the underlying igraph object in the nested source_network bundle.
data	Optional raw assignment/rating data, supplied instead of fit.
person, facets	Names of the person column and one or more facet columns in data. Include only Rater in facets to inspect person-rater links without a common Criterion node obscuring separate panels.
score	Numeric score column for basis = "observed". Recode study-specific missing markers to NA first; no category recoding or estimation is done.
basis	"observed" includes rows with complete identifiers and nonmissing finite scores; "planned" includes complete assignment rows without scores. Planned networks require score = NULL and weight = NULL.
weight	Optional numeric weight column for observed raw data. Missing and zero-weight rows do not create edges. Edge weights still count rows, not sampling weights. Negative or infinite weights are refused.
object	Output from build_mfrm_network_review().
digits	Number of digits for printed numeric values.
...	Reserved for generic compatibility.

Details

build_mfrm_network_review() is a synthesis layer over mfrm_network_analysis(). It keeps the measurement model and graph view in separate sections: MFRM estimates remain the measurement results, while the network review summarizes co-observation connectedness and linking vulnerability in the observed design. This is especially useful for sparse or incomplete rater-mediated

designs, where common-person links, connected subsets, articulation points, and bridge edges can explain why an otherwise estimable model depends on fragile design links.

The review status is deliberately conservative and descriptive. It is not a literature-derived adequacy cut point for fit, separation, recovery, or rater quality. Use it to decide which design links, anchors, or additional observations need inspection before making common-scale claims.

`summary()` method: Summarize an MFRM network review

Value

A bundle of class `mfrm_network_review` containing:

- `overview`: connectedness and front-door review status
- `network_summary`: graph-level metrics from `mfrm_network_analysis()`
- `facet_summary`: facet-level vulnerability summaries
- `top_central_nodes`, `top_cut_nodes`, `top_bridge_edges`: follow-up rows
- `sparse_review`: optional sparse-design linking review
- `peer_review`: optional peer-review assignment and linkage diagnostics
- `reporting_map`: boundary between MFRM, design network, sparse design, peer-review design, and rater-effect network routes

For `summary()`: An object of class `summary.mfrm_network_review`.

References

- Wind, S. A., & Jones, E. (2018). The stabilizing influences of linking set size and model-data fit in sparse rater-mediated assessment networks. *Educational and Psychological Measurement*. doi:10.1177/0013164417703733.
- Wind, S. A., Jones, E., & Grajeda, S. (2023). Does sparseness matter? Examining the use of generalizability theory and many-facet Rasch measurement in sparse rating designs. *Applied Psychological Measurement*, 47(5-6), 351-364. doi:10.1177/01466216231182148.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

Before fitting

Supply `data`, `person`, `facets` and `score` instead of `fit`. Known levels from incomplete/unscored rows remain as nodes, including isolates. A planned assignment roster can be examined with `basis = "planned"`; absent assignments cannot be reconstructed from an observed-only file. `row_usage` records each input row's inclusion and first exclusion reason. With a fit, only its retained rows are available here; review original missing assignments before fitting. No prefit result supplies an identification audit or model-readiness decision. `settings$is_disjoint` describes the source graph, even when subset filters omit components; compare source/retained subset counts. Filters do not establish linking. See `vignette("mfrmr-networks-and-reporting")` for a panel example.

See Also

mfrm_network_analysis(), subset_connectivity_report(), build_summary_table_bundle(), rater_network_analysis(), rater_halo_network_analysis()

build_mfrm_network_review()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_mfrm_network_review", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
if (requireNamespace("igraph", quietly = TRUE)) {
  review <- build_mfrm_network_review(fit)
  summary(review)
  build_summary_table_bundle(review)
}

## End(Not run)
```

---

build_mfrm_replay_script

Build a package-native replay script for an MFRM analysis

Description

Build a package-native replay script for an MFRM analysis

Usage

```
build_mfrm_replay_script(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  data_file = "your_data.csv",
  fit_person_data_file = NULL,
  script_mode = c("auto", "fit", "facets"),
  include_bundle = FALSE,
  bundle_dir = "analysis_bundle",
  bundle_prefix = "mfrmr_replay"
)
```

Arguments

fit	Output from fit_mfrm() or run_mfrm_facets().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are reused from run_mfrm_facets() when available, otherwise recomputed.
bias_results	Optional output from estimate_bias() or a named list of bias bundles. When supplied, the generated script includes package-native bias estimation calls.
population_prediction	Optional output from predict_mfrm_population() to recreate in the generated script.
unit_prediction	Optional output from predict_mfrm_units() to recreate in the generated script.
plausible_values	Optional output from sample_mfrm_plausible_values() to recreate in the generated script.
data_file	Path to the analysis data file used in the generated script.
fit_person_data_file	Optional CSV filename to read for the fit-level latent-regression replay person table. When NULL, the replay script embeds that table inline. export_mfrm_bundle() uses this to keep replay scripts portable while avoiding large inline literals.
script_mode	One of "auto", "fit", or "facets". "auto" uses run_mfrm_facets() when the input object came from that workflow.
include_bundle	If TRUE, append an export_mfrm_bundle() call to the generated script.
bundle_dir	Output directory used when include_bundle = TRUE.
bundle_prefix	Prefix used by the generated bundle exporter call.

Details

This helper builds a reproducible script from mfrmr's installed API. The generated script assumes the user has the package installed and provides a data file at data_file.

Anchor and group-anchor constraints are embedded directly from the fitted object's stored configuration, so the script can replay anchored analyses without manual table reconstruction. The script retains centering_scope; a native fit saved before this setting was recorded uses "remainder". Running it performs a new fit under that convention, not a conversion of the saved object. To adopt whole-facet centering, review the generated call and change it to "facet", preserving the intended population, uncentered facet, rubric and step/slope owners. Recheck the resulting constraints and readiness. Use saveRDS()/readRDS() to reopen an unchanged fit, or the replay from export_mfrm_results() to reopen a saved results bundle without refitting.

Adjustments made with apply_empirical_bayes_shrinkage() are replayed after fitting, using the most recently recorded prior and Person settings. The original fitting settings remain separate so that attaching diagnostic SEs and then applying shrinkage is reproduced in the same order. For a workflow object with a post-fit adjustment, "auto" selects "fit" mode; an explicit "facets" request is refused because it cannot retain that operation order. Older shrinkage results without complete settings must first be refreshed with apply_empirical_bayes_shrinkage(). Manually edited SEs or other table edits require their own reproducible editing steps.

When the supplied fit uses the latent-regression MML branch, the generated fit-mode script also carries the stored replay-ready person table together with the corresponding population_formula / person_id / population_policy arguments needed to recreate the population model. By default that replay-ready table is embedded inline; when fit_person_data_file is supplied, the generated script reads it from that sidecar CSV relative to the replay script location.

For GPCM, replay scripts are available with an explicit gpcm_boundary table. The generated script records step_facet and slope_facet settings, but full FACETS score-side contract review remains outside this replay contract. Role-based design forecasting is available through predict_mfrm_population() as a separate caveated helper route.

Value

A named list with class mfrm_replay_script.

When to use this

Use build_mfrm_replay_script() when you want a package-native recipe that another analyst can rerun later. Compared with related helpers:

- build_mfrm_manifest() records settings but does not run anything.
- build_mfrm_replay_script() produces executable R code.
- export_mfrm_bundle() can optionally write the replay script to disk.

Interpreting output

The returned object contains:

- summary: a one-row overview of the chosen replay mode and whether bundle export was included
- script: the generated R code as a single string
- source_readiness, source_readiness_components, and source_readiness_parameters: readiness provenance from the source fit
- anchors and group_anchors: the exact stored constraints that were embedded into the script

The source readiness row is not copied into the replayed fit. The generated script recomputes readiness and warns when its fit-level decision fields do not match the source record. Identifier columns are read as text, preserving leading zeros and literal "NA" labels. Original column names and factor coding in exported person-level background data are retained. Stored diagnostic settings are replayed as well; older diagnostic objects may not retain interaction selection or PCA limits, so rerun diagnose_mfrm() before exporting them when those settings matter.

If ScriptMode is "facets", the script replays the higher-level run_mfrm_facets() workflow. If it is "fit", the script uses fit_mfrm() directly.

Mode guide

- "auto" is the safest default and follows the structure of the supplied object.
- "fit" is useful when you want a minimal script centered on fit_mfrm().
- "facets" is useful when you want to preserve the higher-level run_mfrm_facets() workflow, including stored column mapping.

Typical workflow

1. Finalize a fit and diagnostics object.
2. Generate the replay script with the path you want users to read from.
3. Write `replay$script` to disk, or let `export_mfrm_bundle()` do it for you.
4. Rerun the script in a fresh R session to confirm reproducibility.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model. Supplied unit scores, plausible values and fit-derived forecasts must also match this calibration. A forecast created with `sim_spec` remains a separate scenario; its source is recorded without asserting a match to the fit.

See Also

`build_mfrm_manifest()`, `export_mfrm_bundle()`, `run_mfrm_facets()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_mfrm_replay_script", run.dontrun = TRUE)
## Not run:
library(mfrmr)
# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit
replay <- build_mfrm_replay_script(fit, data_file = "your_data.csv")
replay$summary[, c("ScriptMode", "ResidualPCA", "BiasPairs")]
cat(substr(replay$script, 1, 120))

## End(Not run)
```

---

build_mfrm_resampling_spec

Build an observed-data resampling specification

Description

Build an observed-data resampling specification

Usage

```
build_mfrm_resampling_spec(
  data,
  person,
  facets,
  score,
  strata = NULL,
  preserve_facets = NULL,
  design = c("stratified_subsample", "stratified_bootstrap"),
  reps = 20,
  sample_fraction = 0.5,
  sample_n = NULL,
  replace = NULL,
  seed = NULL,
  min_per_stratum = 1,
  topup_preserve_facets = TRUE
)
```

Arguments

data	A long-format observed MFRM data set with distinct column names. The `.mfrm_*` metadata columns listed below are reserved and must not already be present. Numeric identifiers must retain distinct text labels.
person	Person/respondent identifier column.
facets	Non-person facet columns used by the target MFRM fit.
score	Ordered score column.
strata	Optional person-level stratification columns, for example a `Region` or L1 group column. Each person must have at most one unique stratum combination.
preserve_facets	Optional facet columns whose level coverage should be reviewed and, when possible, topped up after the stratified person draw. A common choice is the rater facet.
design	Resampling design. `"stratified_subsample"` samples persons without replacement inside each stratum. `"stratified_bootstrap"` samples persons with replacement inside each stratum and re-keys duplicate person instances in the returned data.
reps	Number of resampling replicates to draw.
sample_fraction	Fraction of persons to draw within each stratum when `sample_n = NULL`. Targets are rounded upward and respect `min_per_stratum`. The default is `0.5` for both designs; use `1` for a same-size bootstrap.
sample_n	Optional target number of persons to draw per stratum. Supply either one scalar used for every stratum, or a named numeric vector whose names match `spec$target_plan$Stratum`. An unnamed vector of length equal to the number of strata follows that table's order. Targets are rounded upward and capped at available Persons when sampling without replacement.

`replace` Optional logical override for replacement. By default, replacement is `FALSE` for `"stratified_subsample"` and `TRUE` for `"stratified_bootstrap"`.

`seed` Optional seed used by `draw_mfrm_resamples()`.

`min_per_stratum`
Minimum target persons per represented stratum.

`topup_preserve_facets`
Logical; if `TRUE`, add extra person clusters when possible to recover missing levels of `preserve_facets`. This changes the sampling distribution and can increase stratum sizes; it is not an ordinary bootstrap. Set `FALSE` for ordinary cluster-bootstrap draws.

Details

This helper defines a resampling design for observed-data stability checks. It is intentionally separate from `build_mfrm_sim_spec()` and `simulate_mfrm_recovery()`. The full-data estimates used with these draws are reference estimates, not known truth, so downstream summaries should be described as estimation stability, reproducibility, or agreement with a full-data reference rather than strict parameter recovery.

The design is person-clustered: all rows for a selected person are kept together. For bootstrap draws, duplicated person clusters are re-keyed in the returned data while the original identifier is retained in `.mfrm_original_person`.

Value

An object of class `mfrm_resampling_spec` containing the input data, columns, target plan, person/stratum rosters, `sampling_scheme` and interpretation in `terminology`.

Sampling assumptions and interpretation

For ordinary stratified Person-cluster bootstrap draws, set `design = "stratified_bootstrap"`, `sample_fraction = 1`, `sample_n = NULL`, and `topup_preserve_facets = FALSE`. The unit is a Person, not a rating row; unequal-sized clusters retain all their rows on every draw. This targets variation across independent Persons within strata, conditional on the observed non-Person facet levels and assignment patterns. It does not reproduce sampling new raters or tasks, or dependence across Persons induced by shared random effects or peer-review networks. Crossed random effects require a resampling scheme matched to that sampling target.

A reduced sample size or coverage top-up instead defines a stability experiment. Top-up chooses an additional Person carrying a currently absent preserved level, and can also bring other missing levels. It changes inclusion probabilities and stratum totals. Use `sampling_scheme`, the returned terminology and base/added/final counts to report what was drawn. These helpers do not fit models or construct confidence intervals; the presence of the word bootstrap does not qualify downstream interval coverage. Reference estimates from the full sample still have estimation uncertainty.

Labels and saved specifications

Original Person keys and data values are retained. Stratum labels escape percent signs and vertical bars, and distinguish missing values, empty strings and literal `"(missing)"/"(empty)"` labels. Inspect `target_plan` before supplying named `sample_n` values. Data column names `.mfrm_original_person`,

`.mfrm_draw_unit`, `.mfrm_stratum`, `.mfrm_topup`, and `.mfrm_topup_reason` are reserved for returned metadata.

`draw_mfrm_resamples()` rebuilds derived tables from the specification's declared inputs, including saved older specifications. Corrected labels and top-up rules can change old draws even with the same seed; retain old results separately. `saveRDS(spec, ...)` preserves the input data and seed; `saveRDS(draws, ...)` preserves the actual samples, per-replicate seeds and manifests. Use an explicit seed to reproduce draws in a matching R/RNG environment. Manifest-only results retain the specification and input data but omit the materialized samples.

References

Owen, A. B. and Eckles, D. (2012). Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, 6(3), 895–927. doi:10.1214/12AOAS547. This discusses crossed-factor resampling; it is not the method implemented by this Person-cluster helper.

See Also

`draw_mfrm_resamples()`, `build_mfrm_sim_spec()`

Examples

```
# Bundled synthetic ratings: inspect stability in half-samples by Group.
ratings <- load_mfrmr_data("example_operational")
spec <- build_mfrm_resampling_spec(
  ratings, person = "Person", facets = c("Rater", "Criterion"),
  score = "Score", strata = "Group", preserve_facets = "Rater",
  reps = 2, sample_fraction = 0.5, seed = 99
)
draws <- draw_mfrm_resamples(spec)
summary(draws)$overview
draws$stratum_manifest  # Base, top-up and final Person-cluster counts.
```

---

build_mfrm_sim_spec *Build an explicit simulation specification for MFRM design studies*

Description

Shows the model, design size, assignment, and latent-distribution settings needed to understand the specification without printing its complete machine-readable planning metadata.

Usage

```
build_mfrm_sim_spec(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
```

```
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  thresholds = NULL,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  facet_names = NULL,
 assignment = c("crossed", "rotating", "sparse_linked", "resampled", "skeleton"),
  latent_distribution = c("normal", "empirical"),
  empirical_person = NULL,
  empirical_rater = NULL,
  empirical_criterion = NULL,
  assignment_profiles = NULL,
  design_skeleton = NULL,
  sparse_controls = NULL,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL,
  population_formula = NULL,
  population_coefficients = NULL,
  population_sigma2 = NULL,
  population_covariates = NULL,
  population_assignment = c("matched", "independent")
)

## S3 method for class 'mfrm_sim_spec'
print(x, ...)
```

Arguments

n_person	Number of persons/respondents to generate.
n_rater	Number of rater facet levels to generate.
n_criterion	Number of criterion/item facet levels to generate.
raters_per_person	Number of raters assigned to each person.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases implied by facet_names (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). The nested named-facet form design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both design and the scalar count arguments.

score_levels — Number of ordered score categories.

theta_sd — Standard deviation of simulated person measures.

rater_sd — Standard deviation of simulated rater severities.

criterion_sd — Standard deviation of simulated criterion difficulties.

noise_sd — Optional observation-level noise added to the linear predictor.

step_span — Spread used to generate equally spaced thresholds when thresholds = NULL.

thresholds — Optional threshold specification. Use a numeric vector of common thresholds; a named list such as list(C01 = c(-1, 0, 1)); a numeric matrix with one row per StepFacet and one column per step; or a long data frame with columns StepFacet, Step/StepIndex, and Estimate.

model — Measurement model recorded in the simulation specification.

step_facet — Step facet used when model = "PCM" and threshold values vary across levels.

slope_facet — Slope facet used when model = "GPCM". The current GPCM branch requires slope_facet == step_facet.

slopes — Optional slope specification for model = "GPCM". Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns SlopeFacet and Estimate. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean- one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact PCM reduction. The GPCM model form follows Muraki's generalized partial credit model; the package's slope-regime labels are validation stress labels, not published psychometric cut points.

facet_names — Optional public names for the two simulated non-person facet columns. Supply either an unnamed character vector of length 2 in rater-like / criterion-like order, or a named vector with names c("rater", "criterion").

assignment — Assignment design. "crossed" means every person sees every rater; "rotating" uses a balanced rotating subset; "sparse_linked" uses an incomplete rating design with optional linking persons; "resampled" reuses empirical person-level rater-assignment profiles; "skeleton" reuses an observed person-by-facet design skeleton.

latent_distribution — Latent-value generator. "normal" samples from centered normal distributions using the supplied standard deviations. "empirical" resamples centered support values from empirical_person/empirical_rater/empirical_criterion.

empirical_person — Optional numeric support values used when latent_distribution = "empirical".

empirical_rater — Optional numeric support values used when latent_distribution = "empirical".

empirical_criterion — Optional numeric support values used when latent_distribution = "empirical".

assignment_profiles — Optional data frame with columns TemplatePerson and the public rater-like facet column (optionally Group) describing empirical person-level rater-assignment profiles used when assignment = "resampled". The canonical name Rater is also accepted.

design_skeleton
: Optional data frame with columns TemplatePerson, the public rater-like facet column, and the public criterion-like facet column (optionally Group, Weight, and TemplatePersonReuse) describing an observed response skeleton used when assignment = "skeleton". The canonical names Rater and Criterion are also accepted. TemplatePersonReuse = TRUE asks simulate_mfrm_data() to keep the template-person order instead of resampling templates, which is used by build_peer_review_sim_spec(). This flag must be uniform across the skeleton. Every row is retained, including repeated Person-by-facet cells. Each generates a separate conditionally independent response; Weight is carried into fitting and does not multiply the number of generated responses.

sparse_controls
: Optional named list used when assignment = "sparse_linked". Supported entries are link_fraction (default 0.1), link_persons (overrides link_fraction), link_raters_per_person (default n_rater), assignment_mode ("balanced" or "random"), and min_common_persons_per_rater_pair (diagnostic target, default 1).

group_levels
: Optional character vector of group labels.

dif_effects
: Optional data frame of true group-linked DIF effects.

interaction_effects
: Optional data frame of true interaction effects.

population_formula
: Optional one-sided formula describing a person-level latent-regression population model used when generating person measures, for example ~ X + G. When supplied, person measures are generated from X %*% beta + e rather than from N(0, theta_sd^2).

population_coefficients
: Optional numeric vector of latent-regression coefficients corresponding to the design matrix implied by population_formula.

population_sigma2
: Optional residual variance for the latent-regression person distribution.

population_covariates
: Optional template data frame containing one row per template person and the background variables referenced by population_formula. Numeric/logical and categorical factor/character variables are expanded through the same stats::model.matrix() contract used by latent-regression fitting. During simulation, template rows are resampled with replacement to the requested n_person. Transformations such as scale(), poly() and splines retain the basis learned from the full template; they are not recomputed for each simulated sample. Stored factor levels and contrasts also carry through the study runners' refits. A resampled cohort can still have a rank-deficient design and fail fitting; that trial remains a failure rather than being redrawn until it succeeds.

population_assignment
: How an active population model is associated with "resampled" or "skeleton" assignments. The default "matched" draws one template Person jointly for covariates and assignments. Both tables must cover the same TemplatePerson

IDs, with matching Group labels if present in both. "independent" samples covariate rows separately and intentionally breaks these associations. In that case both tables cannot define Group; rename a covariate if it represents a different grouping. For other assignment designs this setting has no effect. Population-template Group, when present, supplies the response group; group_levels must include all template labels and does not rebalance them.

x An mfrm_sim_spec object.

... Reserved for generic compatibility.

Details

build_mfrm_sim_spec() creates an explicit, portable simulation specification that can be passed to simulate_mfrm_data(). The goal is to make the data-generating mechanism inspectable and reusable rather than relying only on ad hoc scalar arguments.

The resulting object records:

- design counts (n_person, n_rater, n_criterion, raters_per_person)
- latent spread assumptions (theta_sd, rater_sd, criterion_sd)
- optional empirical latent support values for semi-parametric simulation
- threshold structure (threshold_table)
- optional discrimination structure for GPCM (slope_table) and its identified log-slope spread label (slope_regime)
- assignment design (assignment)
- optional sparse linked-design controls (sparse_controls) when assignment = "sparse_linked"
- optional empirical assignment profiles (assignment_profiles) with optional person-level Group labels
- optional observed response skeleton (design_skeleton) with optional person-level Group labels and observation-level Weight values
- optional person-level latent-regression population metadata including population_formula, population_coefficients, population_sigma2, and a reusable template of person-level covariates, including model-matrix xlevel/contrast provenance for categorical covariates
- planning_scope, an explicit record that the current planning/forecasting helpers target the role-based person x rater-like x criterion-like design contract rather than a fully arbitrary-facet planner
- planning_constraints, an explicit record of which design variables can currently be changed from that specification without rebuilding it
- planning_schema, a combined structural contract containing the role descriptor, supported scope, mutability map, facet manifest, and the normalized named-facet design representation. Its structural_design_* fields describe deterministic balance and connectivity reviews; they do not imply support for arbitrary-facet simulation.
- the design$facets parser normalizes named facet counts through the same design representation used by the planning helpers
- optional signal tables for DIF and interaction bias

The current generator targets the package's standard person x rater x criterion workflow, but the public output names for those two facet roles can now be customized with `facet_names`. This naming layer improves public ergonomics; it does not provide a fully arbitrary-facet simulator. Internally, helper objects keep canonical role mappings so that planning functions can treat the first non-person facet as rater-like and the second as criterion-like. When threshold values are provided by `StepFacet`, the supported step facets are the generated levels of the chosen public rater-like or criterion-like column. For convenience, step-facet-specific thresholds can be supplied as a named list or as a numeric matrix whose row names are `StepFacet` labels. When `model = "GPCM"`, the same public facet naming rules apply to the slope table; the current bounded branch keeps `slope_facet` equal to `step_facet`. The `slope_regime` field summarizes the centered log-slope spread so recovery simulations can be read against the intended generator stress level. Its labels (`unit_slopes`, `near_flat`, `moderate`, and `high_dispersion`) are package validation labels; they are not model-fit decisions and should not be interpreted as literature-derived adequacy thresholds. The GPCM data-generating form follows Muraki (1992, doi:10.1177/014662169201600206), while information-function interpretation follows Muraki (1993, doi:10.1177/014662169301700403). The explicit simulation-specification metadata is intended to support ADEMP-style simulation reporting as described by Morris, White, and Crowther (2019, doi:10.1002/sim.8086).

The `assignment = "sparse_linked"` branch follows sparse rater-mediated assessment work in treating incomplete rater assignment as planned missingness rather than incidental nonresponse. Its `link_persons` and `link_raters_per_person` controls emulate a common linking set so users can inspect rater-pair common-person counts before using the generated data in recovery or design studies. This is a design generator and diagnostic metadata layer; it does not impose a universal minimum-linking cutoff. Sparse-design motivation follows Wind, Jones, and Grajeda (2023, doi:10.1177/01466216231182148), Wind and Jones (2018, doi:10.1177/0013164417703733), and DeMars, Shapovalov, and Hathcoat (2023).

If `population_formula` is supplied, the simulation specification carries a person-level latent-regression generator. This affects only the person distribution. The current implementation keeps the non-person facets in the existing many-facet Rasch generator and resamples rows from `population_covariates` to the requested design size before computing $\theta_n = x_n^\top \beta + \varepsilon_n$ with $\varepsilon_n \sim N(0, \sigma^2)$.

Value

An object of class `mfrm_sim_spec`.

`x`, invisibly.

Interpreting output

This object does not contain simulated data. It is a data-generating specification that tells `simulate_mfrm_data()` how to generate them.

See Also

`extract_mfrm_sim_spec()`, `simulate_mfrm_data()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_mfrm_sim_spec", run.dontrun = TRUE)
## Not run:
```

```
spec <- build_mfrm_sim_spec(
  design = list(person = 8, rater = 2, criterion = 2, assignment = 1),
  assignment = "rotating"
)
spec$model
spec$assignment
nrow(spec$threshold_table)

## End(Not run)
```

`build_misfit_casebook` *Build a case-level misfit review bundle*

Description

Build a case-level misfit review bundle

Usage

```
build_misfit_casebook(
  fit,
  diagnostics = NULL,
  unexpected = NULL,
  displacement = NULL,
  administration_id = NULL,
  wave_id = NULL,
  top_n = 25
)

## S3 method for class 'mfrm_misfit_casebook'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
`unexpected`	Optional output from `unexpected_response_table()`.
`displacement`	Optional output from `displacement_table()`.
`administration_id`	Optional scalar identifier describing the current administration or form. It is stored in row-level provenance and summary outputs when supplied.
`wave_id`	Optional scalar identifier for the current wave or occasion. It is stored in row-level provenance and summary outputs when supplied.
`top_n`	For `build_misfit_casebook()`: Maximum number of rows to keep in compact summary outputs. For `summary()`: Number of top case rows to keep in the compact summary.

object	Output from build_misfit_casebook().
digits	Number of digits for printed numeric values.
...	Reserved for generic compatibility.

Details

build_misfit_casebook() is a synthesis layer over package-native screening outputs. It does not invent a new misfit statistic. Instead, it organizes existing evidence families into one case-level review surface:

- element-level Infit / Outfit MnSq misfit from diagnostics$fit (rows whose Infit or Outfit MnSq falls outside the configured Linacre heuristic review band, 0.5-1.5 by default)
- strict marginal cell screens from diagnostics$marginal_fit$top_cells
- strict pairwise screens from diagnostics$marginal_fit$pairwise$top_pairs
- unexpected responses from unexpected_response_table()
- displacement flags from displacement_table()

The result is an operational review bundle. It is not a formal adjudication system, and repeated signals across evidence families should be prioritized over any single isolated case row. In addition to raw case rows, the object includes stable grouping views such as by_person, by_facet_level, by_source_family, and by_wave to support operational triage. The source_support component records which evidence families are currently supported, caveated, or deferred under the active model.

summary() method: Summarize a misfit-casebook object

Value

An object of class mfrm_misfit_casebook.

For summary(): An object of class summary.mfrm_misfit_casebook.

Recommended input route

1. Fit with fit_mfrm().
2. Build diagnostics with diagnose_mfrm().
3. Optionally build unexpected_response_table() and displacement_table() yourself when you want custom thresholds before synthesizing the casebook.

GPCM boundary

For GPCM, the helper is available with caveat. The casebook inherits exploratory screening semantics from the underlying residual and strict marginal sources; it should not be read as a formal inferential case test.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model.

See Also

diagnose_mfrm(), unexpected_response_table(), displacement_table(), plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise")

build_misfit_casebook()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("build_misfit_casebook", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
# A balanced slice retains every Rater and Criterion while running quickly.
toy <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM", quad_points = 5)
diag <- diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")
casebook <- build_misfit_casebook(fit, diagnostics = diag, top_n = 10)
summary(casebook)
casebook$top_cases

## End(Not run)
```

---

build_model_choice_review

Build a model-choice review across RSM, PCM, and GPCM fits

Description

Build a model-choice review across RSM, PCM, and GPCM fits

Usage

```
build_model_choice_review(
  ...,
  labels = NULL,
  run_weighting_review = NULL,
  theta_range = c(-6, 6),
  theta_points = 61L,
  top_n = 10L,
  warn_constraints = TRUE
)

## S3 method for class 'mfrm_model_choice_review'
summary(object, digits = 3, ...)
```

Arguments

...	For build_model_choice_review(): Two or more fitted mfrm_fit objects from fit_mfrm(). For summary(): Reserved for generic compatibility.
labels	Optional labels for the supplied fits. If omitted, names from ... are used when available; otherwise labels are generated from model/method combinations.
run_weighting_review	Logical. If TRUE and the supplied fits include at least one RSM/PCM reference plus one GPCM fit, also run build_weighting_review() for the first such pair.
theta_range, theta_points, top_n	Passed to build_weighting_review() when run_weighting_review = TRUE.
warn_constraints	Passed to compare_mfrm().
object	Output from build_model_choice_review().
digits	Number of digits for printed numeric values.

Details

build_model_choice_review() is a user-facing synthesis helper. It does not estimate new models. It bundles:

- compare_mfrm() for verified AIC/Person-BIC/SABIC/log-likelihood comparison;
- comparison-boundary warnings captured from compare_mfrm() and retained in comparison_warnings for printing and appendix export;
- model-role guidance for RSM, PCM, and GPCM;
- reported step/slope coordinate counts, identified free-parameter counts, and the stored fit-readiness decision for every supplied model;
- downstream-route availability for APA output, score-side export, linking, recovery, fair averages, bias screening, and summary-appendix handoff;
- report wording templates that avoid treating better GPCM fit as an automatic operational-scoring decision;
- gpcm_capability_matrix() when GPCM is present;
- optionally, build_weighting_review() for the first Rasch-family reference versus GPCM pair.

The word "bounded" describes the documented model and workflow scope: the package does not implement every possible generalized partial-credit many-facet extension. It does **not** mean that finite optimizer box bounds define the estimator. The current route uses positive slopes, allows separate slope/step owners in MML (JML requires slope_facet == step_facet), identifies slopes on the log scale with geometric mean 1, and keeps several downstream score-side/reporting helpers outside the documented boundary. Model-choice ranking also requires the current selectable IC contract: q<31 fits retain raw criteria but produce a screening/review-only bundle.

summary() method: Summarize a model-choice review

Value

An object of class `mfrm_model_choice_review`.

For `summary()`: An object of class `summary.mfrm_model_choice_review`.

See Also

`compare_mfrm()`, `build_weighting_review()`, `gpcm_capability_matrix()`, `compute_information()`

`build_model_choice_review()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit_rsm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "RSM", quad_points = 31)
fit_pcm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "PCM", step_facet = "Criterion",
                    quad_points = 31)
review <- build_model_choice_review(RSM = fit_rsm, PCM = fit_pcm)
summary(review)
```

---

build_peer_review_sim_spec

Build a peer-review simulation specification

Description

Build a peer-review simulation specification

Usage

```
build_peer_review_sim_spec(
  n_submission = 50,
  n_criterion = 4,
  reviewers_per_submission = 3,
  anchor_fraction = 0.1,
  anchor_submissions = NULL,
  anchor_reviewers_per_submission = NULL,
  avoid_self_review = TRUE,
  assignment_mode = c("balanced", "random"),
  seed = NULL,
  score_levels = 4,
  theta_sd = 1,
  reviewer_sd = 0.45,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
```

```
  step_facet = "Criterion",
  thresholds = NULL,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL
)
```

Arguments

n_submission	Number of submissions/authors to generate.
n_criterion	Number of rubric criteria.
reviewers_per_submission	Number of peer reviewers assigned to each ordinary submission.
anchor_fraction	Fraction of submissions treated as common-link anchor submissions when anchor_submissions is not supplied.
anchor_submissions	Optional number of common-link anchor submissions. Anchor submissions receive anchor_reviewers_per_submission reviewers.
anchor_reviewers_per_submission	Number of reviewers assigned to each anchor submission. Defaults to all eligible peers when anchors are used and self-review is disallowed; recorded as 0 when no anchor submissions are requested.
avoid_self_review	Logical; if TRUE, a reviewer is never assigned to review their own submission.
assignment_mode	Assignment algorithm. "balanced" assigns reviewers with the lowest current load using deterministic rotating tie-breaks. "random" samples eligible reviewers without replacement.
seed	Optional seed used only for random peer-review assignment when assignment_mode = "random".
score_levels, theta_sd, reviewer_sd, criterion_sd, noise_sd, step_span	Generator settings passed to build_mfrm_sim_spec(). reviewer_sd maps to the standard MFRM rater-severity spread.
model, step_facet, thresholds	Measurement-model settings passed to build_mfrm_sim_spec(). The first public facet is Reviewer; the second is Criterion.
group_levels, dif_effects, interaction_effects	Optional signal settings passed to build_mfrm_sim_spec().

Details

build_peer_review_sim_spec() creates a fixed person-by-reviewer-by-rubric skeleton for peer-assessment or peer-review studies. Submissions and peer reviewers share the same ID universe (P001, P002, ...), so self-review can be structurally excluded and checked in the generated data. The specification uses the existing assignment = "skeleton" generator and records peer-review metadata; it does not introduce a new measurement model. MFRM still estimates person/submission

measures, reviewer severity, and criterion difficulty, while design-network review can inspect whether the peer-review graph is sufficiently linked.

The common-link anchor controls follow the same logic used in sparse rater-mediated designs: when most submissions receive only a few peer reviews, assigning all or many reviewers to a small anchor set can strengthen links among reviewers. The helper labels these rows as design diagnostics, not universal adequacy thresholds for fit, separation, or recovery.

Value

An object of class `mfrm_sim_spec` with `peer_review` metadata and a fixed peer-review design skeleton.

References

- Farrokhi, F., Esfandiari, R., & Schaefer, E. (2012). A many-facet Rasch measurement of differential rater severity/leniency in three types of assessment. *JALT Journal*, 34(1), 79-102. doi:10.37546/JALTJJ34.1-3.
- Uto, M., & Ueno, M. (2020). A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo. *Behaviormetrika*, 47, 469-496. doi:10.1007/s41237-020-00115-7.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

See Also

`simulate_mfrm_data()`, `build_mfrm_network_review()`, `build_mfrm_sim_spec()`

Examples

```
peer_spec <- build_peer_review_sim_spec(
  n_submission = 12,
  n_criterion = 3,
  reviewers_per_submission = 2,
  anchor_submissions = 2
)
peer_spec$peer_review$overview
```

`build_summary_table_bundle` *Build a manuscript-oriented table bundle from* `summary()` *outputs*

Description

Build a manuscript-oriented table bundle from `summary()` outputs

Usage

```
build_summary_table_bundle(
  x,
  which = NULL,
  appendix_preset = NULL,
  include_empty = FALSE,
  digits = 3,
  top_n = 10,
  preview_chars = 160
)

## S3 method for class 'mfrm_summary_table_bundle'
summary(object, digits = 3, top_n = 8, ...)

## S3 method for class 'mfrm_summary_table_bundle'
plot(
  x,
  y = NULL,
  type = c("table_rows", "role_tables", "appendix_roles", "appendix_sections",
    "appendix_presets", "selection_handoff_presets", "selection_tables",
   "selection_handoff", "selection_handoff_bundles", "selection_handoff_roles",
    "selection_handoff_role_sections", "selection_bundles", "selection_roles",
    "selection_sections", "numeric_profile", "first_numeric"),
  which = NULL,
  selection_value = c("count", "fraction"),
  appendix_preset = c("recommended", "compact", "all", "methods", "results",
    "diagnostics", "reporting"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	For `build_summary_table_bundle()`: An `mfrm_fit`, `mfrm_diagnostics`, `mfrm_precision_review`, `mfrm_fit_measures`, `mfrm_facets_fit_review`, `mfrm_person_fit_indices`, `mfrm_data_description`, `mfrm_reporting_checklist`, `mfrm_apa_outputs`, `mfrm_design_evaluation`, `mfrm_signal_detection`, `mfrm_recovery_simulation`, `mfrm_recovery_assessment`, `mfrm_population_prediction`, `mfrm_facets_run`, `mfrm_bias`, `mfrm_anchor_review`, `mfrm_linking_review`, `mfrm_misfit_casebook`, `mfrm_model_choice_review`, `mfrm_weighting_review`, `mfrm_calibration_score`, `mfrm_unit_prediction`, or `mfrm_plausible_values` object, one of their `summary()` outputs, or a compatible precomputed recovery-evidence summary. For `plot()`: Output from `build_summary_table_bundle()`.
which	For `build_summary_table_bundle()`: Optional character vector selecting a subset of named tables.

For plot(): Optional table selector used for numeric plot types.

appendix_preset
: For build_summary_table_bundle(): Optional appendix-oriented table preset: "all", "recommended", "compact", "methods", "results", "diagnostics", or "reporting". Cannot be combined with which. Section-aware presets keep returned tables whose bundle catalog maps to the requested appendix section.

  For plot(): Appendix preset used for selection_* plot types.

include_empty
: If TRUE, retain empty tables in the returned bundle.

digits
: For build_summary_table_bundle(): Digits forwarded when summary() must be computed from a raw object.

  For summary(): Number of digits used for numeric summaries.

top_n
: For build_summary_table_bundle(): Row cap forwarded to compact summary() methods when x is a raw object.

  For summary(): Maximum number of table-profile rows to keep.

preview_chars
: Character cap forwarded to summary.mfrm_apa_outputs() when x is a raw APA-output object.

object
: Output from build_summary_table_bundle().

...
: Reserved for generic compatibility.

y
: Reserved for generic compatibility.

type
: Plot type: "table_rows" for returned-table sizes, "role_tables" for returned-table counts by reporting role, "appendix_roles" for returned-table counts by reporting role under the bundle's appendix-routing contract, "appendix_sections" for returned-table counts by manuscript-facing appendix section, "appendix_presets" for conservative appendix-preset counts, "selection_handoff_presets" for workflow-only preset-level appendix handoff counts, "selection_tables" / "selection_handoff" / "selection_handoff_bundles" / "selection_handoff_roles" / "selection_bundles" / "selection_roles" / "selection_sections" for workflow-only appendix selection surfaces when present in the bundle, "numeric_profile" for column means from a selected numeric table, or "first_numeric" for the distribution of the first numeric column in a selected table.

selection_value
: For selection_* plot types, whether to plot exact counts ("count") or the corresponding exact fraction ("fraction") when that surface exposes one.

main
: Optional title override.

palette
: Optional named color overrides.

label_angle
: Axis-label rotation angle for bar-type plots.

draw
: If TRUE, draw using base graphics.

Details

This helper turns the package's compact summary objects into a reproducible table bundle for manuscript drafting, appendix handoff, or downstream formatting. It does not replace apa_table(); instead, it provides a consistent bridge from summary() to named data.frame components that can later be rendered with apa_table() or exported directly.

The public entry point validates x and the summary-object contract up front, so malformed summaries fail with a package-level message instead of falling through to opaque downstream errors.

The function first normalizes x through the corresponding summary() method when needed, then records a table_index describing every available table and returns the selected tables in tables. Optional appendix presets can be applied at bundle-construction time when you want a conservative manuscript-facing subset before plotting or export.

summary() method: Summarize a summary-table bundle for manuscript QC

For summary(): This summary is designed to answer a manuscript-facing question: which reporting tables are available, how large are they, which roles do they serve, and which of them contain numeric content suitable for quick plotting or appendix export.

plot() method: Plot a summary-table bundle for manuscript QC

For plot(): This helper keeps summary-bundle plotting conservative. It either visualizes the bundle's own bundle-level indexes ("table_rows", "role_tables", "appendix_roles", "appendix_sections", "appendix_presets") or routes a selected table through apa_table() and plot.apa_table() for numeric QC.

Value

An object of class mfrm_summary_table_bundle with:

- overview
- table_index
- plot_index
- tables
- appendix_preset
- notes
- source_class
- summary_class

For summary(): An object of class summary.mfrm_summary_table_bundle.

For plot(): A plotting-data object of class mfrm_plot_data.

Supported inputs

- mfrm_g_study(), mfrm_d_study(), mfrm_multivariate_d_compare(), mfrm_residual_network() or their mfrm_report() objects; supply saved intervals to mfrm_report() first. These routes retain all evidence tables, including signed estimates, settings and unavailable states, without top_n truncation.
- fit_mfrm() or summary(fit)
- diagnose_mfrm() or summary(diag)
- precision_review_report() or summary(precision_review)
- fit_measures_table() or summary(fit_measures)
- facets_fit_review() or summary(facets_fit_review)
- compute_person_fit_indices() or summary(person_fit)

- describe_mfrm_data() or summary(ds)
- reporting_checklist() or summary(chk)
- build_apa_outputs() or summary(apa)
- evaluate_mfrm_design() or summary(sim_eval)
- evaluate_mfrm_signal_detection() or summary(sig_eval)
- simulate_mfrm_recovery() or summary(rec)
- assess_mfrm_recovery() or summary(rec_assessment)
- a compatible precomputed recovery-evidence summary
- predict_mfrm_population() or summary(pred)
- the structural_design_review component of a design or prediction summary
- run_mfrm_facets() or summary(out)
- estimate_bias() or summary(bias)
- review_mfrm_anchors() or summary(review)
- build_linking_review() or summary(review)
- build_misfit_casebook() or summary(casebook)
- build_model_choice_review() or summary(review)
- build_weighting_review() or summary(review)
- predict_mfrm_units() or summary(pred_units)
- score_mfrm_calibration() or summary(scores); includes all review dispositions and unrounded conditional decisions when requested
- sample_mfrm_plausible_values() or summary(pv)

Interpreting output

- overview: one-row metadata about the source summary and table counts.
- table_index: table names, dimensions, roles, and manuscript-oriented descriptions.
- plot_index: which returned tables contain numeric content and which bundle-level plot types can use them directly.
- tables: named data.frame objects ready for formatting or export.
- appendix_preset: active appendix subset mode ("none" when not used).
- notes: short guidance about omitted empty tables or source-level caveats.
- fit-level caveats use the analysis_caveats role; pre-fit data score-support caveats use the score_category_caveats role. Both roles are classified as diagnostics and stay in recommended appendix subsets.
- recovery-assessment and recovery-validation summaries expose diagnostic_reporting_notes before diagnostic_review or diagnostic_oc_summary so fit/separation caveats can be reported without treating them as direct evidence of parameter recovery.
- recovery-validation summaries expose condition_reporting_notes before condition_summary so GPCM generator stress and sparse score support are not mistaken for recovery-metric failures.

- precision-review summaries expose `fit_separation_basis` so fit, ZSTD, separation/reliability/strata, and QC thresholds remain distinct forms of evidence rather than interchangeable summaries.
- fit-measure and FACETS fit-review summaries expose df/ZSTD sensitivity tables under precision-review roles, keeping MnSq status, ZSTD standardization, and external FACETS matching distinct in appendix handoffs.
- latent-regression fit summaries expose `population_coding` in the methods appendix role so categorical levels, contrasts, and encoded columns can be documented with the coefficient table.
- model-choice-review summaries expose `comparison_table`, `comparison_warnings`, `model_roles`, `downstream_routes`, and `report_templates` so RSM/PCM versus `GPCM` comparisons remain tied to their same-basis limits, equal-weighting, sensitivity, and reporting-boundary roles.

For `summary()`:

- `overview`: source class, returned-table count, note count, and whether a numeric table is available for plotting.
- `role_summary`: counts and total size by reporting role.
- `table_catalog`: complete returned-table registry with plot/export bridges.
- `table_profile`: table-level dimensions, numeric-column counts, and missing values for the largest returned tables.
- `plot_index`: which returned tables are plot-ready and which bundle-level numeric QC routes they support.
- `appendix_presets`: conservative `all` / `recommended` / `compact` plus section-aware `methods` / `results` / `diagnostics` / `reporting` appendix-export presets derived from table roles.
- `appendix_role_summary`: counts of returned tables by reporting role under the same conservative appendix routing used by the bundle catalog.
- `appendix_section_summary`: counts of returned tables by manuscript-facing appendix section.
- `selection_handoff_table_summary`: workflow-only table-level appendix handoff crosswalk when present in the bundle.
- `selection_handoff_preset_summary`: workflow-only appendix handoff overview aggregated at the preset level when present in the bundle.
- `selection_handoff_bundle_summary`: workflow-only appendix handoff overview aggregated at the bundle-by-section level when present in the bundle.
- `selection_handoff_role_summary`: workflow-only appendix handoff overview aggregated at the reporting-role level when present in the bundle.
- `selection_handoff_role_section_summary`: workflow-only appendix handoff overview aggregated at the reporting-role by appendix-section level when present in the bundle.
- `selection_summary`, `selection_table_summary`, `selection_table_preset_summary`, `selection_role_summary` `selection_section_summary`, and `selection_catalog`: preset-filtered appendix selection surfaces when workflow-only handoff tables are embedded in the bundle.
- `reporting_map`: where to go next for plotting, APA formatting, and export.
- `notes`: carried forward source-level caveats from the originating summary.

For plot():

- "table_rows": compares returned table sizes to show where reporting mass sits.
- "role_tables": shows how many returned tables belong to each reporting role.
- "appendix_roles": shows how returned tables contribute to conservative appendix routing by reporting role.
- "appendix_sections": shows how returned tables are distributed across methods/results/diagnostics/reporting sections.
- "appendix_presets": shows how many tables the current bundle contributes to the conservative appendix presets.
- "selection_handoff_presets": shows plot-ready appendix handoff counts by preset for workflow-only appendix routing surfaces in the bundle.
- "selection_tables" / "selection_handoff" / "selection_handoff_bundles" / "selection_handoff_roles" / "selection_handoff_role_sections" / "selection_bundles" / "selection_roles" / "selection_sections": show workflow-only appendix selection surfaces already materialized inside the bundle.
- "numeric_profile" / "first_numeric": reuse the same numeric QC logic as plot.apa_table() but start from a summary-table bundle.

Typical workflow

1. Build a compact object with summary(...).
2. Convert it with build_summary_table_bundle(...).
3. Use bundle$tables[[...]] directly, or hand a selected table to apa_table() for formatted manuscript output.
4. If you want a manuscript appendix subset up front, use a preset such as appendix_preset = "recommended", "compact", or "diagnostics".
5. For recovery-assessment or recovery-validation summaries, inspect bundle$tables$reading_order first when it is available.
6. For recovery-assessment or recovery-validation summaries with retained diagnostics, read diagnostic_reporting_notes before the raw diagnostic_review or diagnostic_oc_summary. Read condition_reporting_notes before condition_review or condition_summary when GPCM generator stress is part of the plan.

For summary():

1. Build bundle <- build_summary_table_bundle(summary(...)).
2. Run summary(bundle) to see reporting coverage.
3. Use plot(bundle, type = "table_rows") or plot(bundle, type = "numeric_profile", which = ...) for quick QC.

See Also

summary(), apa_table(), reporting_checklist(), build_apa_outputs(), compare_mfrm(), build_model_choice_review(), build_weighting_review()

build_summary_table_bundle(), apa_table(), plot()

build_summary_table_bundle(), apa_table(), plot.apa_table()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("build_summary_table_bundle", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
bundle <- build_summary_table_bundle(fit)
bundle$table_index
summary(bundle)$role_summary


## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_summary_table_bundle", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
bundle <- build_summary_table_bundle(fit)
summary(bundle)

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_summary_table_bundle", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
bundle <- build_summary_table_bundle(fit)
plot(bundle, draw = FALSE)
plot(bundle, type = "numeric_profile", which = "facet_overview", draw = FALSE)

## End(Not run)
```

`build_visual_summaries` *Build warning and narrative summaries for visual outputs*

Description

Build warning and narrative summaries for visual outputs

Usage

```
build_visual_summaries(
```

```
  fit,
  diagnostics,
  threshold_profile = "standard",
  thresholds = NULL,
  summary_options = NULL,
  whexact = FALSE,
  branch = c("original", "facets")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Output from diagnose_mfrm().
threshold_profile	Threshold profile name (strict, standard, lenient).
thresholds	Optional named overrides for profile thresholds.
summary_options	Summary options for build_visual_summary_map().
whexact	Use exact ZSTD transformation.
branch	Output branch: "facets" adds FACETS crosswalk metadata for manual-aligned reporting; "original" keeps package-native summary output.

Details

This function returns visual-keyed text maps to support dashboard/report rendering without hard-coding narrative strings in UI code.

thresholds can override any profile field by name. Common overrides:

- n_obs_min, n_person_min
- misfit_ratio_warn, zstd2_ratio_warn, zstd3_ratio_warn
- pca_first_eigen_warn, pca_first_prop_warn

summary_options supports:

- detail: "standard" or "detailed"
- max_facet_ranges: max facet-range snippets shown in visual summaries
- top_misfit_n: number of top misfit entries included

For GPCM, this helper returns caveated warning/summary maps over supported diagnostics, direct tables, and plots. The returned object includes gpcm_boundary so score-side, design-forecasting, DFF, and linking routes remain visibly separate capability rows.

Value

An object of class mfrm_visual_summaries with:

- warning_map: visual-level warning text vectors

- summary_map: visual-level descriptive text vectors
- warning_counts, summary_counts: message counts by visual key
- plot_payloads: reusable draw-free mfrm_plot_data objects for comparison, warning_counts, summary_counts, and optionally category_probability_surface
- public_plot_routes: public helper / draw-free route map for follow-up
- crosswalk: FACETS-reference mapping for main visual keys
- branch, style, threshold_profile: branch metadata

Interpreting output

- warning_map: rule-triggered warning text by visual key.
- summary_map: descriptive narrative text by visual key.
- strict marginal keys appear when diagnose_mfrm(..., diagnostic_mode = "both") supplies latent-integrated first-order and pairwise screening summaries.
- warning_counts / summary_counts: message-count tables for QA checks.
- plot_payloads: ready-to-reuse mfrm_plot_data objects for the bundle's own comparison/count plots and, when step estimates are available, the exploratory category_probability_surface data from plot(fit, type = "ccc_surface", draw = FALSE). The surface data carry category_support, interpretation_guide, and reporting_policy tables for zero-frequency category and reporting-boundary checks.
- public_plot_routes: draw-free helper routes for the dedicated public plot functions behind each visual family.

Typical workflow

1. inspect defaults with mfrm_threshold_profiles()
2. choose threshold_profile (strict / standard / lenient)
3. optionally override selected fields via thresholds
4. pass result maps to report/dashboard rendering logic

See Also

mfrm_threshold_profiles(), build_apa_outputs(), plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise")

Examples

```
# Long-running workflow; run explicitly when needed.
# example("build_visual_summaries", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
# A balanced slice retains every Rater and Criterion while running quickly.
toy <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", model = "RSM", quad_points = 7, maxit = 30
```

```
)
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
vis <- build_visual_summaries(fit, diag, threshold_profile = "strict")
vis2 <- build_visual_summaries(
  fit,
  diag,
  threshold_profile = "standard",
  thresholds = c(misfit_ratio_warn = 0.20, pca_first_eigen_warn = 2.0),
  summary_options = list(detail = "detailed", top_misfit_n = 5)
)
vis_facets <- build_visual_summaries(fit, diag, branch = "facets")
vis_facets$branch
summary(vis)
p <- plot(vis, type = "comparison", draw = FALSE)
p2 <- plot(vis, type = "warning_counts", draw = FALSE)
vis$plot_payloads$comparison$data$plot
vis$public_plot_routes[, c("Visual", "PlotHelper", "DrawFreeRoute")]
if (interactive()) {
  plot(
    vis,
    type = "comparison",
    draw = TRUE,
    main = "Warning vs Summary Counts (Customized)",
    palette = c(warning = "#cb181d", summary = "#3182bd"),
    label_angle = 45
  )
}

## End(Not run)
```

`build_weighting_review` *Build a weighting-policy review between Rasch-family and GPCM fits*

Description

Build a weighting-policy review between Rasch-family and GPCM fits

Usage

```
build_weighting_review(
  rasch_fit,
  gpcm_fit,
  theta_range = c(-6, 6),
  theta_points = 101L,
  top_n = 10L,
  nested = FALSE
)
```

```
## S3 method for class 'mfrm_weighting_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

rasch_fit	Output from fit_mfrm() using model = "RSM" or "PCM".
gpcm_fit	Output from fit_mfrm() using model = "GPCM".
theta_range	Numeric vector of length 2 passed to compute_information() for the information-redistribution comparison.
theta_points	Integer number of theta grid points passed to compute_information().
top_n	For build_weighting_review(): Maximum number of rows to keep in compact summary outputs. For summary(): Number of top rows to retain in compact summary tables.
nested	Request the PCM/GPCM equal-slope likelihood-ratio test. Default FALSE. Requires matched MML fits and the checks in compare_mfrm().
object	Output from build_weighting_review().
digits	Number of digits for printed numeric values.
...	Reserved for generic compatibility.

Details

build_weighting_review() is an operational model-choice review helper. It is designed for the common question:

- what changes when a Rasch-family equal-weighting model is replaced with a GPCM that allows discrimination-based reweighting?

The helper does not estimate a new model. Instead, it synthesizes four package-native evidence sources:

- compare_mfrm() for same-data model comparison
- the non-person facet measures from each fit
- the GPCM slope table
- compute_information() for design-weighted information redistribution

The result is intended for substantive review, not for automatic model selection. In particular, a better-fitting GPCM should not by itself be interpreted as a reason to discard an equal-weighting Rasch-family route. The fitted GPCM contains one slope for every level of one designated facet, not one common slope and not simultaneous criterion-by-rater slope blocks. The overview records the slope owner, step owner, level count, free relative slope contrasts, and whether the supplied reference is the exact unit-slope PCM response-kernel reduction. MML information-criterion ranking requires the likelihood and local-solution checks in compare_mfrm(). With nested = TRUE, a PCM/GPCM asymptotic chi-square LRT additionally requires matching population, step and facet settings and G-1 free slope contrasts. The returned comparison_contract records the comparison and test status; observed changes in scores and information need substantive interpretation. A JML log-likelihood increase is not promoted to automatic PCM-versus-GPCM model selection because it is unpenalized and the GPCM contains additional slope parameters. FACETS may serve as a

direct comparator for the PCM/JML side only; its post-fit discrimination statistic is not a jointly estimated free-slope GPCM counterpart.

`summary()` method: Summarize a weighting-review object

Value

An object of class `mfrm_weighting_review`.

For `summary()`: An object of class `summary.mfrm_weighting_review`, including the evidence-tier `comparison_contract` table and any computed `lrt`.

Recommended input route

1. Fit an equal-weighting reference model with `model = "RSM"` or `"PCM"`.
2. Fit a `GPCM` on the same prepared response data.
3. Run `build_weighting_review(rasch_fit, gpcm_fit)`.
4. Read `summary(review)` before deciding whether the discrimination-based reweighting is substantively acceptable.

What the returned tables mean

- `model_comparison`: same-data model-comparison bundle from `compare_mfrm()`. AIC/Person-BIC/SABIC ranking is available only when `ICComparable` is true. Inspect `$lrt` and `$comparison_basis$lrt_reason` for a requested test.
- `comparison_contract`: one-row evidence-tier table stating whether formal model selection is available, how any observed log-likelihood difference may be read, and the bounded role of FACETS in a JML review.
- `facet_shift`: how non-person facet estimates move under `GPCM`.
- `slope_profile`: which `slope_facet` levels are upweighted or downweighted.
- `information_redistribution`: within-facet information-share changes between the Rasch-family fit and `GPCM`.
- `top_reweighted_levels`: compact triage table for the strongest slope-facet-level redistribution signals.

GPCM boundary

This helper is available only for the current `GPCM` branch. It requires `slope_facet == step_facet` even though MML fitting permits separate owners, and should be read as an operational weighting-policy review, not as a formal validity adjudication.

See Also

`compare_mfrm()`, `compute_information()`, `gpcm_capability_matrix()`

`build_weighting_review()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("build_weighting_review", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
rasch_fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "MML",
  model = "RSM",
  quad_points = 9
)
gpcm_fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "MML",
  model = "GPCM",
  step_facet = "Criterion",
  slope_facet = "Criterion",
  quad_points = 9
)
review <- build_weighting_review(rasch_fit, gpcm_fit, theta_points = 41)
summary(review)
review$top_reweighted_levels

## End(Not run)
```

---

category_curves_report

Build a category curve export bundle (preferred alias)

Description

Build a category curve export bundle (preferred alias)

Usage

```
category_curves_report(
  fit,
  theta_range = c(-6, 6),
  theta_points = 241,
  digits = 4,
  include_fixed = FALSE,
  fixed_max_rows = 400,
  newdata = NULL
)
```

Arguments

fit	Output from fit_mfrm().
theta_range	Theta/logit range for curve coordinates.
theta_points	Number of points on the theta grid.
digits	Rounding digits for numeric graph output.
include_fixed	If TRUE, include a legacy-compatible fixed-width text block.
fixed_max_rows	Maximum rows shown in fixed-width graph tables.
newdata	Optional data frame with every non-Person facet, one row per rating context, using fitted level labels and no Theta column. For native GPCM MML, evaluates the full fitted predictor, including one or two slope families. Theta is reserved for the ability grid. Unknown/missing levels are refused; unobserved combinations of known levels are allowed and labelled. NULL retains reference profiles.

Details

Preferred high-level API for category-probability curve exports. model_context records the fitted step/slope owners. curve_profiles maps each CurveGroup to the owner levels, including separate-owner GPCM. By default, curves are conditional reference profiles with additive locations fixed at zero; see the retained CurveBasis and PredictorOffset columns. Explicit newdata enables full-context one- and two-family GPCM MML curves. This helper does not compute calibration SEs or confidence intervals. plot(out) and its selected curve views use base graphics. Automatic as_ggplot() conversion of this bundle is unavailable because curve families have distinct targets and scales; extract probabilities and retain its curve-family and group columns for custom figures. For fitted CCC graphics, plot(fit, type = "ccc") has a dedicated conversion; review its reference-profile settings before comparing curves. Returns tidy curve coordinates and summary metadata for quick plotting/report integration without calling low-level helpers directly. The expected-score table also carries the per-curve score variance and information function. For GPCM, the information column follows the Muraki/Samejima identity $a^2\mathrm{Var}(X \mid \theta)$; for RSM / PCM, this reduces to the usual score variance because discrimination is fixed at one. The category_information table decomposes that total into category-level contributions, $a^2 P_k(\theta)(k - E[X \mid \theta])^2$, whose sum equals the reported information at the same theta value. The cumulative_probabilities table follows the FACETS / Winsteps graph convention of accumulating modeled probabilities across ordered categories (P(X <= k) by default, with P(X >= k) also returned for flipped curves). cumulative_boundaries reports approximate theta values where P(X <= k) = .5, with BoundaryStatus and CrossingCount to avoid over-interpreting boundaries outside the requested theta range or with multiple crossings.

Without newdata, these are reference-profile curves. Estimated step parameters and, for GPCM, each step-facet group's estimated slope are retained, while additive facet main effects and fitted interactions are fixed at zero. The CurveBasis and PredictorOffset columns, and the corresponding entries in settings, make that conditioning explicit. Thus the curves do not represent an arbitrary observed Person-by-facet cell.

Value

A named list with category-curve components. Class: mfrm_category_curves.

Explicit rating contexts

With newdata, curve_profiles maps each curve to InputRow in the saved newdata. It retains the step owner/level, effective discrimination (the product for two families), additive PredictorOffset, and ObservedContext. The predictor is Theta + PredictorOffset; facet locations and interactions are retained. No Person posterior or population integration is involved. Category and expected scores retain the internal scoring scale; CategoryLabel and boundary label columns retain the original score labels. Plot series use these original labels without changing the scoring rule.

cumulative_boundaries solves P(X <= k) = .5, equivalently P(X >= k+1) = .5, directly within theta_range. It is independent of theta_points and retains full numerical precision irrespective of digits. Binary boundaries use the analytic adjacent intersection. Other boundaries use bracketed root finding on a stable difference of unnormalized lower and upper category masses, avoiding subtraction of rounded probabilities. The root tolerance is 1e-10 in native ability units: the mass difference must change sign within that distance on either side of the reported root. The probability residual must also be at most 1e-8. These are numerical checks, not statistical uncertainty bounds. Out-of-range roots remain NA with outside_theta_range; failed numerical checks remain NA with numerical_failure and a reason. AndrichThreshold is the centered step offset; AdjacentThreshold is the ability where the adjacent probabilities are equal, including the context offset. Neither is generally the cumulative ThurstonianThreshold. Positive discriminations order cumulative thresholds even when an intermediate category is never most probable; ordering alone does not establish category adequacy. SE, CI_Lower, and CI_Upper are NA and CIEligible is FALSE.

Finite calculations are retained even when the source fit needs review. fit_readiness, settings$source_converged and the summary/plot notes retain that distinction. Two-family values remain provisional. Unobserved contexts are predictions under the fitted model, not an identification guarantee. Save the complete result with saveRDS(); summary(), plot() and plot_data() replay it without fitting, diagnostics or covariance calculation.

Interpreting output

Use this report to inspect:

- where each category has highest probability across theta
- where cumulative category probabilities cross .5
- whether adjacent categories cross in expected order
- whether probability bands look compressed (often sparse categories)

Recommended read order:

1. summary(out) for compact diagnostics.
2. out$probabilities, out$expected_ogive, and out$category_information for custom graphics.
3. plot(out) for a default visual check, or plot(out, type = "cumulative") to inspect cumulative probabilities. plot(out, type = "information") to inspect curve-level information. Use plot(out, type = "category_information") when category-level contributions are needed.

References

Category response curves follow Andrich's rating-scale formulation, Masters' partial-credit model, and Muraki's generalized partial-credit model. The `Information` column for `GPCM` uses Muraki's item-information result obtained from Samejima's general polytomous information formula.

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176. doi:10.1177/014662169201600206
- Muraki, E. (1993). *Information functions of the generalized partial credit model*. Applied Psychological Measurement, 17(4), 351-363. doi:10.1177/014662169301700403

Typical workflow

1. Fit model with `fit_mfrm()`.
2. Run `category_curves_report()` with suitable `theta_points`.
3. Use `summary()` and `plot()`; export tables for manuscripts/dashboard use. `plot(out)` gives a four-panel overview. Use `preset = "monochrome"` for grayscale/line-type output and `boundary_status = "none"` when cumulative .5 boundary lines should be suppressed. `plot(out, type = "category_probability")` and `plot(out, type = "conditional_probability")` are explicit aliases for the same category-probability curves as `type = "ccc"`. The historical conditional alias is ordinary P(k), not the FACETS adjacent-conditional curve P(k)/(P(k-1)+P(k)); prefer the unambiguous `"ccc"` name. Use `plot_data(out, component = "plot_long")` when rebuilding the curves with ggplot2, plotly, or another R graphics system.

See Also

`category_structure_report()`, `rating_scale_table()`, `plot.mfrm_fit()`, mfrmr_reports_and_tables, mfrmr_visual_diagnostics

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
out <- category_curves_report(fit, theta_points = 101)
summary(out)
head(out$probabilities[, c("CurveGroup", "Theta", "Category", "Probability")])
p_overview <- plot(out, draw = FALSE)
p_overview$data$plot
p_cum <- plot(out, type = "cumulative", draw = FALSE)
head(p_cum$data$cumulative_boundaries)
p_info <- plot(out, type = "category_information", draw = FALSE)
head(p_info$data$category_information)
curve_long <- plot_data(out, component = "plot_long")
head(curve_long[, c("PlotType", "Theta", "Series", "Value")])
```

```
## Not run:
# Which cumulative boundaries apply to a specified rater and criterion?
ratings <- load_mfrmr_data("example_core")
gpcm_fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  model = "GPCM", method = "MML", step_facet = "Criterion",
  slope_facet = "Rater", rating_min = 1, rating_max = 4,
  category_policy = "preserve", quad_points = 61)
contexts <- unique(ratings[c("Rater", "Criterion")])[1:2, ]
curves <- category_curves_report(gpcm_fit, newdata = contexts)
curves$cumulative_boundaries[, c("CurveGroup", "UpperCategoryLabel",
  "ThurstonianThreshold", "AdjacentThreshold", "BoundaryStatus")]
summary(curves) # Inspect source readiness and interval limitations too.
plot(curves, type = "cumulative")

## End(Not run)
```

`category_structure_report` *Build a category structure report (preferred alias)*

Description

Build a category structure report (preferred alias)

Usage

```
category_structure_report(
  fit,
  diagnostics = NULL,
  theta_range = c(-6, 6),
  theta_points = 241,
  drop_unused = FALSE,
  include_fixed = FALSE,
  fixed_max_rows = 200
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional matching output from `diagnose_mfrm()`. Reuses stored uncertainty and marginal-fit results; omitted diagnostics do not trigger a covariance calculation.
`theta_range`	Theta/logit range used to derive transition points.
`theta_points`	Number of grid points used for transition-point search.
`drop_unused`	If `TRUE`, remove zero-count rows from the global category display. Owner-specific counts and their denominators remain complete.
`include_fixed`	If `TRUE`, include a legacy-compatible fixed-width text block.
`fixed_max_rows`	Maximum rows per fixed-width section.

Details

Preferred high-level API for category-structure diagnostics. This wraps the legacy-compatible bar/transition export and returns a stable bundle interface for reporting and plotting.

Value

A named list with category-structure components. Class: `mfrm_category_structure`.

Interpreting output

Key components include:

- category usage and descriptive residual table; category ZSTD and fit flags are unavailable because outcome-selected residuals lack a unit baseline
- threshold ordering and adjacent threshold gaps
- category transition-point table on the requested theta grid

Practical read order:

1. `summary(out)` for compact warnings and threshold ordering.
2. `out$category_table` for sparse/misfitting categories.
3. `out$median_thresholds` for cumulative 50% boundaries and their range status; `AndrichThreshold` separately retains adjacent-category steps.
4. `plot(out)` for quick visual check.

`owner_category_table`, `threshold_table`, `model_context` and `notes` follow `rating_scale_table()`, including source checks and uncertainty status. The global usage display does not replace owner-specific counts. `curve_profiles` maps each `CurveGroup` to its step/slope owners and levels. Curves use zero additive facet locations. `MedianThreshold` now solves a cumulative 50% boundary, replacing the old mislabeled adjacent-category step. Recreate saved category structure reports; no calibration refit is needed. Half-score crossings outside `theta_range` are NA with an explicit status. Mode peaks describe the finite grid, not an unbounded global maximum. Stored step SEs are reported only in `threshold_table`, not transferred to modal, half-score or cumulative boundaries. Two-family GPCM remains unsupported by this helper. Counts support `as_ggplot()`; modal-boundary and half-score views require the base renderer or explicitly grouped custom graphics from `plot_data()`.

Typical workflow

1. `fit_mfrm()` -> model.
2. `diagnose_mfrm()` -> residual/fit diagnostics (optional argument here).
3. `category_structure_report()` -> category health snapshot.
4. `summary()` and `plot()` for draft-oriented review of category structure.

See Also

`rating_scale_table()`, `category_curves_report()`, `plot.mfrm_fit()`, mfrmr_reports_and_tables, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("category_structure_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
out <- category_structure_report(fit)
summary(out)
head(out$category_table[, c("Category", "Count", "Infit", "Outfit")])
p_cs <- plot(out, draw = FALSE)
p_cs$data$plot

## End(Not run)
```

`classification_table` *Evaluate classification accuracy, consistency and conditional precision*

Description

Evaluate a fixed future rating plan under an ordinary RSM/PCM calibration. Lee-style recursion gives the exact finite summed-score distribution; the Rudner comparison uses a normal approximation to an ML ability estimate.

Usage

```
classification_table(
  fit,
  rating_plan,
  cut_scores,
  score_cuts,
  theta,
  theta_weights = NULL,
  operational_cuts = cut_scores,
  facets = NULL,
  readiness_policy = c("error", "review")
)
```

Arguments

`fit`	A native ordinary RSM/PCM `fit_mfrm()` result (MML or JML), without interactions, random effects or a JML correction.
`rating_plan`	Data frame with one row per future rating and exactly the calibrated facet identifier columns. All levels must be known. Duplicate complete facet cells, missing identifiers, score columns and observation weights are not supported. Each row contributes one rating.

`cut_scores` Finite, strictly increasing true-ability category cuts on the calibration scale. Optional names label the cuts.

`score_cuts` Nondecreasing integer cuts on the sum of **model category indices**, each rating scored `0, ..., K-1`. One cut per `cut_scores`. Values from zero through the maximum possible sum plus one are allowed. Repeated cuts retain empty categories; zero and maximum-plus-one retain unreachable edge categories. These are explicit operational decisions, not automatically rounded expected scores or sums of original labels.

`theta` Finite numeric vector of true-ability points to evaluate. Supply a grid around the cuts to inspect conditional precision there. Fitted EAPs are not automatically substituted for true abilities.

`theta_weights` Optional nonnegative population point/quadrature weights corresponding to `theta`, with at least one positive entry. Weights are normalized and retained. Density values alone are generally insufficient on an unequally spaced grid. No default population is used.

`operational_cuts`
Strictly increasing ability-estimate decision cuts for the Rudner approximation, one per true cut. Defaults to `cut_scores`.

`facets` Optional character vector mapping calibrated facet names to `rating_plan` columns, as in `predict_mfrm_units()`.

`readiness_policy`
`"error"` requires a ready source calibration. `"review"` permits a nonready but valid source and retains its assessment in `source`; it does not establish statistical adequacy. Invalid retained parameters cannot be overridden.

Details

Both methods are returned. For an operational decision based on the declared category-score sum, report the `lee_sum` row. For a decision based on an ML ability estimate, `rudner_normal` is an approximate comparison, subject to the limitations below. It does not evaluate an EAP/MAP decision rule. The rows need not describe the same decision; compare their accuracy as competing calculations only after matching the actual operational rules. `Status = "available"` means the calculation succeeded, not that the normal approximation or the fitted model is accurate for the intended population.

Categories are numbered from 1. Values equal to a cut enter the higher category: lower endpoints are inclusive, upper endpoints exclusive. The top category includes the maximum possible sum. True categories use `cut_scores`; operational categories use `score_cuts` for `lee_sum` and `operational_cuts` for `rudner_normal`. These can describe different rules. Expected sums at ability cuts are reported for inspection, never rounded into a decision rule. In an ordinary fixed-plan Rasch model the ML score equation is `S = E(S | theta)`. A matched sum cut therefore requires the actual ML decision mapping, including ties and extreme-score policy. EAP/MAP decisions do not automatically share that mapping.

With conditional operational category probabilities `p[h]` and true category `H(theta)`, accuracy is `p[H(theta)]`; consistency is `sum(p^2)`. Consistency concerns two conditionally independent repetitions with the **same** calibrated plan. Overclassification and underclassification sum probabilities above and below the true category. Population versions are weighted averages over the

supplied true-ability distribution. The joint matrix contains population joint probabilities, not row-normalized error rates. Accuracy need not exceed consistency in a multicategory decision. With two categories, marginal overclassification is the joint probability of a true fail and an observed pass; underclassification is the joint probability of a true pass and an observed fail. Divide each by its true category's population probability for the error rate conditional on that category. A zero denominator makes that conditional rate undefined. Conditioning instead on the observed pass/fail category uses its column sum in the joint matrix and answers how often that reported decision is wrong.

Information sums conditional category-score variances over the plan. `SEM = 1/sqrt(Information)` is a conditional **ability-scale** SEM with calibration fixed, not a posterior SD, sum-score SEM or an SE propagating facet estimation uncertainty. `rudner_normal` assumes an ML estimator's sampling distribution is normal with mean `theta` and this SEM. Short plans, extremes, estimator bias and model misfit can invalidate that approximation. Zero numerical information gives infinite SEM and makes Rudner results unavailable; Lee probabilities can remain available. Arithmetic overflow or unresolved probability mass causes an error. Narrow normal-probability bands use direct integration when subtraction of cumulative probabilities would lose precision. Finite precision does not guarantee relative accuracy in vanishing tails.

Local independence is assumed even when rows share a task or rater. Calibration, cuts and the plan are held fixed. There is no uncertainty interval, quality threshold, automatic cut selection, random-rater/task generalization, retake policy, or propagation of calibration/population estimation error. Model-based accuracy is not observed reference agreement. This is also distinct from posterior ability-band probabilities in `predict_mfrm_units()`. Passing calibration checks does not validate the future plan or the normal approximation.

Value

A plain list. `table` contains conditional indices, expected sum, information, SEM and availability for each point/method. `SourceStatus` in `table`, `probabilities`, `scores`, `cuts`, `marginal` and `classification_matrix` preserves the separate calibration assessment, including in CSV exports. `Status`, where present, describes numerical availability; an available row with `SourceStatus = "review_only"` still needs source review. Inspect `source$reason_codes` for the reasons. `probabilities` contains operational category probabilities; `scores` contains the exact summed-score probability mass. `cuts` reports both cut scales and SEM at true and operational ability cuts. `marginal` and `classification_matrix` are empty unless population weights are supplied. Any unavailable point with positive weight makes its method's marginal results unavailable; such points are never dropped or renormalized away. `population` retains input points and raw/normalized weights. `rating_plan`, `calibration` (offsets, cumulative thresholds and category map), `source`, `settings` and `notes` retain the calculation's conditions. A source review status applies to the entire result, including numerically available rows. Save with `base::saveRDS()` or export tables with `utils::write.csv()`. A table CSV alone does not retain the plan, cut definitions, population distribution and notes; keep the full result for reproducible reuse. Reading saved output does not refit or recalculate anything.

References

Rudner, L. M. (2001). Computing the expected proportions of misclassified examinees. *Practical Assessment, Research, and Evaluation, 7*, Article 14. doi:10.7275/an9m2035.

Rudner, L. M. (2005). Expected classification accuracy. *Practical Assessment, Research, and Evaluation, 10*, Article 13. doi:10.7275/56a56b14.

Lee, W.-C. (2010). Classification consistency and accuracy for complex assessments using item response theory. *Journal of Educational Measurement, 47*(1), 1-17. doi:10.1111/j.17453984.2009.00096.x.

Wyse, A. E., & Hao, S. (2012). An evaluation of item response theory classification accuracy and consistency indices. *Applied Psychological Measurement, 36*(7), 602-624. doi:10.1177/0146621612451522.

Lathrop, Q. N., & Cheng, Y. (2013). Two approaches to estimation of classification accuracy rate under item response theory. *Applied Psychological Measurement, 37*(3), 226-241. doi:10.1177/0146621612471888.

See Also

predict_mfrm_units(), score_comparison_table()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("classification_table", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(ratings, person = "Person", facets = c("Rater", "Criterion"),
  score = "Score", method = "MML", model = "RSM",
  rating_min = 1, rating_max = 4, category_policy = "preserve")
# The planned roster includes the intended ratings even if scores are missing.
data("mfrmr_example_operational_design", package = "mfrmr")
rating_plan <- subset(mfrmr_example_operational_design, Person == "P001",
  select = c("Rater", "Criterion"))
rating_plan # Six ratings: two raters, each scoring three criteria.
# Illustrative rules: ability at least 0 logits; original-score sum at least 18.
# score_cuts uses 0:3 category indices, not the original 1:4 score labels:
# subtract 1 for each of the six ratings, so 18 - 6 = 12.
# These are different decision rules, not automatically equivalent cutoffs.
result <- classification_table(fit, rating_plan,
  cut_scores = 0, score_cuts = 12, theta = c(-0.5, 0, 0.5))
result$table
result$cuts
result$settings$calibration # These probabilities hold calibration fixed.
# No population average is inferred from this three-point inspection grid.

## End(Not run)
```

---

compare_mfrm *Compare two or more fitted MFRM models*

Description

Produce a side-by-side comparison of multiple fit_mfrm() results using AIC, Person-based BIC, Sclove SABIC, log-likelihood, and free-parameter counts. An ordinary RSM paired with a testlet or shared-rater fit instead follows a descriptive facet-effect route, with no automatic ranking. For ordinary models, when exactly two models are supplied and the current conservative nesting review passes, a likelihood-ratio test is included.

Usage

```
compare_mfrm(
  ...,
  labels = NULL,
  warn_constraints = TRUE,
  nested = FALSE,
  response_diagnostics = NULL,
  person_scores = NULL
)

## S3 method for class 'mfrm_extended_comparison'
summary(object, ...)

## S3 method for class 'mfrm_extended_comparison'
print(x, ...)

## S3 method for class 'mfrm_extended_comparison'
plot(
  x,
  style = c("paired", "difference"),
  facet = NULL,
  draw = TRUE,
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = metric %in% c("effects", "infit", "outfit"),
  text_scale = 1,
  point_size = 2.5,
 metric = c("effects", "expected_score", "variance", "infit", "outfit", "probability",
    "person"),
  category = NULL,
  ...
)
```

Arguments

`...`	For `compare_mfrm()`: Two or more `mfrm_fit` objects, or exactly one ordinary `mfrm_fit` and one `fit_mfrm_testlet()` or `fit_mfrm_random_rater()` result. The latter route is descriptive; see the extended-model section. For `plot()`: Unused.
`labels`	Optional character vector of labels for each model. If `NULL`, labels are generated from model/method combinations.
`warn_constraints`	Logical. If `TRUE` (the default), emit a warning when models use different score coding or constraint settings. Ranking is suppressed whenever this basis differs, regardless of whether the warning is displayed.

`nested` Logical. Set to `TRUE` only when the supplied models are known to be nested and fitted with the same likelihood basis on the same observations. The default is `FALSE`, in which case no likelihood-ratio test is reported. When `TRUE`, the function still runs a conservative structural nesting review and computes the LRT only for supported nesting patterns.

`response_diagnostics` Optional list of two saved `mfrm_response_diagnostics()` outputs in the same order as the fits. Supported for an ordinary RSM paired with a testlet or shared-rater fit. This adds descriptive predictive comparisons without computing integrals.

`person_scores` Optional list of two saved `score_mfrm_persons()` results in fit order for an ordinary versus extended RSM comparison. Both must score the same complete source roster, return the same Person IDs and use the same conditional interval level. No scoring is run here.

`object` An extended-model comparison returned by `compare_mfrm()`.

`x` A descriptive extended-model result from `compare_mfrm()`.

Also for `compare_mfrm()`: An extended-model comparison returned by `compare_mfrm()`.

`style` `"paired"` plots comparison against reference with an equality line. `"difference"` plots comparison minus reference against their mean.

`facet` Panels to display; `NULL` includes all. For effects and group indices these are facet names; for row moments the panel is `"Selected ratings"`; for probabilities it is `"Score k"` for category k.

`draw` `FALSE` returns plot data without opening a device.

`palette` `"accessible"` uses blue points and a dashed grey reference; `"mono"` uses dark grey points. Meaning does not depend on colour.

`title, caption` `NULL` uses the default text; a string replaces it; `""` removes it. Interpretation notes remain in the saved plot data.

`show_title, show_notes` Show the title and caption, respectively. `FALSE` hides text without deleting its metadata.

`show_labels` Show matched IDs beside the points. Defaults to `TRUE` for effects and group indices, `FALSE` for rating-level moments and probabilities. Consult the retained table when labels are hidden.

`text_scale` Positive multiplier for text size.

`point_size` Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

`metric` `"effects"` compares centered facet locations (default). `"expected_score"`, `"variance"`, `"infit"`, `"outfit"` and `"probability"` require saved `response_diagnostics` in `compare_mfrm()`. These compare the same posterior predictive target in original score units, squared score units, descriptive mean squares or probabilities. `"person"` requires saved `person_scores` and compares EAPs centered at each fitted population mean, in logits. No difference interval is drawn.

`category` Numeric score categories to display with `metric = "probability"`; `NULL` includes all categories.

Details

GPCM MML and RSM/PCM MML with an estimated normal population use a separate local-solution check for information criteria, without refitting. It checks the retained likelihood, terminal gradient and positive unregularized observed information. `ICFitEligible`, `ICFitBasis` and `ICFitReview` record that decision; `ICComparable` also requires the common data/likelihood and integration checks. This can allow IC comparison while `InferenceReady` remains false for intervals or tests. It shares the joint-information calculation and workspace budget with `confint.mfrm_fit()`. Unknown, unstable or unavailable results remain ineligible with a reason. A local check does not prove a global maximum or negligible integration error; examine starting-value and `mml_quadrature_sensitivity()` results when the decision is close. Known marginal response-dimension deficits and complete-pattern first-order rank failures also prevent inferential model comparison, even with a positive numerical Hessian. More Persons with the same response designs improve precision but cannot repair a structural dimension deficit. Positive but ill-conditioned information may pass with a caution after numerical refinement, unregularized inversion and curvature-scaled gradient checks. A warning is emitted and retained in `ICFitCaution`, `ICFitReview` and the requested LRT's interpretation; it does not establish finite-sample accuracy for model ranking or tests.

Models should be fit to the **same data** (same rows, same person/facet columns) for the comparison to be meaningful. The function checks that observation counts match and warns otherwise.

Information-criterion ranking is reported only when all candidates are eligible solutions under the package's current `MML` contract, use the same prepared observations, score coding, constraints, formula contract, and integration-evaluation identity, are eligible under the weighting policy, and have a selectable integration tier. For fixed-facet MML, `ICSampleSize` is the number of independent Persons and `ICSampleSizeBasis` is `"person_count"`; response rows and their weighted total are retained separately. Explicit all-unit weights are eligible, whereas every non-unit observation-weight fit fails closed under the current package contract.

Canonical `AIC`, `BIC`, and `SABIC` are recomputed from the retained objective, optimizer-vector dimension, and Person count. A stale stored value, a legacy object without the current contract identity, a JML fit, or an ineligible weighted fit cannot enter ranking. Earlier raw values may be shown only as explicitly labelled `LegacyAIC` and `LegacyBIC` fields.

Integration selection guard: raw canonical criteria are retained at every valid MML quadrature count, but automatic comparison is screening-only below 15 points and review-only at 15–30 points. `ICSelectable` and `ICIntegrationSelectable` become `TRUE` at q>=31. q=31 is a starting grid, not a universal guarantee: close or consequential comparisons should be reevaluated at a denser shared grid (normally q>=61). The guard also applies to `nested = TRUE`, so a coarse-grid LRT cannot bypass it.

Nesting: Two models are *nested* when one is a special case of the other obtained by imposing equality constraints. The most common nesting in MFRM is RSM (shared thresholds) inside PCM (item-specific thresholds). Models that differ only in estimation method (MML vs JML) on the same specification are not nested in the usual sense, and their information criteria do not share the common MML contract. Do not use either the LRT or IC ranking as a direct cross-method comparison.

In the **current** `mfrmr` **model space**, the automatic nesting review is intentionally conservative. It currently supports the following restrictions under shared data and shared constraints:

- `RSM` nested inside `PCM` when the `PCM` fit has an explicit `step_facet` and retains every `facet_interactions` term in the RSM. The PCM may add interaction terms; omitting an RSM interaction is not repaired by freeing category steps;

- PCM nested inside GPCM with the same step facet, population design, other facet/step constraints and interactions. The only additional parameters must be G-1 relative log-slope contrasts for G slope levels. The GPCM slope owner may differ from the shared step owner;
- same-family additive-vs-interaction comparisons when the smaller fit's facet_interactions set is a subset of the larger fit's set.

Cross-method comparisons, comparisons that change anchors/dummying/centering, and same-family comparisons that do not add fixed interaction terms are not automatically promoted to LRT claims.

Automatic nesting also requires a shared population specification: either both fits use the fixed standard-normal population or both use the same estimated-normal design, with the same declared columns and design values aligned by Person. Coefficient and variance estimates may differ. Row and column permutations are aligned; other recodings, changed designs, and unavailable design metadata require separate review. Passing this check does not grant inference readiness to an estimated-population fit. For a matched PCM/GPCM pair, the local-solution checks described above supply the numerical requirement independently of slope-interval availability.

The **likelihood-ratio test (LRT)** is reported only when exactly two models are supplied, nested = TRUE, the structural nesting review passes, and the difference in the number of parameters is positive:

$$\Lambda = -2(\ell_{\text{restricted}} - \ell_{\text{full}}) \sim \chi^2_{\Delta p}$$

The LRT is asymptotically valid only under its regularity assumptions. With small samples or boundary/singular conditions, the reference chi-square p-value can be incorrect. At large Person counts, a very small practical improvement can also become statistically significant. Read the p-value as formal nested-fit evidence, not as an automatic practical model preference or evidence that a subscore is useful.

plot() method: Plot matched facet effects or posterior predictions across MFRMs

For plot(): Points are centered at the unweighted mean within each complete matched facet. These displays compare fitted summaries, including shrinkage, rather than testing differences or choosing a better model. The equality/zero line is a descriptive reference. No difference intervals or limits of agreement are computed. Unavailable rows remain in plot data. Select facet or use show_labels = FALSE for crowded displays.

Predictive quantities are not centered. Each row label maps through data$response_events and the comparison's responses$rows, which retain both original input row numbers. Infit/Outfit use selected rows in each group; posterior conditioning retains all observed source ratings. Smaller residual indices do not establish better prediction or model adequacy. There are no ordinary fit cutoffs, tests or automatic ranking.

Value

With one extended model, an mfrm_extended_comparison containing models, checks, centered effects, notes, and source/omission metadata. With ordinary models only, an object of class mfrm_comparison (named list) with:

- table: data.frame of model-level statistics including LogLik, Deviance, Npar (and equal compatibility alias npar), AIC, BIC, SABIC, criterion deltas and candidate-set weights, ResponseRows,

WeightedResponseTotal, Persons, ICSampleSize, formula and integration identities, integration tier/selectability, stored-value consistency fields, convergence fields, ICComparable, SABICComparable, and ConstraintComparable.

- lrt: data.frame with likelihood-ratio test result (only when two models are supplied and nested = TRUE). Contains ChiSq, df, p_value.
- evidence_ratios: data.frame of pairwise Akaike-weight ratios (Model1, Model2, EvidenceRatio, LogEvidenceRatio, RatioReason). NULL when comparison is ineligible. Extremely small normalized weights can round to zero.
- preferred: named list with the preferred model label by each criterion.
- comparison_basis: list describing whether IC and LRT comparisons were considered comparable. Includes a conservative nesting_review plus lrt_status / lrt_reason so withheld LRTs are explicit rather than silently absent.

For plot(): Invisibly, an mfrm_plot_data object with source effects, checks, interpretation notes and a text alternative. as_ggplot() supports further styling without estimation. Both renderers respect the display controls.

Ordinary versus extended RSMs

Exactly one ordinary RSM MML fit and one testlet/shared-rater fit return an mfrm_extended_comparison, with models, checks, effects and notes. The information-criterion and LRT sections describe ordinary mfrm_fit comparisons only. This route does not rank models, compute difference intervals, or provide an AIC/BIC preference. nested = TRUE is refused.

Both fits must use identical observed rating events, including repeated events, category coding, unit weights and fixed facets except for the deliberately changed rater treatment. Use unanchored additive severity facets, noncenter_facet = "Person", no facet shrinkage, and matching person/score column roles. Category recoding must preserve the same consecutive adjacent-category scale. These checks cannot be disabled by warn_constraints = FALSE.

Match the population assumption: ordinary population_formula = ~1 with both extension defaults estimates a common normal ability SD; the ordinary default and extension person_sd = 1 impose known N(0,1). Covariate-dependent populations and fixed nonunit SDs have no matching ordinary route here. Population intercepts and free step locations may use different origins; they are recorded, not compared as substantive mean differences.

The source roster and omitted counts must agree. New ordinary fits retain prep$omitted_data and prep$omitted_input_rows. When scores were omitted, their identities must match, including repeated events. Older fits without this provenance require refitting from the complete assigned-score roster. Missing IDs, excluded weights and population-data omissions are unsupported.

Effects are centered at the unweighted mean of the same complete set of levels within each facet. This preserves pairwise contrasts while removing arbitrary facet origins. SourceReference/SourceComparison retain raw values and CenterReference/CenterComparison retain the subtracted means. Difference is comparison minus reference. Conditional rater modes and fixed coefficients are labeled separately; shrinkage is not evidence of greater accuracy or rater quality. Failed numerical checks or a missing level estimate withhold affected differences without dropping source rows. Native inference readiness remains explicit even when descriptive numerical checks pass. No SE or confidence interval for model differences is implied.

With `response_diagnostics`, `responses` retains matched category probabilities, means, full mixture variances and grouped descriptive Infit/Outfit. Both saved diagnostics must use the same probability target, selected event multiset and shared `group_by` identifier columns. Matching uses event contents, including repeated-event multiplicities and missing scores, rather than assuming row numbers agree. Identical repeated events match in source occurrence order. `responses$events` and `responses$rows` retain the matching and original row numbers. Missing or unavailable rows remain explicit and differences involving them are withheld.

All observed source events condition each model; selecting a Person's rows does not remove other Persons from shared-rater inference. These are same-data descriptive comparisons with calibration held fixed, not held-out predictive performance. A smaller Infit/Outfit is not evidence of improvement. Do not substitute ordinary plug-in diagnostics or their reference cutoffs. See `plot.mfrm_extended_comparison()` for predictive paired and difference displays.

With `person_scores`, `persons$table` compares conditional EAPs after subtracting each fitted population mean, retaining original estimates, origins, posterior SDs and aligned conditional endpoints. Unit Rasch slopes retain the logit unit; centering removes the arbitrary origin, not shrinkage or changes in the assumed population variance. Prior-only and unavailable differences are withheld. Separate conditional intervals are not intervals for model differences or tests of Person differences. Use `metric = "person"` in `plot.mfrm_extended_comparison()` for a paired/difference display.

The default facet-only output does not compare Person scores, steps, predictive Infit/Outfit or replacement-rater predictions. For testlets, zero local variance retains the fixed facets; zero shared-rater variance removes rater differences and is not the fixed-rater model. Numerical zero-reduction agreement does not establish model adequacy or variance-boundary inference.

Use `plot(comparison, style = "paired")` for matched effects, or `style = "difference"` for differences versus means. See `plot.mfrm_extended_comparison()` for display and accessibility controls. Attach saved results with `mfrm_results(extended_fit, comparison = comparison)` for static reports, tables, figures and replay. No comparison/report step fits, scores or resamples either model.

Information-criterion diagnostics

In addition to the canonical criteria, the function computes:

- **Delta_AIC / Delta_BIC / Delta_SABIC**: difference from the minimum value in the supplied candidate set. For AIC, the often-used Delta ranges below 2, 4–7, and above 10 are rough descriptions of decreasing relative support (Burnham & Anderson, 2002), not significance thresholds, adequacy checks, or universal rules for BIC and SABIC.
- **AkaikeWeight / BICWeight / SABICWeight**: relative candidate-set weights derived from `exp(-0.5 * Delta)` and normalized only across the supplied models. They are not posterior probabilities, probabilities of model truth, or guarantees that the candidate set is adequate.
- **Evidence ratios**: pairwise ratios of Akaike weights, quantifying relative support within this candidate set. A ratio of 5 means only that one normalized Akaike weight is five times the other; it does not mean that one model is five times more likely to be true. Ratios are computed from AIC differences, so they remain available when both normalized weights round to zero. `LogEvidenceRatio` retains the natural log ratio; `RatioReason` explains `NA` ratios outside numeric range (`"overflow"`, `"underflow"`, or `"nonfinite_difference"`, otherwise `"available"`).

AIC, BIC, and SABIC answer different approximation/penalty questions. SABIC is a sensitivity criterion, not a universal tie-breaker. At 22 or fewer Persons its Sclove penalty is non-positive, so `SABICSelectable` and `SABICComparable` are `FALSE` and no automatic SABIC preference is returned.

What this comparison means

`compare_mfrm()` is a same-basis model-comparison helper. Its strongest claims apply only when the models were fit to the same response data, under a compatible likelihood basis, and with compatible constraint structure.

What this comparison does not justify

- Do not treat AIC/BIC/SABIC differences as primary evidence when `table$ICComparable` is `FALSE`.
- Do not infer numerical adequacy merely because all models share the same coarse quadrature identity. Below q=31, raw criteria are diagnostic only and automatic deltas, weights, preferences, and LRT are suppressed.
- Do not interpret the LRT unless `nested = TRUE` and the structural nesting review in `comparison_basis$nesting_revie` passes.
- Same-family additive-vs-interaction fits are considered nested only when all other structural settings match and the smaller model's `facet_interactions` set is a subset of the larger model's set.
- Do not assume that `nested = TRUE` overrides the package's conservative nesting boundary; unsupported relations remain unsupported.
- PCM is the unit-slope reduction of an aligned GPCM. `nested = TRUE` tests equal relative slopes only after population, constraint, free-dimension and numerical checks pass. The relation is recorded as `PCM_in_GPCM`. Default PCM uses a fixed standard-normal population, whereas default GPCM estimates its mean and variance: changing only `model` does not isolate slope differences. Supply `population_formula = ~1` and the same `person_data` to both fits for an estimated-normal comparison. Unit slopes are interior positive values; the reference is ordinary asymptotic chi-square with G-1 degrees of freedom, not a boundary mixture. IC eligibility alone does not establish nesting or qualify slope intervals.
- Do not compare models fit to different datasets, different score codings, or materially different constraint systems as if they were commensurate.
- At large Person counts, a small systematic likelihood gain can dominate an IC difference; boundary or singular fits can also invalidate routine asymptotics. Evaluate effect size, stability, interpretability, and score consequences separately.

Interpreting output

- Lower AIC/BIC values indicate relative support only when `table$ICComparable` is `TRUE`; use SABIC ranking only when `table$SABICComparable` is also `TRUE`.
- A significant LRT p-value is formal evidence against the restricted model under the stated assumptions; practical gain, score utility, and dimensional interpretation require separate checks.
- `preferred` identifies the minimum criterion within the supplied candidate set; it is not an unconditional model verdict.

- `evidence_ratios` gives pairwise Akaike-weight ratios (returned only when Akaike weights can be computed for at least two models).
- When comparing more than two models, interpret evidence ratios cautiously—they do not adjust for multiple comparisons.

How to read the main outputs

- `table`: first-pass comparison table; start with `ICComparable`, `ICSelectable`, `ICIntegrationTier`, `Model`, `Method`, `ICStatus`, `AIC`, `BIC`, and `SABIC`.
- `comparison_basis`: records whether IC and LRT claims are defensible for the supplied models. Inspect `comparison_basis$nesting_review$relation` and `reason` before reading any LRT output.
- `lrt`: nested-model test summary, present only when the requested and reviewed conditions are met.
- `preferred`: candidate preferred by each criterion when those summaries are available.

Recommended next step

Inspect `comparison_basis` before writing conclusions. If comparability is weak, treat the result as descriptive and revise the model setup (for example, explicit `step_facet`, common data, or common constraints) before using IC or LRT results in reporting.

Typical workflow

1. Fit two models with `fit_mfrm()` (e.g., PCM and GPCM) on the same prepared rows, explicit step owner, constraints, and MML quadrature setting. Use at least 31 common quadrature points for selectable ICs.
2. Compare with `compare_mfrm(fit_pcm, fit_gpcm)` and start by checking `comparison$table$ICComparable`.
3. Inspect `summary(comparison)` for AIC/BIC/SABIC diagnostics and the reasons for withholding ranking or tests. GPCM IC ranking has separate solution checks; request the matched PCM/GPCM test with `nested = TRUE`.
4. Use `build_weighting_review(fit_pcm, fit_gpcm)` to inspect which selected facet levels and information shares were reweighted by the slopes.

References

- Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed.). Springer.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716-723.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*(2), 461-464.
- Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika, 52*(3), 333-343.

See Also

`fit_mfrm()`, `diagnose_mfrm()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("compare_mfrm", run.dontrun = TRUE)
## Not run:
# Load one dataset for both models
library(mfrmr)
toy <- load_mfrmr_data("example_operational")

# RSM: shared category thresholds
fit_rsm <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  model = "RSM"
)

# PCM: separate category thresholds for each criterion
fit_pcm <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  model = "PCM",
  step_facet = "Criterion"
)

# Check that the fitted models can be compared
comparison <- compare_mfrm(fit_rsm, fit_pcm, labels = c("RSM", "PCM"))
comparison$table[, c("Label", "Converged", "ICComparable", "ICSelectable")]

# Smaller AIC/BIC indicates better relative support among these models
comparison$table[, c("Label", "AIC", "Delta_AIC", "BIC", "Delta_BIC")]
# Delta is the difference from the lowest value of that criterion
# For close or consequential comparisons, check a denser shared grid with
# mml_quadrature_sensitivity() before choosing a model
# For a PCM/GPCM equal-slope test, see the matched-population example in
# vignette("mfrmr-gpcm-scope"); request compare_mfrm(..., nested = TRUE).

## End(Not run)
```

`compatibility_alias_table` *List retained compatibility aliases and preferred names*

Description

List retained compatibility aliases and preferred names

Usage

```
compatibility_alias_table(
  scope = c("all", "functions", "arguments", "fields", "columns", "plot_metrics")
)
```

Arguments

scope	Which alias surface to return: "all", "functions", "arguments", "fields", "columns", or "plot_metrics".

Details

This helper is a compact public registry of the compatibility aliases that mfrmr intentionally keeps visible for older scripts and downstream handoffs. It is meant to answer two questions quickly:

1. Which old names are still accepted?
2. Which package-native names should new code use instead?

Non-exported soft-deprecated helpers are deliberately excluded here. This table is only for retained user-facing aliases that remain part of the public surface.

Value

A data.frame with one row per retained alias and columns:

- Alias
- PreferredName
- Surface
- Lifecycle
- RetainedFor
- RemovalPlan
- Notes

Typical workflow

1. Call compatibility_alias_table() when reading older scripts or reports.
2. Use PreferredName when updating older analysis code.
3. Prefer the package-native name in all new outputs and scripts.

See Also

mfrmr_compatibility_layer, run_mfrm_facets(), analyze_dff(), reporting_checklist(), fair_average_table(), plot(x, type = "fair_average")

Examples

```
compatibility_alias_table()
compatibility_alias_table("functions")
compatibility_alias_table("fields")
compatibility_alias_table("columns")
```

`compute_facet_design_effect` *Compute descriptive design-effect approximations for each facet*

Description

Combines per-facet average cluster size with ICC estimates to return the Kish-style approximation `Deff = 1 + (m - 1) * rho`, where `m` is the average number of observations per facet element and `rho` is the ICC variance share. Each facet is evaluated separately.

Usage

```
compute_facet_design_effect(
  data,
  facets,
  icc_table = NULL,
  score = NULL,
  person = NULL,
  missing = c("error", "omit")
)
```

Arguments

`data`	Data frame in long format, used to fit the ICC model when `icc_table` is `NULL`. With a supplied ICC table, sample sizes come from its retained row accounting, not from `data`.
`facets`	Character vector of facet column names.
`icc_table`	Output from `compute_facet_icc()` (optional; will be computed on the fly when `NULL`). Must retain its `data_usage` attribute. Rerun older saved ICC results from their original data and settings before calculating design effects; their analysis sample cannot be reconstructed from the ICC table alone.
`score`	Score column name; required when `icc_table` is `NULL`.
`person`	Person column; passed through to compute_facet_icc().
`missing`	Missing-value policy passed to `compute_facet_icc()` when `icc_table` is `NULL`. A supplied table retains its original policy.

Value

A data.frame of class `mfrm_facet_design_effect` with columns `Facet`, `AvgClusterSize`, `ICC`, `DesignEffect`, `EffectiveN`, `InputRows`, `UsedRows`, and `ExcludedRows`. Its `data_usage` attribute is retained from the ICC result. Cluster sizes and effective sample sizes use only rows included in that model.

Interpreting output

The formula describes the variance inflation of an unweighted mean under a single clustering factor with equal cluster sizes, independent clusters, and common within-cluster correlation. This helper substitutes the average cluster size and one fitted facet variance share. It does not calculate the variance of a specified estimator under the full sampling design.

- `Deff = 1` means this approximation adds no inflation for that facet; it does not establish independence of observations or adequate precision.
- `Deff > 1` signals potential clustering influence under this approximation. `EffectiveN = UsedRows / Deff` is a descriptive equivalent row count, not a count of independent Persons or an assurance of matching precision.
- Unequal cluster sizes, crossed or nested dependencies, sampling weights, and finite-population corrections are not accounted for. Per-facet values must not be added or multiplied to obtain an overall design effect.
- Reported `ICC` is pulled from `icc_table$ICC` (the variance share); interpretation is the same as in `compute_facet_icc()`.

Typical workflow

1. Run `compute_facet_icc()` to get the variance-component shares.
2. Feed the result and the data into `compute_facet_design_effect(data, facets, icc_table = icc)`.
3. Use these values to flag facets for design review. For standard errors, sample-size planning, or comparisons of precision, use an estimator and variance calculation that represent the actual design.

References

Kish, L. (1965). *Survey Sampling*. New York: Wiley.

Park, I., & Lee, H. (2004). Design effects for the weighted mean and total estimators under complex survey sampling. *Survey Methodology, 30*(2), 183-193. https://www150.statcan.gc.ca/n1/pub/12-001-x/2004002/article/7751-eng.pdf

See Also

`compute_facet_icc()`, `analyze_hierarchical_structure()`.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("compute_facet_design_effect", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
if (requireNamespace("lme4", quietly = TRUE)) {
  icc <- compute_facet_icc(toy, facets = c("Rater", "Criterion"),
                           score = "Score", person = "Person")
  deff <- compute_facet_design_effect(toy,
                                      facets = c("Rater", "Criterion"),
```

```
                                        icc_table = icc)
  print(deff)
  # Review clustering influence; EffectiveN is a descriptive row count.
}

## End(Not run)
```

`compute_facet_icc` *Compute intra-class correlations for each facet*

Description

Fits a random-effects variance-components model `Score ~ 1 + (1 | Person) + (1 | Facet1) + (1 | Facet2) + ...` using `lme4::lmer` (in `Suggests`) and returns the proportion of observed score variance attributable to each facet. This is a descriptive summary complementary to the Rasch-metric rater separation/reliability reported elsewhere.

Usage

```
compute_facet_icc(
  data,
  facets,
  score,
  person = NULL,
  reml = TRUE,
  ci_method = c("none", "boot"),
  ci_level = 0.95,
  ci_boot_reps = 1000L,
  ci_boot_seed = NULL,
  ci_boot_parallel = c("no", "multicore", "snow"),
  ci_boot_ncpus = 1L,
  missing = c("error", "omit")
)
```

Arguments

`data`	Data frame in long format.
`facets`	Character vector of facet column names.
`score`	Name of the score column.
`person`	Optional person column. If supplied it is added as a separate random intercept so Person-level variance is partitioned out.
`reml`	Logical; whether to fit with REML. Default `TRUE`.
`ci_method`	Confidence-interval method for the ICC column. One of `"none"` (default, point estimate only) or `"boot"` (parametric percentile bootstrap via `lme4::bootMer()`). Each simulated data set is refitted and its full variance decomposition used to calculate the ICC ratios. The former `"profile"` method is no longer supported; see Updating saved intervals below.

`ci_level` Confidence level when `ci_method != "none"`; default `0.95`. Koo & Li (2016) recommend banding the CI rather than the point estimate when classifying reliability as Poor / Moderate / Good / Excellent.

`ci_boot_reps` Number of bootstrap replicates used when `ci_method = "boot"`. An integer of at least 2; default `1000`. Small counts give imprecise tail quantiles; choose enough replicates for the precision required in the application.

`ci_boot_seed` Optional integer seed for the bootstrap path (between 0 and `.Machine$integer.max`). `NULL` uses the current random-number state. Bootstrap simulation advances that state.

`ci_boot_parallel` Parallelisation strategy for the parametric-bootstrap CI path, passed through to `lme4::bootMer()`: `"no"` (default), `"multicore"` (POSIX `mclapply`), or `"snow"` (PSOCK cluster). `"multicore"` does nothing on Windows and falls back to serial; use `"snow"` there.

`ci_boot_ncpus` Number of CPUs to use for the parallel bootstrap path (ignored when `ci_boot_parallel = "no"`). A positive integer. Interactive progress is available for serial execution.

`missing` How to handle missing scores or selected grouping values: `"error"` (default) or explicit complete-case omission with `"omit"`. Numeric character/factor score labels retain their numeric values. Nonnumeric scores, infinite values, and blank grouping labels are refused.

Value

A data.frame of class `mfrm_facet_icc` with one row per variance component (including a `"Residual"` row) and columns:

- `Facet`: the grouping factor name (or `"Residual"`).
- `Variance`: unrounded variance estimate (REML by default, ML if `reml = FALSE`); `NA` when the variance shares are undefined.
- `ICC`: variance share (`Variance / sum(Variance)`), in `[0, 1]`.
- `Interpretation`: band label according to the facet's scale.
- `InterpretationScale`: `"Koo-Li reliability"` for the person facet, `"Variance share"` for others.
- `ICC_CI_Lower` / `ICC_CI_Upper` / `ICC_CI_Level` / `ICC_CI_Method`: CI bounds (unavailable bounds are `NA`), requested level, and method.
- `ICC_CI_Status`: whether intervals are available and, otherwise, why.
- `ICC_CI_NRequested` / `ICC_CI_NReps` / `ICC_CI_NUnavailable`: requested bootstrap count, number of converged refits with all ICCs finite, and number without such a result (absent for `"none"`). Counts are `NA` when the bootstrap aborts without returning its draws. Warnings can withhold intervals even when all draws are finite and all refits converge.

The `icc_ci` attribute retains the original fit's convergence and singularity diagnostics and warnings. When bootstrap results are returned, its `bootstrap` entry contains every draw, per-refit convergence and singularity indicators, the number of refit errors, and lme4's message/warning/error tables.

Rows used

Missingness is checked only in the score, facets, and optional person column. `InputRows`, `UsedRows`, and `ExcludedRows` are included in the table. `attr(x, "data_usage")` retains these counts, excluded input row positions, missing columns per row, and observed grouping-level counts after omission. `compute_facet_design_effect()` uses these retained counts for its sample sizes. Omission does not impute scores or correct missing-data bias.

Score units and zero variation

A small positive variance is not treated as zero using a fixed cutoff. Multiplying scores by a nonzero constant leaves the variance shares unchanged, up to fitting precision, while variances change by its square. Variance estimates are retained without decimal rounding. If all retained scores are equal, or the fitted total variance is not positive and finite, variances and ICCs are unavailable (`NA`); numerical fitting residue is not interpreted as observed variation. A constant-response bootstrap refit is also unavailable and withholds the interval.

Interpreting output

The `Interpretation` column uses **two scales** so the same numeric ICC reads correctly for each facet role:

- For the `person` facet, higher ICC = better. Koo & Li (2016, p. 161) bands are applied: `< 0.5` Poor, `[0.5, 0.75]` Moderate, `(0.75, 0.9]` Good, `> 0.9` Excellent. The strict `>` boundary at 0.9 follows Koo & Li's wording "values greater than 0.90 indicate excellent reliability" (so an ICC of exactly 0.9 reads as Good).
- For non-person facets (Rater, Criterion, Task, Region, ...) the same numeric value is a **variance share**: how much of the total observed score variance sits at that facet. The bands used here are different (`Trivial share` < 0.05, `Small share` < 0.15, `Moderate share` < 0.30, `Large share` >= 0.30), and a large rater share is generally *bad* news (raters disagree about averages), not good news.

The `InterpretationScale` column explicitly records which scale applies to each row, so downstream reporting does not confuse the two. FACETS (Linacre, 2026) reports rater separation/reliability on the Rasch metric instead of an ICC; mfrmr surfaces both, with the Rasch-metric version in `diagnostics$reliability` and this variance-share view here. For agreement or consistency between specific pairs on common units, use `pairwise_icc_table()` instead. A small rater variance share here does not establish pairwise agreement; residual disagreement can still be large.

Set `ci_method = "boot"` to request intervals alongside the point estimates. The `Interpretation` column still uses point estimates. The bootstrap simulates Gaussian random effects and errors from the fitted model; it does not correct model misspecification or missing-data bias. Percentile coverage can be unreliable near zero variance components or with few grouping levels.

Intervals are withheld if the original fit has convergence problems or warnings, or if any requested bootstrap refit fails, has convergence problems or warnings, or produces an undefined ICC. Finite draws are retained but never silently selected to calculate an interval. Singular fits (zero random-effect components) are recorded separately and retained when they converge; they are not automatically treated as failures. Inspect `ICC_CI_Status` and `attr(x, "icc_ci")` before reporting intervals.

Updating saved intervals

The former `ci_method = "profile"` transformed separate standard-deviation intervals while holding other variance components fixed. These are not profile-likelihood intervals for the ICC ratio and should not be reported as ICC confidence intervals. Requests now stop with an explanation. Rerun `compute_facet_icc()` or `analyze_hierarchical_structure()` with the original data and settings, choosing `ci_method = "boot"` explicitly if intervals are needed. Saved bootstrap results from earlier versions must also be rerun to obtain complete failure accounting. Printing, summarizing, or plotting old interval results cannot correct their calculations.

Typical workflow

1. Fit the MFRM model with `fit_mfrm()` for the Rasch-metric separation/reliability.
2. Call `compute_facet_icc(data, facets, score, person)` to get the complementary variance-share summary.
3. Feed into `compute_facet_design_effect()` to convert ICCs and average cluster sizes into descriptive, per-facet design-effect approximations. These do not estimate the precision of the full design.

References

Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine, 15*(2), 155-163.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1-48.

See Also

`compute_facet_design_effect()`, `pairwise_icc_table()`, `analyze_hierarchical_structure()`, `detect_facet_nesting()`, `facet_small_sample_review()`.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("compute_facet_icc", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
if (requireNamespace("lme4", quietly = TRUE)) {
  icc <- compute_facet_icc(toy, facets = c("Rater", "Criterion"),
                           score = "Score", person = "Person")
  print(icc)
  # Look for:
  # - Person ICC reads as Koo & Li (2016) reliability: < 0.5 poor,
  #   0.5-0.75 moderate, 0.75-0.9 good, > 0.9 excellent.
  # - Rater / Criterion ICC reads as variance share, NOT reliability;
  #   small values indicate limited systematic facet differences, and
  #   shares > 0.10 hint at meaningful systematic facet differences.
  # - `Interpretation` summarises the variance-share band the helper
  #   has assigned to each row.
}
```

```
## End(Not run)
```

`compute_information` *Compute design-weighted precision curves for ordered many-facet fits*

Description

Calculates design-weighted score-variance curves across the latent trait (theta) for a fitted ordered-category `RSM`, `PCM`, or `GPCM` model. Returns both an overall precision curve (`$tif`) and per-facet-level contribution curves (`$iif`) based on the realized observation pattern.

Usage

```
compute_information(fit, theta_range = c(-6, 6), theta_points = 201L)

## S3 method for class 'mfrm_information'
plot(
  x,
  type = c("tif", "iif", "se", "sem", "csem", "both"),
  facet = NULL,
  draw = TRUE,
  ...
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`theta_range`	Numeric vector of length 2 giving the range of theta values. Default `c(-6, 6)`.
`theta_points`	Integer number of points at which to evaluate information. Default `201`.
`x`	Output from `compute_information()`.
`type`	`"tif"` for the overall precision curve (default), `"iif"` for facet-level contribution curves, `"se"` / `"sem"` / `"csem"` for the conditional standard error of measurement implied by that curve, or `"both"` for precision with conditional SEM on a secondary axis.
`facet`	For `type = "iif"`, which facet to plot. If `NULL`, the first facet is used.
`draw`	If `TRUE` (default), draw the plot. If `FALSE`, return reusable `mfrm_plot_data` invisibly.
`...`	Additional base-graphics arguments for the selected view.

Details

For RSM / PCM, the score variance at theta for one observed design cell is:

$$I(\theta) = \sum_{k=0}^{K} P_k(\theta)\,(k - E(\theta))^2$$

where P_k is the category probability and $E(\theta)$ is the expected score at theta. In mfrmr, these cell-level variances are then aggregated with weights taken from the realized observation counts in fit$prep$data.

The resulting total curve is therefore a design-weighted precision screen rather than a pure textbook test-information function for an abstract fixed item set. The associated standard error summary is still $SE(\theta) = 1/\sqrt{I(\theta)}$ for positive information values.

In an ordered Rasch-family model, category discrimination is fixed at 1, so this score-variance representation is the natural conditional information identity rather than a separate approximation. For binary data it reduces to the familiar $p(\theta)\{1 - p(\theta)\}$ form. For PCM, the package evaluates each observed design cell using the threshold vector associated with that cell's realized step_facet level. For GPCM, the same design-weighted score variance is scaled by the squared discrimination attached to the realized slope_facet level, which is the $a_j^2 \cdot \mathrm{Var}(T \mid \theta)$ item-information identity that Muraki (1993, Equation 10) derives by applying Samejima's (1974) polytomous information formula to the GPCM kernel of Muraki (1992).

plot() method: Plot design-weighted information or conditional precision

Plot a saved compute_information() result to see where the observed assessment design provides more or less precision across ability.

Value

An object of class mfrm_information (named list) with:

- tif: tibble with columns Theta, Information, SE. The Information column stores the design-weighted precision value.
- iif: tibble with columns Theta, Facet, Level, Information, and Exposure. Here too, Information stores a design-weighted contribution value retained under that column name for compatibility.
- theta_range: the evaluated theta range.

For plot(): Invisibly, an mfrm_plot_data object.

What tif and iif mean here

In mfrmr, this helper supports ordered-category RSM, PCM, and the current GPCM fit. The total curve ($tif) is the sum of design-weighted cell contributions across all non-person facet levels in the fitted model. The facet-level contribution curves ($iif) keep those weighted contributions separated, so you can see which observed rater levels, criteria, or other facet levels are driving precision at different parts of the scale. For PCM, step-facet-specific thresholds are respected when each observed design cell is evaluated. For GPCM, those same cell-level variances are additionally scaled by the squared discrimination associated with the realized slope_facet level.

What this quantity does not justify

- It is not a textbook many-facet test-information function for an abstract fixed item set.
- It should not be used as if it were design-free evidence about a form's precision independent of the realized observation pattern.
- It does not currently extend beyond the ordered-category `RSM` / `PCM` / `GPCM` family implemented by `fit_mfrm()`.

When to use this

Use `compute_information()` when you want a design-weighted precision screen for an `RSM`, `PCM`, or `GPCM` fit along the latent continuum. In practice:

- start with the total precision curve for overall targeting across the realized observation pattern
- inspect facet-level contribution curves when you want to see which raters, criteria, or other facet levels account for more of that design-weighted precision
- widen `theta_range` if you expect extreme measures and want to inspect the tails explicitly

Choosing the theta grid

The defaults (`theta_range = c(-6, 6)`, `theta_points = 201`) work well for routine inspection. Expand the range if person or facet measures extend into the tails, and increase `theta_points` only when you need a smoother grid for reporting or custom graphics.

References

The ordered-category probability structures come from Andrich's `RSM` formulation and Masters' `PCM`. The `GPCM` information identity $a_j^2 \cdot \mathrm{Var}(T \mid \theta)$ is derived in Muraki (1993, Equation 10) by applying Samejima's (1974) general polytomous information formula $I_j(\theta) = \sum_k P_{jk}(\theta)[-\partial^2 \ln P_{jk}/\partial\theta^2]$ to the GPCM probability kernel of Muraki (1992). For the integer scoring function $T_k = k$ used by `mfrmr`, this reduces to $a_j^2 \cdot \mathrm{Var}(K \mid \theta)$. In `mfrmr`, those formulas are applied to the realized many-facet observation design, so the output should be read as a design-weighted precision summary rather than as a design-free abstract test function.

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176. doi:10.1177/014662169201600206 (See Equations 6, 10, and 13 for the probability kernel and the $\partial P_k/\partial\theta = a_j P_k(k - E[K])$ derivative used by all GPCM helpers in `mfrmr`.)
- Muraki, E. (1993). *Information functions of the generalized partial credit model*. Applied Psychological Measurement, 17(4), 351-363. doi:10.1177/014662169301700403 (Equation 10 derives the item information function for the GPCM, $I_j(\theta) = D^2 a_j^2 \mathrm{Var}(T \mid \theta)$, by applying Samejima's (1974) polytomous information formula to the GPCM kernel; this is the canonical reference for `compute_information()` under `GPCM`.)

- Samejima, F. (1974). *Normal ogive model on the continuous response level in the multidimensional latent space*. Psychometrika, 39, 111-121. (Source for the general polytomous information formula that Muraki 1993 specializes to the GPCM.)

Interpreting output

Category variance is computed from squared deviations about its mean to retain small positive information near an endpoint category. This avoids cancellation in raw second moments; it does not eliminate floating-point underflow or account for calibration uncertainty.

- `$tif`: design-weighted precision curve data with theta, Information, and SE.
- `$iif`: design-weighted facet-level contribution curves for the fitted non-person facets.
- Higher information implies more precise measurement at that theta.
- SE is inversely related to information.
- Peaks in the total curve show the trait region where the realized calibration is most informative.
- Facet-level curves help explain *which observed facet levels* contribute to those peaks; they are not standalone item-information curves and should be read as design contributions.
- The total curve peaks where the realized design is most precise.
- Conditional SEM is derived as `1 / sqrt(precision)`; lower is better.
- Facet-level curves show which facet levels contribute most to that realized precision at each theta.
- For `GPCM`, those contributions include the squared discrimination scaling implied by the fitted `slope_facet`.
- If the precision peak sits far from the bulk of person measures, the realized design may be poorly targeted.

How to read the main columns

- `Theta`: point on the latent continuum where the curve is evaluated.
- `Information`: design-weighted precision value at that theta.
- `SE`: approximate `1 / sqrt(Information)` summary for positive values.
- `Exposure`: total realized observation weight contributing to a facet-level curve in `$iif`.

Recommended next step

Compare the precision peak with person/facet locations from a Wright map or related diagnostics. If you need to decide how strongly SE/CI language can be used in reporting, follow with `precision_review_report()`.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Run `compute_information(fit)`.
3. Plot with `plot(info, type = "tif")`.
4. If needed, inspect facet contributions with `plot(info, type = "iif", facet = "Rater")`.

1. Compute information with compute_information().
2. Plot with plot(info) for the total precision curve.
3. Use plot(info, type = "iif", facet = "Rater") for facet-level contributions.
4. Use draw = FALSE when you want reusable plot data for custom graphics or reporting helpers.

Plot types

- "tif": overall design-weighted precision across theta.
- "se" / "sem" / "csem": conditional SEM across theta.
- "both": precision and conditional SEM together, useful for presentations.
- "iif": facet-level contribution curves for one selected facet in a supported RSM, PCM, or GPCM fit.

Which type should I use?

- Use "tif" for a quick overall read on precision.
- Use "sem" or "csem" when standard-error language is easier to communicate than precision.
- Use "both" when you want both views in one figure.
- Use "iif" when you want to see which facet levels are shaping the total precision curve.

Returned data when draw = FALSE

draw = FALSE returns an mfrm_plot_data object. The underlying plotting data are stored in $data$plot. For type = "tif", "se", or "both", those rows come from x$tif. For type = "iif", the returned rows come from x$iif filtered to the requested facet. The plot data also include plot_long, information_long, conditional_sem, summary, and settings so ggplot2, plotly, Quarto, and table workflows can reuse the information and conditional-SEM series without parsing the drawn figure.

See Also

fit_mfrm(), plot(x)

compute_information(), mfrmr_plot_migration

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("compute_information", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM",
                quad_points = 7, maxit = 30)
info <- compute_information(fit)
head(info$tif)
info$tif$Theta[which.max(info$tif$Information)]
```

```
## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_information", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM",
                quad_points = 7, maxit = 30)
info <- compute_information(fit)
tif_data <- plot(info, type = "tif", draw = FALSE)
head(tif_data$data$plot)
iif_data <- plot(info, type = "iif", facet = "Rater", draw = FALSE)
head(iif_data$data$plot)

## End(Not run)
```

`compute_person_fit_indices` *Person fit indices: lz and Snijders-corrected lz**

Description

Computes person-level fit statistics for an MFRM bundle, extending the Infit / Outfit / ZSTD columns that `diagnose_mfrm()$measures` already exposes with the standardized log-likelihood `lz` and, when justified by the person-estimation method, Snijders' `lz*`.

Usage

```
compute_person_fit_indices(diagnostics, fit = NULL)

## S3 method for class 'mfrm_person_fit_indices'
summary(object, digits = 3, top_n = 10, include_person = FALSE, ...)
```

Arguments

`diagnostics`	Output from `diagnose_mfrm()`.
`fit`	Optional `mfrm_fit` from `fit_mfrm()`. Required to decide whether the person estimates are JML/fixed-effect estimates for which the Snijders (2001) correction is computed. MML/EAP person scores return `NA` for `lz_star` with an explanatory status. Supplied fit and diagnostics must identify the same analysis.
`object`	Output from `compute_person_fit_indices()`.
`digits`	Number of digits used when printing numeric columns.
`top_n`	Number of review rows retained in `top_review`.
`include_person`	If `TRUE`, print person identifiers in the review preview. The default retains the rows in the returned object but suppresses identifiers from console output.
`...`	Unused.

Details

summary() method: Summarize person-fit indices

summary() method: summary() for compute_person_fit_indices() output gives a compact, report-ready reading order: first the number of persons and flags, then status counts, then the highest-priority review rows. The summary keeps lz_star availability visible so users do not silently treat uncorrected lz as Snijders-corrected output.

Value

A data frame of class mfrm_person_fit_indices with one row per Person and columns:

Person Person ID.

TotalN, N, UnavailableN Total responses, responses with valid probability and log-likelihood moments, and unavailable responses. An incomplete response set does not produce a person-fit statistic.

LogLik Sum of log P(X = x | theta) under the fitted model. Computed from the per-observation category probability PrObserved (the model probability of the observed category), not from a Gaussian residual approximation.

lz Drasgow et al. (1985) standardized log-likelihood, in its polytomous form, treating each retained response once regardless of calibration weights. lz_status records computational availability.

lz_star Snijders-corrected lz* when the source fit used JML/fixed-effect person estimates, conditioning on the fitted non-person calibration, and the diagnostics include the required derivative terms. Numerical convergence, a finite person estimate and unit observation weights are required; otherwise NA.

lz_star_status Status string for lz_star, such as "computed_jml_conditional_calibration", "fit_required", "not_applicable_eap", or "insufficient_information".

lz_star_c Estimated Snijders projection coefficient c_n for each person, when available.

lz_star_variance Corrected variance denominator used for lz_star, when available.

lz_flag_5pct, lz_flag_1pct Logical flags for practical two-sided lz thresholds of |z| > 1.96 and |z| > 2.58.

lz_star_flag_5pct, lz_star_flag_1pct The same flags for lz_star. All index flags, including ReportFlag, remain NA when the corresponding statistic is unavailable.

ReportIndex, ReportValue, ReportFlagLevel, ReportFlag, ReviewStatus, ReviewReason, ReportCaveat Compact reporting columns. ReportIndex prefers lz_star when the Snijders correction was computed; otherwise it falls back to lz with an explicit caveat.

The 5pct and 1pct flag names refer to standard-normal reference cutoffs: |lz| > 1.96 and |lz| > 2.58. Treat them as screening thresholds, not a guarantee of those false-positive rates for fitted person scores. Read ReportCaveat and lz_star_status with every flagged result. When lz_star_status == "computed_jml_conditional_calibration", lz_star applies Snijders' estimated-ability correction for JML person estimates, conditional on the fitted non-person parameters. This does not propagate non-person calibration uncertainty. For MML/EAP person scores, use lz with its documented caveat rather than treating EAP scores as if they satisfied the Snijders estimating equation. Positive and negative flags indicate more and less predictable patterns, respectively; neither alone establishes invalid responses or justifies exclusion. The implemented correction does not

cover non-unit observation weights, even if normalized to mean one. No multiple-person error-rate control is provided.

Small positive probabilities are used without a lower floor. Zero, invalid or missing observed probabilities make the person statistic unavailable. Missing derivative information withholds the correction for the whole person. With the matching fit supplied, older probability-moment diagnostics are refreshed from that fit without re-estimation. Otherwise recompute diagnostics first. Older saved person-fit tables/summaries must be rebuilt.

Note: this implementation reads the model category probabilities directly from the diagnostics bundle. Earlier mfrmr releases used a Gaussian-residual approximation $\log P(X = x) \approx -\frac{1}{2}(R^2/V) - \frac{1}{2}\log(2\pi V)$ as a stand-in for $\log P$, which overstated the per-item variance of $\log P$ for polytomous items, shrinking the reported `lz` toward zero. Numerical `lz` values are therefore not directly comparable across mfrmr releases; treat the values returned here as the polytomous statistic and re-evaluate any historical `|lz| > 1.96` flagging that was based on the earlier approximation.

For `summary()`: An object of class `summary.mfrm_person_fit_indices` with:

- `overview`
- `status_summary`
- `report_index_summary`
- `lz_star_status_summary`
- `top_review`
- `caveats`
- `thresholds`
- `reporting_map`
- `notes`

References

- Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*(1), 67-86.
- Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika, 66*(3), 331-342.
- Magis, D., Raiche, G., & Beland, S. (2012). A didactic presentation of Snijders's lz* index of person fit with emphasis on response model selection and ability estimation. *Journal of Educational and Behavioral Statistics, 37*(1), 57-81.
- Sinharay, S. (2016). Asymptotically correct standardization of person-fit statistics beyond dichotomous items. *Psychometrika, 81*(4), 992-1013.

See Also

`diagnose_mfrm()`

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Use the diagnostics saved with this fit
diagnostics <- example_fit$diagnostics

# Screen for unusual person response patterns
person_fit <- compute_person_fit_indices(diagnostics, fit = fit)
head(person_fit[, c("Person", "N", "ReportIndex", "ReportValue", "ReviewStatus")])

# Inspect flagged persons and the limitations of their reported index
subset(person_fit, ReportFlag,
       c("Person", "ReportValue", "ReviewReason", "ReportCaveat"))
# With MML/EAP scores, lz_star is unavailable; lz is an uncorrected screening index
# A flag alone does not establish invalid responses or justify excluding a person
```

---

confint.mfrm_fair_average

Intervals for common-plan Person expected scores

Description

Fixed-Person parametric refit intervals or a local delta approximation for a declared common rating plan. Both preserve the calibration-to-anchor-to-scoring estimator and account for explicitly shared response events. A separate conservative bounded-response method uses the scoring observations directly.

Usage

```
## S3 method for class 'mfrm_fair_average'
confint(
  object,
  parm = NULL,
  level = 0.95,
  nsim,
  seed,
  contrasts = NULL,
  anchor_source = NULL,
  generating_fit = NULL,
  response_map = NULL,
  anchor_uncertainty = c("estimated", "fixed"),
  gradient_tol = 1e-04,
  method = c("refit", "delta", "bounded"),
```

```
  generator_method = c("ordinary", "jeffreys"),
  mc_level = 0.95,
  ...
)

## S3 method for class 'mfrm_fair_intervals'
summary(object, ...)

## S3 method for class 'mfrm_fair_intervals'
print(x, ...)

## S3 method for class 'mfrm_fair_intervals'
plot(x, draw = TRUE, show_title = FALSE, show_notes = FALSE, ...)
```

Arguments

object	For confint(), output from fair_average_table() with reference_plan. For summary(), a saved mfrm_fair_intervals result.
parm	Character Person identifiers. Default selects all Persons.
level	Pointwise confidence level in (0, 1).
nsim	For method = "refit", required integer number of replications, at least two. Small values are useful for checking the procedure, not precise tail estimation. Omit for method = "delta" or "bounded".
seed	For method = "refit", required nonnegative integer seed. The caller's RNG state is restored and the RNG kind is recorded. Omit for method = "delta" or "bounded", which do not simulate or change the RNG state.
contrasts	Optional numeric matrix with named rows and columns equal to parm (order may differ). Rows must sum to zero and define differences of common-plan expected scores. Without it, return individual scores.
anchor_source	Native ordinary or explicit Jeffreys JML fit that supplied every direct anchor in object. Values must match the unrounded primary estimates. Its own anchors are not supported in this initial two-stage route. Omit calibration/generator inputs, response_map, anchor_uncertainty and gradient_tol for method = "bounded": its bounds do not use fitted anchors or an estimated generating model.
generating_fit	Fit on the union of calibration and scoring response events, supplying the fitted response model. Defaults to anchor_source for estimated anchors and the scoring fit otherwise. A separate union fit is needed when scoring includes events outside the calibration data.
response_map	For estimated anchors, a required list with numeric vectors calibration and scoring. Each maps rows of that fit's prep$data to rows of generating_fit$prep$data. Indices identify actual response events, not just equal Person/facet labels. Shared indices mean the same observation; distinct occasions require distinct generating rows. Identifiers and observed scores must agree, and the maps must cover the union. Providing these maps asserts the event correspondence.

anchor_uncertainty
: "estimated" (default) requires anchor_source when direct anchors are present. "fixed" explicitly conditions on them. It does not propagate estimated-anchor uncertainty.

gradient_tol
: Positive tolerance for the maximum absolute estimating gradient (ordinary or penalized according to the fitted estimator), default 1e-4, applied to sources and every refit. Optimizer convergence, identification and boundary checks are also required.

method
: "refit" (default) for empirical basic limits from whole-pipeline refits; "delta" for pointwise normal-reference limits using the local derivative of the same estimator. These two methods are experimental. "bounded" gives conservative pointwise Hoeffding bounds for the same true expected-score target from independent bounded scoring responses, retaining the full score range for unobserved reference-plan cells.

generator_method
: For method = "refit" only, "ordinary" (default) uses the supplied ordinary-JML generating estimates. Explicit "jeffreys" re-estimates that generating model with a canonical Jeffreys information penalty. It retains ordinary JML for the original point and every refit. This experimental option can provide a finite generator when Person MLEs are infinite; it does not qualify coverage or correct the reported points. For a score result from fit_mfrm(jml_penalty = "jeffreys"), the default is instead "jeffreys": use that already fitted generating model, and retain Jeffreys estimation for original points and every refit. Explicit "ordinary" is refused for these results.

mc_level
: For method = "refit", confidence level in (0, 1) for Monte Carlo enclosures of both interval endpoints together, per target. Default .95. This measures finite-replication error, separately from level, and does not qualify sampling coverage. Omit for other methods.

...
: Must be empty.

x
: For print.mfrm_fair_intervals(): A saved mfrm_fair_intervals result.
For plot(): Saved output of confint.mfrm_fair_average().

draw
: Logical; draw with base graphics when true.

show_title, show_notes
: Logical; display title/cautions, off by default.

Details

For refit and delta intervals, each Person is fixed at the generating fit's primary estimate. Responses are locally independent under its RSM/PCM model, with the retained assignment and missingness pattern. The procedure does not draw a new Person population or resample rows. Calibration and scoring must share the same model, category coding, facet signs, estimator and parameter meanings. Only unit-weight additive RSM/PCM JML without shrinkage or group anchors is supported. Score recoding must be the identity; transformed category labels are not accepted here. Ordinary points and Jeffreys points cannot be mixed across stages. With explicit Jeffreys fits, method = "refit" uses the same penalty for original calibration, anchor extraction, scoring and every replication. Numerical checks evaluate the penalized equations; ordinary likelihood recession does not turn these

penalized points into infinities. Rank deficiency and numerical failures still stop or remain unresolved in the planned denominator. No failed replicate is redrawn or replaced by an ordinary estimate. `method = "delta"` uses the analytic penalized score Jacobian, including the penalty's dependence on estimated anchors. It does not use ordinary information or inverse penalized curvature as covariance. `method = "bounded"` retains its separate response-based assumptions and fitted-point display. The penalty changes the points; it does not qualify score bias or coverage.

With `method = "refit"`, the original two-stage estimator is preserved: first refit calibration, extract the specified anchors, then refit the scoring data under those anchors and compute scores on the saved common plan. The generator is a plug-in response model, not a replacement joint estimator for the data. The bootstrap error is replicate score minus the common-plan score under the **generating** model; this need not equal the original two-stage estimate. Basic limits subtract its empirical quantiles from the original estimate. Contrasts use paired scores from each whole replication, retaining covariance.

Unresolved refits remain in the planned denominator. Replacing their unknown errors by minus/plus infinity gives outer empirical-quantile bounds; the function does not silently condition on successful refits. Nonfinite bounds are returned as missing with `Status = "unresolved_tail"`. Finite bounds outside the score range are retained, not clipped to disguise approximation error. Type-1 empirical quantiles and expected observations per tail are recorded. These are pointwise intervals, without multiplicity adjustment. Inspect `LowerMCLower`, `LowerMCUpper`, `UpperMCLower` and `UpperMCUpper` for Monte Carlo enclosures of the lower and upper basic endpoints. Unlike the reported empirical interval, these enclosures retain infinite limits when replication counts or unresolved refits cannot bound an endpoint. `MCFinite` means all four limits are finite, not sufficiently precise. The `monte_carlo` attribute retains quantile probabilities and order ranks. For each error quantile p, the ranks are `qbinom(a, nsim, p)` and `qbinom(1-a, nsim, p)+1`, where `a = (1-mc_level)/4`. Ranks zero and `nsim+1` denote infinities. The four one-sided allocations jointly cover both infinite-replication basic endpoints for one target at least at `mc_level`, conditional on a fixed generating model and independent draws. They are conservative for discrete errors/ties. There is no simultaneous guarantee across targets, reruns or adaptively chosen stopping times. Unknown errors retain their minus/plus-infinity envelopes; the enclosures include every completion of failed draws. If the estimator is not defined on some outcomes, this does not create a sampling distribution for it. Increasing `nsim` cannot repair structural nonidentification or qualify bias. With finite qualified structural calibration, a replicate Person at minus or plus infinity has a well-defined limiting expected score (the scale endpoint). That limit is retained and counted in `trials$BoundaryPersons`, without using a finite optimizer proxy. An infinite estimated anchor cannot be passed into the next fitting stage and remains unresolved. The generating fit and originally selected Persons must have finite primary estimates for the default ordinary generator. The explicit Jeffreys option below also accepts original selected limiting expected scores under finite structural calibration; its own generating parameters must be finite and qualified.

With `method = "delta"`, the derivative is evaluated at the observed calibration and scoring fits. Each stage's information uses that stage's fitted category probabilities; the variance and cross-covariance of response events use the declared generating fit. Shared events contribute to both stages before their categorical covariance is taken. Distinct occasions with identical labels are independent under this fixed-Person model unless their event indices are shared. The target covariance retains cross-Person and cross-stage terms, which can increase or decrease a contrast's variance. A direct anchor changes both its own location and any dependent centering coordinate; the derivative includes that affine change. This is model-based propagation, not an empirical robust/cluster sandwich estimate.

Delta propagation requires finite primary Person estimates throughout the generating, calibration and scoring fits and numerically positive-definite score Jacobians. Selected-target solves use diagonal scaling without regularization. For ordinary JML no full parameter inverse is formed. The Jeffreys derivative additionally solves for centered category designs and processes one derivative direction at a time, without event-squared or parameter-cubed arrays. Its scaled Jacobian reciprocal condition must exceed `sqrt(.Machine$double.eps)`; numerical availability can fail in weak designs. Information, conditioning or boundary failures preserve the point estimates, return missing SEs/limits and a `Reason`, and set `Status = "unavailable_information"`. Normal limits are not clipped to the score support; large widths or impossible endpoints can expose a poor local approximation. No simulations, resolved-replication counts or tail counts are implied by this method (those count columns are `NA`). Even an unselected extreme Person in the generating fit can make uncertainty unavailable for otherwise finite selected targets; their point estimates and the reason are retained. Do not remove that Person merely to obtain intervals. Delta limits are centered on the observed estimate and do not perform a bootstrap bias correction or correct a discrepancy between stage models.

Refit and delta are experimental fitted-model approximations. Neither removes JML incidental-parameter bias, certify nominal coverage, include uncertainty in the chosen plan, predict a future observed score, or establish fairness. Model misspecification, dependence beyond the model and changing Persons or calibrations across occasions require different procedures. `CIEligible` remains false pending finite-sample qualification; `Available` describes numerical availability only. No external-software coverage guarantee is implied. Correct local covariance does not guarantee adequate normal limits. In the supported binary special case with six independent responses, the Jeffreys expected-score estimate is $(S + 0.5)/7$. At true success probability 0.5, the plug-in delta 95% interval covers only for $S = 2, 3, 4$: exact coverage is 0.78125, despite an unbiased point at that probability. At probability 0.25, exact coverage is 0.784423828125. These are exhaustive binomial calculations, not Monte Carlo estimates. They also hold when the estimated Person anchor is passed to fully fixed scoring on the same responses. The normal-reference approximation, including plug-in standard errors and discrete outcomes, therefore has a demonstrated small-sample limitation. More Persons do not necessarily increase the responses for each focal Person. The basic refit route is not a general remedy: exact, infinite-replication binomial refit quantiles for the same estimator give coverage 0.78125 at probability 0.5 and 0.452709 at probability 0.1. For an all-zero sample, its upper basic limit equals the fitted score $0.5/7$; the bootstrap estimator cannot fall below its own score floor. This is an interval-method limitation, not inner Monte Carlo error, and more refits do not remove it.

`plot()` method: Plot common-plan expected-score intervals

For `plot()`: Points are common-plan expected scores or declared paired differences. Bars show only available pointwise bounds under the saved method. Missing bars are not zero uncertainty. Plotting never recalculates the fit or removes unresolved rows from the saved result. For refit results the bars show the empirical interval; endpoint Monte Carlo enclosures are retained in the returned table and `data$monte_carlo`, not drawn as sampling intervals. See `confint.mfrm_fair_average()` for the estimand, assumptions and finite-sample limitations.

Value

An `mfrm_fair_intervals` data frame with estimates, numerical bounds, availability and interpretation columns. Refit attributes retain `draws`, `person_draws`, `anchor_draws`, `trials`, `settings`, `reference_plan`, `generating_target`, `response_map`, `source_checks` and source identities. Refit results also retain `MCLevel`, four endpoint-enclosure columns, `MCFinite` and the `monte_carlo`

rank table, without changing `Available` or promoting `CIEligible` on the strength of numerical precision. The explicit Jeffreys route additionally retains `generating_model`, with free parameters, probabilities, numerical checks and optimization controls. Delta results additionally have `SE` and `Reason` columns and attributes `covariance` (ordered by Target), `covariance_components` and `linearization` (target/anchor derivatives, selected solve loadings, parameter coordinates and solve residuals). The source/plan/settings attributes are shared by both methods. Unavailable delta covariance is `NA`, not zero. Bounded results instead retain `Informative`, `DesignEstimate`, `ObservedContribution`, `SamplingRadius`, unobserved-cell and target-support limits, a per-Person `support` attribute and sparse `person_event_weights`. Their `settings$score_support` retains the declared limits and coding basis. Their SE and simulation-count columns are missing; the radius is not an SE. `summary()` reports the same table without recalculation; `as.data.frame()` supports export and `saveRDS()` preserves the full evidence. See `plot.mfrm_fair_intervals()` for display and interpretation.

For `plot()`: Invisibly an `mfrm_plot_data` object. Its `data$data` retains all result rows and uncertainty labels; `data$notes` retains interpretation.

Explicit Jeffreys generating model

`generator_method = "jeffreys"` minimizes $-\ell(b) - \frac{1}{2}\log|I(b)|$ in the generating fit's actual affine free coordinates, where I is the ordinary categorical model information. Direct anchor values enter as fixed offsets in this generating model. The calibrated anchors used by the scoring estimator are still re-estimated on every replication, with shared response events generated only once. This option requires declared rubric endpoints and unchanged full score coding; it is unavailable for delta and bounded-response intervals.

Computation starts at the free-coordinate origin and uses BFGS, the generating fit's `maxit`, and the stricter of its `reltol` and `1e-14`. The analytic penalized gradient must pass `gradient_tol`. Information must be numerically positive definite; scaled sparse solves must pass a `1e-8` residual check. There is no ridge, finite parameter box or probability clipping. Optimization, rank or numerically noninterior probability failures stop with a reason; success is not a certificate of a global optimum. Ordinary source fits and every refit retain their existing stationarity and structural checks.

The original ordinary-JML point and its estimator are unchanged. Basic limits use replicate ordinary-JML errors relative to the Jeffreys generating expected score, not the original fitted generating score. Finite expected-score limits for original extreme Persons are retained; unresolved structural calibration or an infinite anchor passed between stages remains unsupported. Inspect `generating_model` and `generating_target` attributes for the distinct generating parameters, probabilities, target and numerical diagnostics.

Firth's first-order bias reduction concerns canonical parameters in regular fixed-dimensional asymptotics. It does not establish bias reduction for nonlinear expected scores, fixed-short-exposure JML as Person count increases, or this two-stage interval procedure. For example, with $S \sim Bin(n, p)$, the induced estimate $(S + 1/2)/(n + 1)$ has bias $(1/2 - p)/(n + 1)$, whereas S/n is unbiased for p. Thus a finite generator can worsen a score bias. Sparse/weak-link coverage and the bias/variance tradeoff require separate qualification; `CIEligible` remains false. This is distinct from FACETS Xtreme and from the corrected-GPCM `jml_correction_order` estimator.

Conservative bounded-response intervals

`method = "bounded"` targets the same true common-plan expected score or contrast, using the original scoring rows. It does not propagate a fitted covariance or generate responses from extreme Per-

son estimates. A finite-sample probability inequality applies under independent responses, a fixed assignment, plan and contrast chosen independently of the observed scores, known score bounds, and stable expected scores within each Person-by-plan cell. Repeated rows must represent independent observations, not copied ratings. Dependence, adaptive allocation and response-dependent missingness require other methods. Both `rating_min` and `rating_max` must have been explicitly declared from the rubric when fitting, with an unchanged full `score_map`. Use `category_policy = "preserve"` to retain that scoring rule. Limits inferred from observed minima/maxima, missing range provenance and internally collapsed categories are rejected: a small sample can omit a valid category, and its observed range is not known population support. Declaring the sample extrema after seeing the data does not satisfy this assumption. If a retained empty category prevents fitting, that failure remains in an end-to-end denominator; do not collapse the category merely to obtain an interval.

Each observed cell contributes its sample mean times its normalized plan weight; cells outside the plan receive zero weight. An unobserved cell retains the full declared score range. For a contrast, signed event weights preserve its definition. Bounds are intersected with the target's mathematically known range. This preserves the coverage inequality and is not clipping a normal approximation. Sparse plans can produce very wide or uninformative bounds; the full range has `Informative = FALSE`, even though it is a valid bound. `Informative = TRUE` means only narrower than the full target range, not necessarily precise enough for a substantive decision.

`Estimate` retains the original JML score, including an endpoint limit or missing estimate. The interval need not be centered on or contain that point. `DesignEstimate` is a separate unbiased weighted-mean estimate when every relevant Person has all reference-plan cells observed; otherwise it is missing. It is not a corrected JML estimate. The confidence statement concerns the true expected score, not the accuracy or bias of the retained JML point.

Estimated anchors, even from the same sample, affect the retained JML point but are not inputs to these bounds. Their uncertainty is therefore neither conditioned away nor approximated by a fixed-anchor SE. The price is foregoing model-based extrapolation to unobserved plan cells. `CIEligible = TRUE` here denotes the finite-sample inequality under the stated assumptions, not empirical qualification of JML or guaranteed usefulness. Bounds are pointwise; inspecting many Persons or selecting a contrast after seeing results is not covered. The inequality is unconditional over the prespecified response experiment. It does not imply nominal coverage conditional on successful model fitting, an extreme-score subgroup or another response-dependent selection. This API requires an existing common-plan result: if fitting cannot create that result, retain the failure in the planned denominator of an end-to-end evaluation.

If a_e is a signed event weight and scores lie in $[l, u]$, write $T = \sum_e a_e Y_e$ for the observed contribution. With $\alpha = 1 - \text{level}$, the sampling radius is

$$r = \sqrt{\tfrac{1}{2}\log(2/\alpha)\sum_e a_e^2 (u-l)^2}.$$

Each missing cell with signed plan weight b contributes between $\min(bl, bu)$ and $\max(bl, bu)$. Add these lower/upper sums to $T - r, T + r$. Hoeffding's two tails bound the probability of missing the true expected-score target by at most α, uniformly over those missing means. Extreme observed totals do not invalidate this argument.

Local two-stage derivative

Let b_c, b_s be the actual free calibration/scoring coordinates, $a(b_c)$ the extracted direct anchors, and $t(b_s, a)$ the selected expected scores or contrasts. Category-logit coefficients in the scoring stage are X_s for free coordinates and O_s for anchors, including centering changes. Write H_c, H_s for stage information, $B = \sum_e \mathrm{Cov}_s(X_s, O_s)$, $A = \partial a/\partial b_c$, and G_s, G_a for target derivatives. For explicit Jeffreys estimation replace each H by the Hessian of $-\ell - \frac{1}{2}\log|I|$ in its stage's free coordinates, and replace B by the mixed free-coordinate/anchor derivative of that same objective. This includes the anchor dependence of I; holding the penalty fixed during anchor propagation would give a different derivative. The penalty does not depend on response outcomes at fixed design/parameters, so the response-event coefficients X_s, X_c remain unchanged. Differentiating the two score equations gives event-category coefficients

$$U_s = X_s H_s^{-1} G_s^T, \qquad U_c = X_c H_c^{-1} A^T (G_a - G_s H_s^{-1} B)^T.$$

Put both coefficients on their explicit union-event indices (zero when an event is absent from a stage), then compute $V = \sum_e \mathrm{Cov}_{p_{g,e}}(U_s + U_c)$ over categories. Centering under the generating probabilities matters when stage and generator fits differ. The saved decomposition is direct scoring, propagation through calibration, and the signed shared-event cross term. Their sum is the target covariance; the cross term alone need not be positive semidefinite. This local derivation does not establish fixed-item JML asymptotics or finite-sample normality. Estimated plans and unmodelled dependence are excluded.

References

Murphy, K. M., and Topel, R. H. (1985). Estimation and inference in two-step econometric models. Journal of Business & Economic Statistics, 3(4), 370–379. doi:10.1080/07350015.1985.10509471. Related estimating-equation framework: Hardin, J. W. (2002). The robust variance estimator for two-stage models. The Stata Journal, 2(3), 253–266. doi:10.1177/1536867X0200200302. The present method uses a model-based event covariance and does not claim the robust estimator described there. Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301), 13–30, Theorem 2. doi:10.1080/01621459.1963.10500830. Firth, D. (1993). Bias reduction of maximum likelihood estimates. Biometrika, 80(1), 27–38. doi:10.1093/biomet/80.1.27. Burke, S. (2016, revised 2018). Confidence Intervals for the Median and Other Percentiles. STAT COE-Report-08-2016. Binomial order-statistic construction; the present Monte Carlo diagnostic additionally allocates errors across both endpoints and retains unresolved-draw envelopes.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("confint.mfrm_fair_average", run.dontrun = TRUE)
## Not run:
d <- load_mfrmr_data("example_core")
# Calibrate the observed ratings with ordinary joint maximum likelihood.
f <- fit_mfrm(d, "Person", c("Rater", "Criterion"), "Score",
  method = "JML", rating_min = 1, rating_max = 4, category_policy = "preserve",
  optimizer = "BFGS", maxit = 500, reltol = 1e-11)
# The bundled example's rubric is 1--4; declare it independently of sample extrema.
# Every Person is evaluated against these same Rater-Criterion cells.
plan <- unique(d[c("Rater", "Criterion")])
```

```
fair <- fair_average_table(f, reference_plan = plan)
head(fair$reference_scores) # Expected average, in internal score units.
# Two replications exercise the API only; they cannot qualify coverage.
# Explicit experimental generator; points and every refit remain ordinary JML.
ci <- confint(fair, parm = c("P001", "P002"), nsim = 2, seed = 71,
  generator_method = "jeffreys",
  contrasts = matrix(c(1, -1), 1,
    dimnames = list("P001 minus P002", c("P001", "P002"))))
summary(ci) # Read Status and Available alongside the limits.
attr(ci, "trials") # Includes unresolved refits and their reasons.
attr(ci, "monte_carlo") # Infinite enclosures expose the two-draw tail uncertainty.
plot(ci, draw = FALSE) # Saved plotting data, including the cautions.
# Deterministic local approximation to the same expected-score target:
delta <- confint(fair, parm = c("P001", "P002"), method = "delta")
delta[, c("Target", "Estimate", "SE", "Available", "CIEligible")]
attr(delta, "covariance") # Includes the covariance between Persons.
# Conservative alternative with explicit observed/unobserved plan support:
bounded <- confint(fair, parm = c("P001", "P002"), method = "bounded")
bounded[, c("Target", "Estimate", "DesignEstimate", "CI_Lower", "CI_Upper", "Informative")]
attr(bounded, "support") # Check how much of the plan was actually observed.

## End(Not run)
# Continue the confint.mfrm_fair_average() example:
# plot(ci)
# p <- plot(ci, draw = FALSE)
# p$data$notes
```

`confint.mfrm_fit` *Confidence intervals for native fit coefficients and GPCM slopes*

Description

By default, compute pointwise Wald intervals in the same free coordinates as coef() and vcov(). Explicit `parm = "slopes"` instead computes intervals for relative or standardized discriminations and slope comparisons in a native GPCM MML fit. Model/sandwich methods do not refit; an explicitly requested profile reoptimizes nuisance parameters at each candidate value. Within `parm = "slopes"`, the default is a pointwise model-based interval for each relative slope in a one-family fit. Two-family fits use their fixed standard-normal scale and the separately checked experimental approximation described below. A relative slope above one means a steeper response curve than the geometric-average slope; it is not a measure of rater agreement or a recommended scoring weight.

Usage

```
## S3 method for class 'mfrm_fit'
confint(
  object,
  parm = NULL,
  level = 0.95,
  scale = c("relative", "standardized"),
```

```
  contrasts = NULL,
  contrast_scale = c("ratio", "difference"),
  method = c("model", "sandwich", "profile"),
  clusters = NULL,
  adjust = FALSE,
  simultaneous = c("none", "bonferroni"),
  ...,
  slope = NULL,
  profile_control = list()
)

## S3 method for class 'mfrm_cml_fit'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'mfrm_comparison_fit'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	A native MML or CML result from `fit_mfrm()`. Comparison-only CML fits return intervals for their saved targets. JML structural-truth intervals are unsupported. Extended models have their own methods.
`parm`	`NULL` (default) selects all entries of `coef(object)`; exact coefficient names or positive integer positions select a distinct subset in the requested order. `"slopes"` explicitly selects the separate GPCM slope interface described below. Inspect `names(coef(object))` before selecting coefficients: their names identify free parameter coordinates, not all displayed facet levels.
`level`	Nominal confidence level strictly between zero and one.
`scale`	`"relative"` (one-family default) keeps geometric mean one. `"standardized"` reports population-SD times slope, including uncertainty in estimated SD. With covariates, this SD is the residual population SD. Two-family fits require `"standardized"` and use it when omitted: SD is fixed at one, the first slope family has geometric mean one, and the second family is free on that scale.
`contrasts`	Optional numeric matrix: one named comparison per row, all slope levels as named columns, finite nonzero rows summing to zero. For A versus B use coefficients 1 and -1, and zero for other levels.
`contrast_scale`	`"ratio"` exponentiates log-slope contrasts (A/B); `"difference"` contrasts positive slopes (A-B) by the delta method.
`method`	`"model"` uses joint observed information; `"sandwich"` uses Person marginal-likelihood scores, optionally grouped into larger clusters. Experimental `"profile"` reoptimizes nuisance parameters for one specified relative slope, or one two-family component, using a chi-square(1) likelihood-ratio cutoff.
`clusters`	Optional data frame mapping each fitted `Person` once to a nonmissing `Cluster`, as in `mfrm_facet_intervals()`. Persons are independent units by default. Shared random effects across clusters are not supported.
`adjust`	For sandwich covariance only, multiply by G/(G-1) for G clusters; default FALSE. This is not a small-sample accuracy guarantee.

`simultaneous` "none" (default) gives pointwise intervals; "bonferroni" adjusts for all rows requested in this call. It does not adjust for other analyses or post-selection of targets.

`...` Unused; additional arguments are rejected.

`slope` For `method = "profile"`, one exact level from `object$slopes$SlopeFacet`, selected for the research question. Required; profiling all levels implicitly is not supported. For two families use a named character value, e.g. `slope = c(Task = "t1")`: the name is the fitted owner column and the value is its level. This distinguishes equal level labels belonging to different facets.

`profile_control` Named list for profiling: `maxit` (400 iterations per optimizer stage), `max_steps` (8 bracketing trials per side, including contractions after failed trials), and `initial_step` (0.5 on the log-slope scale, multiplied by 1.5 during outward expansion). These limit numerical searches, not the mathematical parameter space. Numerical acceptance tolerances are not relaxed by these settings.

Details

Coefficient intervals and migration. Before this API revision, `confint(fit)` was a GPCM slope-only interface. Use `confint(fit, parm = "slopes", ...)` to retain that target, its plotting and report attachments. Omitting `parm` now selects the same coordinates as `coef(fit)`, including free step offsets, log slopes and estimated population parameters. Log-variance intervals are not SD intervals; log-slope intervals are not positive relative-slope intervals. Anchors and dependent centered levels are absent from these free coordinates. For labelled facet levels or contrasts, use `mfrm_facet_intervals()`; for population targets use `mfrm_population_intervals()`.

Coefficient intervals use $\hat{b}_j \pm z_{(1+\text{level})/2}\sqrt{V_{jj}}$. MML reuses the qualified full joint covariance from `vcov(fit)`, with the same model/design restrictions and numerical checks; it can reevaluate information but does not refit. CML reuses saved conditional information. Its comparison-only method also respects saved target eligibility, retaining unavailable targets and reasons. When only CML information fails, finite numerical points remain in the fit; the default interval matrix contains missing bounds and its `diagnostics` records the information failure. The same missing-bound contract applies to eligible ordinary fixed-population RSM/PCM MML fits whose information is unavailable, provided the `NA` covariance matrix fits the information memory budget (see mfrm_fit_extractors). Inspect `coef(fit)` and `summary(fit)` for the retained points and limitations. No interval is inferred from a finite point alone. Neither route scores Persons, supplies JML structural-truth inference, guarantees finite-sample coverage, or includes uncertainty in fixed anchors. Saved CML covariance is not a new evaluation of the likelihood or information. Inspect `attr(ci, "diagnostics")` and the saved checks/cautions before reporting.

All arguments after `level`, other than `...`, belong to `parm = "slopes"`. Supplying them without that selection is an error, so a request for a standardized slope, sandwich or profile interval cannot silently return a free-coordinate Wald interval. Dedicated extended-model, G/D-study and bootstrap-result `confint()` methods retain their documented defaults.

GPCM slope intervals (`parm = "slopes"`). Relative slopes have geometric mean one. If η denotes the G-1 free log-slope coordinates and J expands them to G sum-zero log slopes, their covariance is $JV_\eta J^T$. Here V_η is the slope block of the inverse **full joint** observed information, including estimated population parameters, rather than the inverse of the slope information block alone. Each interval is $\exp(\log(a_g) \pm zSE(\log(a_g)))$.

The current likelihood, gradient and joint information must pass the MML solution checks. The fit also needs supported score categories, consistent likelihood metadata, unit observation weights, and at least 31 quadrature points. Regularized, singular, nonconverged or mismatched results do not receive intervals. These one-family rules differ from the two-family checks below. Positive curvature is a local check, not a proof of a global maximum or accurate integration. Use `mml_quadrature_sensitivity()` and different starts for consequential results. A known deficit in the marginal response-distribution dimension or a recorded complete-pattern first-order rank failure prevents regular inference, even if numerical differentiation produces a positive Hessian. For example, two binary items answered once by every Person have only three independent response probabilities, whereas the default one-family GPCM estimates four free parameters. More Persons on that same design cannot identify this model. Returned points are descriptive and unavailable intervals retain the reason; this is not a general minimum sample-size rule. Absence of this deficit is only a necessary condition, not proof of identification. A positive but ill-conditioned information matrix can be used with a warning when two refined numerical Hessians agree, inversion without regularization is accurate and a curvature-scaled gradient is small. Failure of any check keeps inference unavailable. Passing these numerical checks does not resolve weak information in the data: an interval may still be extremely wide or unreliable near a boundary. Warnings are saved in the `cautions` attribute and `InferenceReview`, and follow printing, default plot subtitles and reports.

Model intervals are asymptotic. Standardized slopes use the full Jacobian for `log(slope) + log(sigma2)/2`, including cross-covariances. Ratio contrasts cancel a common population scale. Difference contrasts use the full joint positive-slope covariance. Extra output includes `PValue` for zero log contrast (ratio one) or difference zero; Bonferroni applies to these p-values too. Containment of one for an individual relative slope is not a pairwise test. The matched `compare_mfrm()` LRT tests all relative slopes equal jointly.

Sandwich inference concerns the working model's limiting parameter; it does not correct bias, informative assignment or a wrong population model. It needs many independent clusters. Rank-deficient cluster scores withhold intervals. Scores aggregate complete Person response vectors, including population-parameter derivatives and the fitted quadrature method. Small, incomplete correctly specified samples already showed undercoverage and extremely wide model intervals; see the GPCM scope vignette. The added methods do not imply universal finite-sample coverage. For an explicit model-based refit alternative use `bootstrap_mfrm_gpcm()`. Relative and standardized slope differences are different targets; their interval performance need not be the same. See the GPCM scope vignette for these distinctions and the observed limits of approximation.

Value

For coefficient intervals, a numeric matrix with percentage-labelled lower/upper columns and row names matching the selected coefficients. Attributes `level`, `method`, `target`, `diagnostics` (estimate, SE, availability and reason), and available covariance checks/cautions retain the interpretation. Unidentified CML comparisons remain missing rows. For `parm = "slopes"`, a matrix of class `mfrm_slope_intervals` with columns `Lower` and `Upper`, named by slope level. Printing shows estimates, bounds and any explanations when a bound cannot be calculated. Attributes `level`, `method`, `target`, and `diagnostics` describe the result. The diagnostics table includes estimates, log-scale and positive- scale SEs, `CIEligible`, `CIUse`, and an `InferenceReview` explaining availability. Unavailable intervals have missing bounds.

Fixed anchors in a one-family fit

Model and sandwich slope intervals can use fixed facet anchors and group means when the joint-information and numerical checks pass. They condition on the supplied values and do not propagate uncertainty from their source calibration. A location anchor does not fix a slope or a step threshold; a fixed group mean does not fix every group member. This does not extend the separate profile-slope, native-location or portable-GPCM routes to anchored fits. Those routes retain their own exclusions.

Information workspace

The joint-information calculation includes all estimated parameters. Option `mfrmr.max_information_bytes` (default 256 MiB) budgets eight dense p-by-p matrices before allocation, or twenty when weak-information refinement is needed. The extra budget is checked before refinement. This workspace estimate excludes response data, quadrature and other allocations and is not a process memory guarantee. A budget refusal retains unavailable results and a reason. There is no separate 80-parameter inference cutoff. Computation still grows with p.

`diagnose_mfrm()` returns the default relative/model 95% calculation in `parameter_uncertainty$slopes`; `fit_mfrm(attach_diagnostics = TRUE)` attaches it to `fit$slopes`. `CIEligible` refers to the requested slope interval only. It does not establish that the fitted likelihood has no boundary solutions or that intervals for other parameters are valid. This method recomputes uncertainty for saved fits; it does not trust old interval flags or alter the supplied object.

Experimental two-family slope intervals

For the fixed-standard-normal MML model with two slope facets, `confint(fit, parm = "slopes")` requests log-Wald intervals for both sets of component slopes. Fixed-grid EM and direct adaptive fits use their own fitted marginal objective. For adaptive fits, differentiation includes movement of the quadrature nodes, their scale and the integration Jacobian; it does not freeze the fitted nodes. It uses the inverse full observed marginal information, including all locations, steps and cross-family covariance; it does not invert the frozen EM Q function or the slope block alone. The first family's log slopes sum to zero; the second family's slopes are free. The result retains `SlopeOwner`, `SlopeLevel` and `ScaleReference` separately. A slope of one has different reference meanings across these families, so no individual unit-slope p-values are supplied.

The default `method = "model"` uses `scale = "standardized"`, without contrasts or cluster adjustments. Bonferroni adjusts for all component slopes in that call. An explicit `method = "profile"` profiles one named owner/level, retaining both families' scale constraints and reoptimizing all remaining slopes, locations and steps. The N(0,1) population is fixed, not an estimated nuisance parameter. Use, for example, `confint(fit, parm = "slopes", method = "profile", slope = c(Task = "t1"))`. Replace Task and t1 with your actual column and level names. Both fixed-grid EM and adaptive direct fits retain their fitted integration method. Adaptive nodes, scales and Jacobians are recalculated at every trial; the source's posterior quadrature is not held fixed during optimization. The interval uses the likelihood loss after nuisance reoptimization for each constrained log slope. Its chi-square(1) calibration remains asymptotic. The result preserves owner, level, scale reference and the same-target Wald comparison through saved plots, reports and exports. This is a component interval, not an interval for the product of two slopes. Experimental location and within-facet contrast intervals use `mfrm_facet_intervals()` and have their own target-specific limitations. Effective slope-product and curve intervals, sandwich intervals and model ranking are not supplied for this model.

Checks reevaluate the source identity, category support, engine-specific convergence and unregularized marginal information. Person marginal-score derivatives use steps 1e-5 and 5e-6; after scaling both by the first matrix's column lengths, their maximum difference must be at most 1e-6. Empirical score rank is retained at relative SVD tolerances 1e-10, 1e-8 and 1e-6, with minimum ranks for both derivative steps in `ScoreRank` and `FineScoreRank`. `ScoreRankDeficient` and an additional warning flag deficient rank. Full empirical score rank is not required for the inverse observed Hessian used here: this Hessian is not the empirical outer product of scores. Rank deficiency alone neither proves structural nonidentifiability nor makes the model-based interval unavailable. It does prevent a nonsingular full-parameter empirical sandwich covariance. The other numerical checks remain necessary, and small-sample coverage is not established.

The fresh mean marginal score must meet the smaller of `em_score_tol` and 1e-6 for EM, or 1e-6 for adaptive direct fitting; the full score scaled by its covariance must have length at most 0.01. Values above 1e-4 retain an additional warning and `OptimizationCaution = TRUE` in the numerical checks. This length is $\sqrt{g^T V g}$, where g is the full negative-log-likelihood gradient and V its inverse information. Under a local quadratic approximation it bounds the Newton displacement of any component log slope in its own SE units. A small residual does not establish a global optimum or accurate coverage. The infinity-norm inverse residual must be at most 1e-6. At the saved parameters, the quadrature order is increased from q to 2q-1 without refitting, preserving the fitted integration method. The change in the score, measured with the original covariance, and the spectral norm of the covariance change, standardized by the original information, must each be at most 0.01. A failed integration check calls for refitting with more points. `mml_quadrature_sensitivity()` preserves the model, engine and integration controls while comparing refits; its `intervals` entries retain each order's interval checks for plotting and reporting. The score workspace is also subject to `mfrmr.max_information_bytes`.

Integration failure and a slope-boundary problem need different remedies. More quadrature points can repair inaccurate integration, but do not ensure a stationary interior solution. For example, let a second-family slope a_r approach zero while keeping $B_r = a_r b_r$ and $T_{rk} = a_r \tau_{rk}$ fixed, where b_r is its location and τ_{rk} its step. An adjacent logit $a_t a_r(\theta - b_t - b_r - \tau_{rk})$ then tends to $-a_t(B_r + T_{rk})$. Response probabilities can have finite limits even while the original location and step diverge. This differs from setting the slope to zero with all other parameters fixed. A small slope alone does not prove a boundary optimum. If such a competing path improves the likelihood, however, merely increasing quadrature order, relaxing stationarity checks or regularizing the information does not justify local Wald intervals at the retained point. Boundary-aware fitting and inference require separate qualification; these intervals do not implement that extension.

These are numerical safeguards, not thresholds calibrated for coverage. Profiling additionally requires absolute low-versus-high quadrature NLL agreement within 1e-5 and gradient agreement within 1e-4, including at constrained solutions. A grid adequate for the local Wald approximation may fail during profiling away from the estimate. Refine and refit the calibration explicitly; no automatic grid change or Wald fallback occurs. Source stationarity uses the two-family SE-scaled checks above; every constrained nuisance solution must pass the absolute 1e-4 gradient check at both quadrature orders. The shared profile algorithm and failure rules described below also apply. No superiority to Wald or general coverage is established, and this does not make a rejected near-zero-slope source eligible. Two-family profile optimizers use a per-Person objective scale internally; likelihood ratios and all acceptance checks use the total likelihood. Passing adaptive log-Wald checks does not by itself qualify the constrained profile searches; every candidate still needs its own integration checks.

Every available two-family interval carries an experimental warning: global identification, absence

of boundary alternatives and sampling coverage remain unestablished. Failed checks retain estimates, missing bounds and reasons. Inspect `attr(ci, "checks")`, `attr(ci, "numerical_checks")` and `attr(ci, "score_rank")`, or `apa_table(ci, which = "checks")`. In `attr(ci, "diagnostics")`, `CIEligible` records numerical interval availability; `CoverageQualified = FALSE` separately records that nominal sampling coverage has not been established for the target and design. This is neither a coverage estimate of zero nor a numerical-failure flag. Both fields persist in interval tables, saved results and exports, including explicitly requested two-family profile intervals. Older saved results may lack `CoverageQualified`; absence does not establish coverage. Plots and saved results/reports retain the approximation and its checks. This calculation does not change the original point estimates or resolve other warnings about the fitted model.

For an unresolved two-family profile, distinguish the original fit from the constrained searches. `maxit` in `fit_mfrm()` controls the original fit; `profile_control = list(maxit = ...)` in `confint()` controls each constrained BFGS stage. Inspect `apa_table(ci, which = "profile_checks")` and `which = "profile_endpoints"`. Checks at the fitted slope can fail when the two constrained starts disagree even though the original fit passed its checks. Increasing `quad_points` requires refitting and addresses integration discrepancies; it does not resolve disagreement between starts. `mml_quadrature_sensitivity()` compares refits while preserving the model, engine and integration method. Its `intervals` entries are model-based Wald intervals; they do not repeat a profile search. To assess a profile on a finer grid, explicitly call `confint()` with `parm = "slopes", method = "profile"` on the corresponding entry in `review$fits`. If the problem is instead a constrained optimizer reaching its iteration limit, increase `profile_control$maxit`; increasing the original fit's `maxit` does not change that profile limit. A search limit or numerical failure does not establish an infinite interval. With little information about a slope, the original fit or a constrained search can fail to converge. Obtaining numerical endpoints after changing search settings does not establish their sampling coverage. The profile method remains an explicit sensitivity analysis, not a routine replacement for log-Wald intervals; no coverage advantage is established.

One prespecified profile interval for one slope family

Use `confint(fit, parm = "slopes", method = "profile", slope = "R01")` to investigate how well one relative discrimination is determined. Choose a level that actually belongs to the fit's slope facet. At each candidate log slope the sum-zero log-slope constraint is retained, including when the selected level is the dependent last coordinate. All remaining slopes, steps, facet locations and estimated population mean and variance are reoptimized. The current scope requires an estimated intercept-only normal population, unit weights, and no anchors or interactions. Standardized slopes, contrasts, JML, covariate populations and simultaneous profile intervals are not supplied.

The connected local interval solves twice the log-likelihood loss equal to `qchisq(level, 1)`. This reference is asymptotic and requires a regular model; replacing Wald intervals does not repair nonregularity or misspecification. Profile coverage and superiority over Wald have not been established in mfrmr. This experimental method is for explicit sensitivity analysis. In small or incomplete datasets, profile limits can remain unavailable even when Wald limits are returned. Evaluating coverage only among successful searches can give a misleading impression of accuracy. A failed search does not prove that a statistical interval does not exist. See the GPCM scope vignette for examples and limits of interpretation. This method is not recommended as a routine Wald replacement.

Two starts (retained and neutral) must agree within 1e-5 NLL. Optimizer codes, nuisance gradients (at most 1e-4) and higher-order integration (NLL change at most 1e-5, gradient change and reference gradient at most 1e-4) are checked. The constrained negative-log-likelihood Hessian must also be finite and positive definite at both quadrature orders: two starts can otherwise agree at

a stationary nuisance maximum or saddle. The attempt table retains the smallest eigenvalues as `NuisanceMinEigenvalue` and `ReferenceMinEigenvalue`. Failed or unresolved curvature leaves the profile point unavailable; no eigenvalue adjustment is used. This local check is not a global optimum test. For two slope families, if the initial two optimization stages do not converge, the search can retry in coordinates scaled by local curvature, with at most `maxit` iterations in this additional stage. This changes only how the optimizer moves through the parameter space; constraints and acceptance checks remain unchanged. The additional stage is available for up to 64 nuisance parameters; an unavailable or unsuccessful retry leaves the earlier result and its reason. After two BFGS tolerances, a stalled converged search may receive the single curvature-based refinement. Its original optimization stages stay recorded; the proposal must improve the gradient without worsening the NLL. Both source and constrained calculations use the fit's integration method; these checks do not establish that variance or slope boundary solutions are absent. Endpoint likelihood-ratio residuals must be at most 1e-4. A better likelihood than the source invalidates both bounds. After a failed outward trial, the search contracts toward the last passing point within the same `max_steps` budget. This can locate an endpoint before the failed trial; it does not skip failed points inside a bracket or change quadrature. All failed attempts remain saved, including when both endpoints are found. Unresolved failures, observed nonmonotonicity or an exhausted search retain missing bounds and reasons, rather than falling back to Wald or declaring an infinite interval. A finite search cannot exclude unsampled or disconnected likelihood regions.

`attr(ci, "profile")` retains all evaluated points, both starts and optimizer stages, parameter vectors, endpoint statuses and elapsed time. `attr(ci, "wald")` retains the same-target model-based Wald comparison. Printing, `plot(ci, type = "profile")`, `apa_table()` and saved-results reporting do not repeat fitting. The curve uses a log slope axis, shows the likelihood cutoff and Wald limits, and does not connect failed points. Inspect `apa_table(ci, which = "profile_checks", digits = 8)` for numerical checks and `which = "profile_endpoints"` for unresolved searches. Large disagreement with Wald motivates investigating asymmetry and weak information; it does not establish that either method has better coverage. Search limits are reported separately from mathematical unboundedness.

References

Fischer, S. M., and Lewis, M. A. (2021). A robust and efficient algorithm to find profile likelihood confidence intervals. Statistics and Computing, 31, 38. doi:10.1007/s1122202110012y. This implementation uses constrained BFGS searches and root finding, not their RVM algorithm.

Zeileis, A. (2006). Object-Oriented Computation of Sandwich Estimators. Journal of Statistical Software, 16(9), 1–16. doi:10.18637/jss.v016.i09.

Zeileis, A., Koll, S., and Graham, N. (2020). Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R. Journal of Statistical Software, 95(1), 1–36. doi:10.18637/jss.v095.i01.

See Also

`gpcm_capability_matrix()`, `build_weighting_review()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("confint.mfrm_fit", run.dontrun = TRUE)
```

```
## Not run:
# Compare intervals for free coefficients and relative discriminations.
ratings <- load_mfrmr_data("example_core")
# Reuse the synthetic GPCM fit; estimation uses 61 quadrature points.
# Full regeneration recipe: examples/measurement-fits.R in the package.
fit <- readRDS(system.file("examples", "measurement-fits.rds",
  package = "mfrmr", mustWork = TRUE))$gpcm
# The default matches coef(fit): steps, locations and log slopes.
coefficients_ci <- confint(fit, level = 0.95)
head(coefficients_ci)
attr(coefficients_ci, "diagnostics") # Availability and interpretation
# Explicitly request positive relative slopes for rater comparison:
intervals <- confint(fit, parm = "slopes", level = 0.95)
intervals
attr(intervals, "diagnostics")[, c("SlopeFacet", "Estimate",
  "CIEligible", "InferenceReview")]
# Relative slopes have geometric mean one. A value above one means a
# steeper response curve, not greater severity or better rater quality.
# Missing bounds need the saved reason; availability is not a coverage test.
plot(intervals)
file <- tempfile(fileext = ".rds")
saveRDS(intervals, file) # Use a lasting path for your analysis
saved <- readRDS(file)
apa_table(saved) # Reuse the intervals without fitting again

# A separate, prespecified profile calculation can be requested with
# confint(fit, parm = "slopes", method = "profile", slope = chosen_rater).
# Choose chosen_rater from fit$slopes$SlopeFacet; see the profile section.

## End(Not run)
```

`confint.mfrm_multivariate_d_study` *Confidence intervals for dependability and SEM in D-study plans*

Description

Quantify uncertainty in relative G, absolute dependability Phi or measurement SEM for specified future counts of common tasks, raters or other random facets. Most methods use a complete balanced ANOVA source. An explicitly selected experimental REML profile also accepts identifiable incomplete MINQUE sources for G/Phi and SEM; future plans still describe complete balanced ratings. For one-facet relative SEM, residual contrasts also accept incomplete sources without likelihood fitting, under the normal residual model. For one-facet G, explicit spectral F intervals accept incomplete sources under independent normal Person effects and residuals. Explicit delta-method coefficient SEs and normal-reference intervals also use the original ANOVA/MINQUE estimator for supported crossed/nested sources.

Usage

```
## S3 method for class 'mfrm_multivariate_d_study'
confint(
  object,
  parm = "G",
  level = 0.95,
  score = NULL,
  composite = NULL,
  assumption,
  draws = 100000L,
  seed = 1L,
  reference = NULL,
  method = "auto",
  ...
)
```

Arguments

object	A result from mfrm_d_study() based on a balanced ANOVA G-study with one or two crossed random measurement facets, or the supported balanced nested design. Nested G/Phi defaults require unchanged source child counts per parent; changed counts require an explicit experimental method. See "Nested plans" below. For G/Phi and SEM, method = "reml_profile" also accepts an incomplete or unequal MINQUE source with its original used ratings retained in the object. method = "residual_contrast" accepts retained one-facet ANOVA/MINQUE ratings for parm = "RelativeSEM" only, including plan differences. method = "spectral_f" accepts these sources for parm = "G", including plan differences, with positive residual degrees of freedom. method = "delta" accepts ANOVA or identifiable MINQUE sources for individual-plan G/Phi, including changed nested child counts. MINQUE requires retained assignments and moment inputs for its sampling covariance.
parm	One of "G" (default), "Phi", "RelativeSEM" or "AbsoluteSEM". With reference, return differences in that metric for the selected source and supported method.
level	Confidence level strictly between zero and one.
score	One original score name.
composite	One named composite, or "Composite" for vector weights. Select either score or composite. With neither specified, select the sole composite, or the first original score if no composite is present. Multiple composites require a selection. Weights must be prespecified.
assumption	Required: "normal" asserts independent normal random effects with constant component variances. The source persons and facet levels are sampled from their intended populations. This is not tested.
draws	Number of independent pivotal draws for one-facet Phi GCI, default 100000. Must be an integer at least 1000, with at least 100 expected draws in each tail at the requested level. Increase it when the returned endpoint Monte Carlo bands are too wide for the intended decision. Unused by other methods.

`seed` Nonnegative integer seed for one-facet Phi GCI, default 1. The caller's random seed is preserved. The current R RNG kind is used and recorded. Unused by other methods.

`reference` Optional row number in `object$design_grid`. Return each other plan's selected metric minus this reference plan's value. The default supports one-facet sources and nested plans with a common future child count (unchanged from the source for G/Phi). Map the whole F set for G or generalized interval for Phi, preserving their different coverage qualifications. Explicit `method = "reml_profile"` also supports crossed/nested plans changing both facet counts, including incomplete/unequal sources. One-facet and common-child nested plans map a shared profile interval; other comparisons constrain the difference directly, using optional nloptr. Its calibration remains experimental, especially at zero differences. For general two-facet crossed/nested comparisons, explicitly select `method = "mean_square_region"` for a conservative joint-region enclosure for G, Phi or SEM. Approximate-F differences are not supplied. At least two plans are required. `NULL` returns individual-plan intervals.

`method` `"auto"` (default) selects `"exact_f"` for one-facet G and `"gci"` for one-facet Phi, including the nested reduction below. For individual G/Phi with two crossed random facets, it selects `"mls"`, an approximate signed-MLS null inversion with conservative completion of undefined widths; see "Two-crossed-facet MLS intervals" below. To compare other choices, or for nested plans with changed child counts, select `"approximate_f"`, `"mean_square_region"`, or a vector containing both to compare methods on the same data and target. Neither is an automatic default. `"mls"` can also be compared with these methods for crossed G/Phi. `"mean_square_region"` requires the optional lpSolve package. Balanced two-crossed-facet G also accepts explicit `"score"`, an approximate score inversion with extended nuisance parameters. It uses stored mean squares, preserves original points and requires no optional optimizer package. It can be compared with the other supported G methods. It is not a Phi, SEM, nested, incomplete-source or `reference` method. For SEM, `"auto"` selects `"mls"`: a single contributing mean square uses its chi-square interval; positive sums use Graybill-Wang MLS and signed sums use Ting MLS. This is a specified calculation rule, not selection based on which interval looks favorable. `Method` records the calculation used for each plan. Two-facet crossed/nested SEM also accepts `"mean_square_region"`, using linear error-variance projections followed by square roots. Select one method at a time unless including `"reml_profile"`, or comparing supported experimental coefficient methods above. `"reml_profile"` is an explicit G/Phi or SEM choice for balanced or incomplete sources. It requires optional lme4 and minqa, fits a separate normal REML model, and retains the original point estimate. It can be compared with other methods for the same supported target in a vector; `reference` uses a shared interval image or a direct difference profile as described below. The latter additionally requires nloptr. Its chi-square calibration is approximate, especially at component boundaries. It is never selected automatically. No fallback follows a failed calculation. Explicit `"residual_contrast"` supplies one-facet RelativeSEM intervals and differences from retained ANOVA/MINQUE ratings without likelihood fitting or simulation. It maps one residual chi-square interval, preserves original points and reports the separate projected `IntervalCenter`. It

can be compared with MLS or REML for this supported target. It is not an AbsoluteSEM, G/Phi or multifacet method; see *Residual-contrast SEM*. Explicit `"spectral_f"` supplies one-facet G intervals and paired differences from retained ANOVA/MINQUE ratings. It removes facet effects and inverts an exact normal-model F pivot without likelihood fitting or simulation. It preserves original points and can be compared with `"exact_f"` (balanced sources) or `"reml_profile"`. It is not an automatically selected, Phi, SEM, nested or multifacet method. Explicit `"delta"` supplies G/Phi sampling SEs and normal-reference intervals from the original component estimator and its full sampling covariance. It accepts supported one-facet, crossed and nested sources, including incomplete MINQUE, and can be included in a method vector. It requires no refitting or simulation. It is not a SEM or `reference` method. Defaults are unchanged; see *Delta coefficient SEs and intervals*.

`...` Reserved; additional arguments are rejected.

Details

For a source-by-target choice table, see *Choose the G-theory target and uncertainty* in mfrmr_methods_reference. Incomplete-source calculations are available through explicit methods; this does not qualify a generally calibrated automatic default. Read the selected method's assumptions below.

Each interval concerns a population coefficient or SEM of a future plan: uncertainty comes from the source G-study, not from collecting the proposed future ratings. GENOVA-adjusted D-studies can use the same population-parameter intervals. `Estimate` and `PointStatus` describe the adjusted point; `IntervalEstimate` and `IntervalPointStatus` retain the original source-study projection. `ComponentTreatment` and `IntervalBasis` distinguish their bases. The original source mean squares, data, bounds, diagnostics and unavailable states are preserved; adjusted components never replace the source inputs. The bounds are not recentered and need not contain the adjusted point. An existing method-specific `IntervalCenter` remains separate from both. Where a method supplies a coefficient sampling SE, it is named `IntervalSE` for adjusted results. `SamplingVariance`, error-variance columns and other diagnostics also retain their original method basis. None describes the sampling distribution of the zero-substituted or EMS-recalculated estimator. In the method descriptions below, references to the original `Estimate` mean `IntervalEstimate` when component treatment is adjusted.

This distinction follows from the common population target: changing a reported point does not change the confidence-set event based on the source study. Exact procedures keep their stated normal-model qualifications; approximate or experimental methods do not become exact through adjustment. GENOVA's unchanged component SEs cannot be interpreted as an adjusted estimator's SE. Its truncation is nonsmooth at zero; an adjusted-estimator delta interval would require additional derivation and boundary validation. Saved intervals must match the D-study's treatment as well as its source. Use `confint(d, reference = ...)` for supported population plan differences; the separate `mfrm_multivariate_d_compare()` delta-estimator API still requires raw components.

For a balanced two-crossed-facet source, start with `confint(d, parm = "Phi", assumption = "normal")` for absolute dependability, or `parm = "G"` for relative ordering. Read `Estimate`, `Lower`, `Upper`, `Status` and `FormulaComplete`. The default is an approximation: a returned interval is not evidence that the model assumptions hold. Use `plot(d, metric = "Phi", intervals = TRUE, assumption = "normal")` to add its ribbon, and `attr(ci, "notes")` to read cautions saved with a result `ci`.

Value

A data frame of class `mfrm_dstudy_intervals`, with scenario/facet counts, `Kind`, `Score`, `Metric`, `Estimate`, `PointStatus`, `Lower`, `Upper`, unrestricted `RawRatioLower` and `RawRatioUpper` (balanced F/GCI coefficient methods), `Level`, `Method`, interval `Status`, `MethodCode`, `Available`, `NegativeComponent`, and `Validation`. Each selected method supplies a row per plan, including unavailable rows. Delta results additionally supply `SE`, `SamplingVariance`, `SEStatus`, `AtBoundary`, `OutsideParameterSpace` and `SamplingCovariancePSD`. Attribute `sampling_covariance` retains the component covariance with its own diagnostics and weights divided by their largest absolute value; `weights` retains the original score definition. Row-aligned `diagnostics` records normalized gradients, error coefficients and scales. MINQUE results retain `source_data` for source-identity checks. Method comparisons including delta retain the complete per-method objects in `method_results`. Spectral F additionally supplies `LogVarianceRatioLower/Upper` and projection/root diagnostics in `pivot`; it does not supply a replacement point estimate or SE. Coefficient MLS additionally retains `NumericalComplete`, `FormulaComplete` and `Empty`. Its row-aligned diagnostics contain the null problem, quadratic coefficients, accepted pieces/nodes, and any conservatively retained `undefined` ranges. Formula completeness and numerical availability are separate from the approximate statistical calibration. Score intervals additionally retain `NumericalComplete` and `Empty` to distinguish a resolved empty numerical set from a failed calculation; either has missing accepted endpoints. Scanned pieces and root checks remain in the row-aligned `diagnostics` attribute. `plot()` compares saved intervals without recomputation. `design_grid` retains future counts. For two facets, `diagnostics` is a row-aligned list of F moments/roots or solver status/residuals; `mean_squares` stores the normalized source mean squares, their scale, df and component map. Attributes `weights`, `assumption`, `source_design`, `source_rows`, `pivot` (G's log ratio/endpoints and degrees of freedom; Phi's single-rating bounds and degrees of freedom), and `notes` retain context. For Phi, `RawRatioLower/Upper` are `NA` because there is no G-type F ratio; `monte_carlo` retains numerical precision diagnostics. Phi endpoints and their numerical bands are withheld when a positive width rounds away. If the base pivotal interval has lost resolution, plan differences also remain unavailable, including identical-count comparisons. The pivot attribute retains the diagnostic base bounds. A sampled atom at zero retains its separate `"Upper endpoint at zero"` status. Save the whole object with `saveRDS()`. `source_summary` retains the design, covariance components, mean products and degrees of freedom for checking that a curve overlay uses the same source. REML profiles additionally retain `source_data` to check the used ratings, `reml_source` with the fit and numerical diagnostics, and row-aligned profile `diagnostics` with scans, accepted pieces, optimizer attempts and gradient checks. No live fitting environment is saved. `IntervalCenter` is the separate REML estimate; `CandidateLower/Upper` can remain finite when accepted bounds are unavailable. Shared-image REML `reference` results retain the common fit and scans in `base_interval`, with row-aligned transformation data in `diagnostics`. Direct difference profiles instead retain per-comparison scans and constraints in `diagnostics`, and all reference counts in `attr(result, "reference")$counts`. Residual-contrast SEM also retains `source_data` and a separate `IntervalCenter`; `base_interval` contains the residual variance interval, `residual_projection` its numerical checks, and row-aligned `diagnostics` the count multipliers. Method comparisons that include REML or residual contrasts retain each complete single-method object in `method_results`, so distinct bases and centers are not conflated by the combined table. For MLS SEM, individual-plan tables additionally retain `ErrorVariance`, `ErrorVarianceSE`, `RawVarianceLower` and `RawVarianceUpper` in squared score units. The squared `ErrorVarianceSE` uses the unbiased `df + 2` sampling-variance formula described in `confint.mfrm_multivariate_gstudy()`; no delta-method SEM SE is supplied. `diagnostics` retains component and mean-square weights indexed by plan, and `mean_squares`

retains the scale and source df. With `reference`, rows exclude the reference plan, `Metric` is the selected metric followed by `" difference"`, and `Estimate` is comparison minus reference. `Reference`, `ReferenceCount`, `ReferenceValue`, and `Value` retain the two plans' context. Attribute `reference` also retains its point status. For MLS SEM differences, it retains the reference SEM interval, raw error-variance bounds, estimate and SE in place of per-plan variance columns. Nested results also retain `nesting_context` with facet identities, the source children per parent, and both counts for every future plan. A nested reduction/MLS `reference` includes `count_facet` and `children_per_parent`. General region comparisons instead retain all reference `counts`, its interval, status and diagnostics; `individual_intervals` stores the planwise projections before forming the difference enclosure. Their `diagnostics` list aligns with comparison rows. No scalar `ReferenceCount` is invented for a comparison changing both facets. No sampling SE or covariance matrix is inferred from the interval width. Stored G-study summaries are used and point projections are recomputed without refitting for the summary-based methods; REML profiles fit the stored used ratings. Every method leaves the input unchanged.

Two-crossed-facet MLS intervals

Under the balanced independent normal-effects model, mean squares `M[j]` are independent `lambda[j] * chi-square(df[j])/df[j]`. Let `theta=A*lambda` be the variance components, `U=theta[Person]`, and `E=error' theta` the relative or absolute error variance for the prespecified future counts. A candidate `rho=U/(U+E)` is equivalent to the linear null `h(rho)'lambda=0`, with `h(rho)'=((1-rho)e_P' - rho*error')*A`. The denominator must be positive; positive source mean-square expectations and residual variance are assumed. No estimated ratio or component is clipped.

Compute the unrestricted Ting MLS interval for this signed linear combination and retain `rho` when it contains zero. The asymmetric chi-square/F corrections use source degrees of freedom. This is mfrmr's inversion of the published linear-combination construction; Ting et al. and Hess/Iyer do not directly establish coverage for this G/Phi procedure. The inversion inherits approximate, rather than exact, calibration. The checked two-term cases at the usual 95% level reproduce F inversion. This is not an identity for arbitrarily low confidence levels: squaring a reversed chi-square correction can lose its direction. The sampling evaluations below concern 95% intervals.

Between coefficient sign changes, the squared widths and squared null point are quadratic in `rho`. All polynomial crossings in `[0,1]` are evaluated, including sign-change points, where the MLS formula can be discontinuous. Diagnostics retain accepted pieces and individual nodes; `Lower`/`Upper` enclose their union. This avoids a finite grid or repeated model fitting. The polynomials use shifted and scaled coordinates to resolve narrow sets near a boundary. If root separation cannot be represented on the coefficient scale, bounds are missing with an explicit `Status` and `NumericalComplete=FALSE`; the point estimate is retained. This is distinct from a genuine empty set or a formula-defined endpoint at zero or one.

A squared MLS width can be negative for some candidate values, especially with few degrees of freedom and highly unequal components. Such candidates are conservatively retained rather than assigned a zero width or silently dropped. `FormulaComplete=FALSE` identifies this completion, and each row's `attr(ci, "diagnostics")` entry retains the `undefined` ranges. This can only enlarge the retained set; it cannot make the approximate calibration exact. A numerical inversion failure remains unavailable and is distinct from this defined completion rule. An empty set also remains explicit. An unavailable original ANOVA point does not prevent calculation of an interval.

Normal-model checks illustrate both potential and limits. In a matched 400-source study with 60 persons, 4 raters, 12 tasks and component variances `(1,.2,.3,.3,.4,.1,1)`, the source-plan G

interval had mean width .0945 and coverage .950; Phi had mean width .3287 and coverage .970. The conservative mean-square region gave widths .1974/.7249 and coverage 1/1 in those same samples. The component order is Person, Rater, Task, Person:Rater, Person:Task, Rater:Task, residual. A further eight-condition normal study used 2,000 sources per condition, Person counts 5–80, source facet counts from 2-by-2 to 2-by-20, zero/near-zero Person variance, dominant facets/interactions and changed future counts. All-attempt inclusion was .9505–.9885 for G and .948–1 for Phi after the stated completion. At .95, Monte Carlo SE is about .0049 with 2,000 sources. Some small-source Phi conditions returned `[0,1]` in over 90% of available cases. These are limited normal-model evaluations, not uniform coverage, robustness, or a promise of useful precision in every design. Increasing future counts does not compensate for a weak source design.

This choice supplies pointwise individual-plan intervals for balanced two- crossed-facet sources and prespecified scores/composites. It does not supply simultaneous curve bands, plan-difference intervals, or incomplete/nested- source inference. The exact one-facet G and generalized one-facet Phi defaults retain their different constructions. The name `"mls"` for SEM instead refers to a direct error-variance interval followed by a square root.

Interpreting the target and other defaults

More future tasks can improve G but do not increase the source degrees of freedom. A narrow interval is not guaranteed by a large proposed task count. G concerns relative ordering; these intervals do not give pass/fail accuracy, or Person-rank intervals. Phi includes the random measurement facet's main effect in absolute error; use it when absolute score levels matter.

The F pivot is exact under the balanced independent normal-effects model, with positive residual variance. It is not a distribution-robust interval for arbitrary numeric ordinal scores. It applies to a selected original score or a fixed linear composite, including prespecified difference scores. General two-facet plans and incomplete-source MINQUE(0) do not have this pivot. The nested unchanged-child-count subcase below does reduce to it. Balanced F/GCI, MLS, score and mean-square-region methods use stored ANOVA summaries. The score method fits null nuisance expectations to those summaries. One-facet Phi GCI draws independent chi-square quantities; it is not an exact frequentist-coverage guarantee. Spectral F and residual contrasts require retained ratings; delta uses ANOVA summaries or retained MINQUE assignments and moment inputs. REML profiles require used ratings and refit a separate model. These input requirements also apply to saved G-study objects; reopening already computed intervals does not refit them.

Score intervals for two-crossed-facet G

Explicit `method = "score"` uses the independent normal-theory mean-square sampling law of a balanced Person-by-two-random-facet source. For proposed G = ρ, write the population target as $\theta_P/(\theta_P + e^T\theta)$ and test $h(\rho)^T\lambda = 0$, where $\theta = A\lambda$, $h(\rho)^T = ((1-\rho)e_P^T - \rho e^T)A$, and λ contains the positive mean-square expectations. Nuisance components may be negative as long as this restricted marginal covariance remains positive definite. These computational values do not replace the ANOVA components. The nuisance score is checked independently of optimizer status labels. If $S_\lambda = \kappa h$, the expected-information statistic is $T = \kappa^2 \sum_j 2h_j^2\lambda_j^2/df_j$. Invert its approximate chi-square(1) test over $0 \le \rho \le 1$.

This construction is motivated by Shedden and Ekvall's extended-nuisance score approach; their preprint does not establish G/Phi ratio intervals. For the active independent strata, the spectral/Frobenius bound in their asymptotic condition is $1/\sqrt{\min_j df_j}$. G uses Person-containing strata, whose degrees of freedom increase with source Persons even with fixed facet counts. This does not give

exact finite-sample coverage. A retained normal-model evaluation used 360 independent sources per condition, six conditions and three future plans: true-value inclusion was 93.1–97.8%, with Monte Carlo SEs of 0.8–1.3 percentage points. These checks test the statistic at the truth, not complete endpoint coverage. The lowest cell's exact binomial 95% interval was 89.9–95.5%. The method remains experimental and is never selected automatically.

The numerical inversion compares two finite grids, includes contrast sign-change locations, and verifies cutoff crossings. Diagnostics retain all scanned fits and accepted pieces; bounds are their envelope. These checks do not prove global nuisance optima or exclude narrower undetected pieces. `NumericalComplete` describes the completed numerical check. `Empty = TRUE` means the scanned confidence set is empty: it is not an estimate of zero or zero uncertainty. Empty sets and numerical failures both have missing `Lower`/`Upper`, but distinct status and diagnostics. `CandidateLower`/`CandidateUpper` retain exploratory bounds after failure. Cutoff-crossing errors must be smaller than `min(1e-5, 1e-5 * critical)`. If this tolerance falls below `64 * .Machine$double.eps * max(1, critical)`, the cutoff is numerically unresolved: bounds are missing, `NumericalComplete = FALSE` and `Empty = FALSE`. Original points remain available; this is not an empty confidence set or a minimum sample-size restriction. Diagnostics retain `critical`, `cutoff_tolerance`, `numeric_resolution`, `root_tolerance` and `grid_tolerance`. Root and grid-comparison tolerances tighten for small critical values. These arithmetic checks do not certify global nuisance roots or complete confidence-set topology. Positive observed mean squares are required. Few source Persons can yield broad intervals or inaccurate calibration. The method is pointwise for a prespecified score/composite and future plan, with no distribution-robust or simultaneous-coverage claim.

Phi also depends on few-df facet strata. Extending this same score rule to Phi produced very broad sets and changes between nuisance roots near zero; it is therefore not offered for Phi. Separate score intervals cannot be subtracted to obtain a paired-plan difference interval.

REML profiles for balanced or incomplete sources

For a fixed score or composite, this method fits the normal random-effects model of the declared grouping assignment: one common mean and independent Person, facet, interaction and residual effects. Source omissions must be noninformative conditional on that fixed assignment. It does not model outcome-dependent missingness, unequal future exposure or selection of score weights from the observed data. The residual variance is positive. Raw used ratings are needed; ANOVA/MINQUE component summaries alone are insufficient for the incomplete-source likelihood.

Write the variance ratios as `v`, with residual ratio one. For each plan, let `c` contain its relative or absolute error divisors, with `c[Person] = 0`. The coefficient is `rho = v[Person] / (v[Person] + sum(c * v))`. Fixing a proposed `rho` eliminates the Person ratio as `rho / (1 - rho) * sum(c * v)`. The remaining nonnegative ratios are optimized; the common mean and residual scale are profiled analytically using the restricted Gaussian likelihood. The accepted numerical set has deviance increase no larger than `qchisq(level, df = 1)`.

For SEM, constrain the error variance `tau = sum(c * theta)` directly, where `theta = s2 * v` includes the positive residual variance `s2`. The profiled residual scale at fixed ratios is `s2_hat(v)`. Restore it to the criterion using `D(v,s2) = D_profile(v) + (n-1)*(log(s2/s2_hat) + s2_hat/s2 - 1)`. At fixed positive `tau`, substitute `s2 = tau / sum(c*v)` and optimize the remaining ratios. Take square roots of the error-variance endpoints and restore the original score units. This is a constrained likelihood for the error variance, not addition of component bounds or a delta method. The residual scale cannot be profiled freely while fixing an absolute variance target. SEM intervals de-

scribe sampling uncertainty in the population measurement SEM, not intervals for individual Person scores.

This is an implementation of target-specific restricted-likelihood inversion, not a MINQUE likelihood or a published exact G-theory pivot. Bates et al. describe the REML criterion and sparse computation, not a validation of these G/Phi or SEM intervals. It also differs from lme4's ordinary parameter-profile intervals, which use an ML criterion. The chi-square cutoff is asymptotic; a generic boundary mixture is not substituted. The special-case results of Crainiceanu and Ruppert illustrate why boundary calibration cannot be assumed for arbitrary multi-component designs.

A limited normal-model feasibility check used 40 samples per assignment and population condition. True-value likelihood-ratio acceptance ranged from 92.5% to 100% with positive components and from 87.5% to 100% in the checked zero-component populations, counting numerical failures as misses. These imprecise checks do not establish 95% coverage or general robustness, and true-value acceptance is not a full endpoint-performance study. Numerical endpoint calculations were separately checked on selected data. Use this explicit experimental option only with its qualifications; agreement with another interval in one dataset does not establish coverage.

A separate SEM feasibility check reused 40 samples per assignment and population. Positive-component true-value acceptance ranged from 92.5% to 100%; checked boundary populations ranged from 82.5% to 100%. The lowest result was absolute SEM in an incomplete crossed source with 30 persons, 4 raters and 5 tasks and zero facet/interaction variances; numerical availability there was 97.5% and the inclusion Monte Carlo SE was 6.0 percentage points. These are coarse warning signals, not evidence that nominal 95% inference is reliable. They concern acceptance at the true target, not coverage of all returned interval endpoints.

Three starting points, an independent BOBYQA optimization, and finite differences with boundary checks verify numerical solutions. For G/Phi, a grid including the REML estimate locates cutoff crossings; root finding refines them. Disconnected accepted pieces are retained in `diagnostics`, while `Lower/Upper` give their envelope. If the REML coefficient center rounds to one, the current coefficient-scale search is unavailable: bounds remain missing with an explicit numeric-resolution `Status`. The source optimization may still have passed its checks; the rounded center does not establish perfect reliability or zero uncertainty, and the uncomputed interval need not have zero width. This interval search also does not certify a global optimum or exclude narrow pieces missed between grid points. An unresolved profile or upper tail is flagged, with exploratory `CandidateLower/Upper` retained separately; it is not replaced by a zero-width or `[0,1]` interval. For a very small requested confidence level, a likelihood-ratio cutoff indistinguishable from numerical error leaves the interval unavailable. Endpoint tolerance is `min(1e-5, 1e-5 * cutoff)` and must exceed the floating-point resolution check `64 * .Machine$double.eps * max(1, abs(source_deviance))`. Row-aligned `diagnostics[[i]]$cutoff` retains these values. This check does not certify optimizer accuracy. A positive interval width or nonzero endpoint lost on restoring score units also gives missing bounds with an explicit `Status`, keeping the original point and representable REML center. These checks apply to component, covariance, G/Phi, SEM and direct-difference profiles; an identical-plan direct contrast remains an exact zero when its source checks pass. Separate score and weight scaling preserves a composite under reciprocal changes of units.

For SEM, `log1p` variance-ratio coordinates retain exact zero components while reducing scale differences. Search outward from the REML center until the first checked cutoff crossing in each direction, then refine the roots and check their likelihood-ratio residuals. This returns a numerical interval around the center; it does not enumerate disconnected remote accepted regions. A finite scan cannot prove global nuisance optima or rule out narrower undetected crossings. Unresolved required evaluations, roots or tails retain unavailable bounds and separate candidates; reaching a

search limit does not justify an infinite endpoint. Proportional error-weight vectors reuse the same profile by scaling the error-variance targets. This is an exact transformation, without new fits.

`Estimate` and `PointStatus` always retain the source ANOVA/MINQUE result. `IntervalCenter` is the REML estimate of the same population coefficient or SEM. A source point can be unavailable or outside the profile interval; fitting nonnegative REML variances does not repair the original component matrices. Forests and D-study curves retain the original source points. Save the interval object to avoid fitting again when plotting or reopening it. Individual-plan intervals are pointwise. The shared-profile differences below have a separate common-base justification. The normal assumption and a successful optimizer do not imply accurate finite-sample calibration.

Plan differences from one shared REML interval

With `method = "reml_profile"` and `reference`, compare one-facet plans or nested plans with the same future child count per parent. The child count may differ from the source. The observed assignment, including incomplete cells and unequal child counts, stays in the full REML model; it is not averaged or replaced by balanced data.

These future plans have error variance `e0 / m`, where `m` counts the measurement-facet levels or nested parents. For G/Phi, let `t = U / e0`, where `U` is Person variance. The coefficient at count `m` is `m*t / (1+m*t)`. Calculate one base profile at count one, then map its whole interval through the comparison-minus-reference function. With counts `b` and `a`, its stationary point is `t = 1/sqrt(a*b)`; this must be included when inside the base interval. Subtracting marginal confidence endpoints or transforming only the two base endpoints is wrong. For SEM, multiply the shared base interval for `sqrt(e0)` by `1/sqrt(b) - 1/sqrt(a)`, reversing limits when this factor is negative.

Whenever the base interval contains its population target, all returned images contain their true differences for this fixed score/composite. This event transfers the base method's coverage; it does not establish exact nominal coverage or remove numerical-search limitations. Mapping an envelope can be conservative, and SEM still uses the interval around its REML center. No additional model is fitted for each comparison. Original ANOVA/MINQUE differences remain in `Estimate`, with the REML differences in `IntervalCenter`. An unavailable base remains unavailable, even for identical plans; candidate images are exploratory only.

`attr(result, "base_interval")` retains the complete count-one interval and its profile diagnostics. Plot the comparison with `plot(result)`; difference intervals cannot be used as individual-plan curve ribbons.

Direct REML profiles for other plan differences

For crossed plans or nested plans with different future child counts, the same call uses a direct difference constraint on the shared source likelihood. It does not subtract marginal confidence endpoints. Let `a` and `b` be reference and comparison error weights, `v` the variance ratios with residual ratio one, `ea = sum(a*v)`, `eb = sum(b*v)` and `U = v[1]`. The G/Phi difference is `U*(ea-eb)/((U+ea)*(U+eb))`. A nonzero target can have two positive solutions for `U`, on either side of `sqrt(ea*eb)`; both seed the search. At zero difference, fit both the zero-Person and equal-error branches. The equal-error solution also seeds nearby targets to avoid following only the zero-Person branch. Multiple starts and independent constraint and stationarity checks still cannot certify global optima.

The SEM difference is `sqrt(s2)*(sqrt(eb)-sqrt(ea))`, in standardized score units. The constraint therefore restores the residual scale in the full REML criterion. At zero SEM difference the errors must be equal; zero Person variance alone does not satisfy this null. Final SEM bounds are multiplied by the source score scale, retaining signed score units.

A grid and checked cutoff crossings retain detected accepted pieces; `Lower/Upper` enclose those pieces, not a guaranteed enumeration of the global set. `attr(result, "diagnostics")` retains each comparison's `sets`, `scan`, `profiles`, `roots`, numerical failures and candidate bounds. SEM diagnostic targets and sets are standardized; multiply by `attr(result, "reml_source")$score_scale` for the original score units. Required failed evaluations or unresolved tails leave accepted bounds unavailable. A search limit is not an infinite confidence bound.

These comparisons are **pointwise and experimental**, not simultaneous. Their chi-square(1) calibration and finite-sample coverage are not established generally. In particular, the two branches of the zero coefficient null intersect at a singularity. Drton (2009) explains why singularities and boundaries can invalidate ordinary likelihood-ratio calibration; that paper is not a derivation of the present G-theory method. Neither numerical completion nor a warning supplies a coverage guarantee. Prefer a method with justified coverage for the design and inferential purpose; use this explicit choice to examine the fitted likelihood while retaining its qualifications. It is never selected automatically.

Sampling uncertainty in measurement SEM

SEM describes measurement error in score units; its confidence interval describes how uncertain that population SEM is given the source study. It is not a confidence interval for a Person's score, a sampling SE of G, or an interval obtained by adding component confidence bounds.

Write the raw component estimates as `A %*% MS`. If `w` contains the future-count divisors for relative or absolute error, the error-variance estimate is `w' A MS`. Apply the normal-model interval to this one linear combination of independent mean squares, intersect it with nonnegative variances, then take square roots. This monotone transformation preserves the original interval's coverage event. It does not improve an approximate MLS interval's coverage. Component formulas and nested child-count units are given in `mfrm_d_study()`. With `method = "mean_square_region"`, project the common simultaneous mean-square region onto each error variance, then take square roots. This retains conservative normal-model coverage, including joint coverage over future plans for the fixed score. It can be much wider than MLS and has no MLS sampling-SE columns; `diagnostics` retains variance-scale bounds.

The single-mean-square interval is exact under the stated normal model when its expectation is positive. For example, one-facet relative error is `MS_residual / future_count`. A nested relative error also reduces to one mean square when the future number of children per parent equals that in the source. Otherwise positive combinations use Graybill-Wang and signed combinations use Ting MLS, as set out by Hess and Iyer (2002). Betebenner (1998) applies these constructions to G-theory error variances. Both are approximations, not distribution-robust or simultaneous intervals. Their squared chi-square corrections can lose their usual direction at very low confidence levels; the sampling evaluation below concerns 95% limits. If a positive variance interval or its square-root image loses its width through numerical rounding, the SEM limits are missing and `Status` explains the loss of resolution. The original SEM and any calculable error-variance SE are retained. This also applies to residual-contrast SEM. An unavailable common source interval does not become a zero-width plan-difference interval, even for identical counts. Recompute `confint(d, ...)` from a saved D-study to apply current numerical checks; a G-study refit is not required.

A bounded check used nine normal-model source conditions with 2,000 replications each: one-facet, crossed and nested small/larger designs, including zero components. Each had three future plans and both SEMs. Across the 51 positive-error targets, nominal 95% coverage ranged from 93.45% to 99.00% (Monte Carlo SE 0.22–0.55 percentage points), and availability from 99.75% to

100%. The lowest coverage concerned absolute SEM with 10 persons, 2 raters and 3 tasks, projected to one rater and one task. The all-zero relative-error cases returned unavailable intervals. This limited check identifies undercoverage and conservatism; it is not a guarantee for other populations, designs or nonnormal scores.

For a one-facet source, every SEM is `s / sqrt(m)` for the same single-rating error SD `s` and future count `m`. With reference count `a` and comparison `b`, transform the whole interval for `s` by `1/sqrt(b) - 1/sqrt(a)`, reversing endpoints when negative. This retains dependence between plans. The common base coverage event protects all returned differences for that fixed score; relative SEM has exact normal-model coverage for positive residual variance, while absolute SEM retains approximate MLS coverage. Lower SEM differences favor the comparison plan. Identical counts give `[0,0]` only when the base interval is available. Nested comparisons with a common future child count use the same mapping in the parent count; their base may require Ting MLS. `mfrm_multivariate_d_compare()` retains its separate paired-delta scope.

Spectral F intervals for one-facet G

Use `method = "spectral_f"` when a one-facet source is incomplete or unequal and the independent normal-effects model is appropriate. The future plan must still be complete and balanced. Unlike residual-only SEM inference, this method requires normal Person effects as well as normal residuals. Assignment and omissions must preserve the assumed model.

Write the scalar or fixed-composite response as `y = Z_P p + Z_F f + e`, with Person variance `sigma_P^2` and positive residual variance `sigma_E^2`. The facet incidence contains the intercept. Removing its column space also removes every realized facet effect without changing the population G target. Let `U diag(sqrt(lambda)) V'` be the thin singular decomposition of the projected Person incidence, keeping its `r` positive singular values. Then `z = U' y` has independent normal coordinates with variances `sigma_E^2 + lambda_i * sigma_P^2`. Projecting off both Person and facet incidence supplies independent `S_E / sigma_E^2 ~ chi-square(nu)`, where `nu = n - rank([Z_P, Z_F])`.

For `t = sigma_P^2 / sigma_E^2`, `F(t) = {sum(z_i^2 / (1 + lambda_i*t)) / r} / (S_E / nu)` has an F distribution with `r, nu` degrees of freedom at the true ratio. This is a specialization of the spectral pivot family in E, Hannig and Iyer (2008), equations 3–7, using all positive Person eigenvalues in the numerator and pure residual directions in the denominator. It is not their fiducial procedure or Burch and Iyer's optimal choice among partitions. The pivot decreases with `t`. Invert its central probability interval on `t >= 0`; an empty intersection remains an empty set. Analytic brackets using the smallest and largest eigenvalues locate endpoints, followed by checked one-dimensional root finding on the log-ratio scale. There is no likelihood optimizer, arbitrary upper parameter limit or pivotal simulation.

Each future count `m` maps the same set by `G_m = m*t/(1+m*t)`. With `reference`, map the entire set through the G difference, including any interior extremum; see *Differences in G between plans*. The common event protects all returned plans/differences for this prespecified score. G intervals have exact normal-model coverage; difference images may be conservative. This is not a joint guarantee across scores or metrics, or protection against nonnormality, residual dependence or selected weights.

In a balanced source all positive eigenvalues equal the source facet count, recovering the usual `"exact_f"` limits. In an incomplete source, removing facet directions can discard useful information. Exact coverage does not establish optimal precision: sparse, small sources can give very wide intervals. No remaining residual contrasts, unresolved projection rank/variation and failed roots are

reported explicitly. The residual combines Person-by-facet interaction and observation error. Distinct log-ratio limits that round to the same G value are also marked unavailable on that numeric scale, rather than displayed as zero uncertainty. This also applies to nonconstant plan-difference images whose widths or nonzero endpoints are lost to rounding. Identical-count differences are exactly zero when their base set is available; they cannot bypass a failed base calculation. A genuine singleton `{0}` from intersecting the F set with nonnegative ratios retains `Status = "Upper endpoint at zero"`; it is distinct from coincident finite bounds caused by lost numerical resolution. Extremely small confidence levels can make the two F quantiles numerically indistinguishable. On the log-F scale, endpoint tolerance is the smaller of `1e-8` and `1e-5` times their separation. If it falls below `64 * .Machine$double.eps * max(1, abs(log_quantiles))`, the interval is unavailable with an explicit reason. This is an arithmetic-resolution check, not a small-sample restriction or a test of the model assumptions.

`Estimate` retains the original ANOVA/MINQUE G or difference, including unavailable points; no replacement center or sampling SE is supplied. `LogVarianceRatioLower/Upper` concern log(Person variance/residual variance), where `-Inf` denotes zero. These differ from the mean-square-ratio fields of the balanced F method. The `pivot` attribute saves eigenvalues, normalized squared coordinates, residual mean square, df and root checks. `endpoint_tolerance` and `quantile_resolution` record the numerical checks. Source-matched summaries and ribbons can reuse saved results.

Residual-contrast SEM

For a one-facet source, relative measurement SEM at future count `m` is `sigma_E / sqrt(m)`. Thus `method = "residual_contrast"` first obtains one residual variance interval using the projection in `confint.mfrm_multivariate_gstudy()`, then takes square roots and divides by `sqrt(m)`. This applies to incomplete or unequal observed assignments that retain residual degrees of freedom. The future plan is still a complete balanced assignment; its counts do not describe the sparse source.

Under independent homoskedastic normal residuals and positive residual variance, the residual chi-square law is exact. Every plan interval is a monotone image of that same event. With `reference`, multiply the common residual-SD interval by `1/sqrt(comparison count)-1/sqrt(reference count)`; reverse endpoints for a negative multiplier. The common event protects all returned plans/differences jointly for this one prespecified score/composite. Identical-count differences are zero whenever the base interval is available. This does not provide a joint guarantee across different scores, metrics or adaptively chosen weights, nor robustness to selective omissions, nonnormal residuals or residual dependence. The source, not the future count, supplies the degrees of freedom and information.

`Estimate` and `PointStatus` retain the original ANOVA/MINQUE SEM or difference. `IntervalCenter` is the separately projected SEM or difference; both and the endpoints use score units. A negative original residual estimate can leave `Estimate` unavailable while the projected interval remains available. The original point is not replaced by the projected center or zero. No sampling SE of the original SEM estimator is supplied. The `base_interval` attribute retains the residual **variance** interval in squared score units, while `residual_projection` retains df and checks. No remaining contrasts, numerically unresolved variation or a failed base interval leaves limits unavailable, including for identical-count pairs.

Absolute SEM and multifacet SEM involve additional components; this transformation does not supply their intervals or G/Phi intervals. Use `summary(d, intervals = TRUE, parm = "RelativeSEM", method = "residual_contrast", assumption = "normal")` to collect supported results, or the same method in `plot(d, type = "sem", metric = "RelativeSEM", intervals = TRUE, assumption = "normal")` for curve ribbons. Save the complete result to reuse it without repeating the projection.

Nested plans

Suppose raters are nested within tasks: different raters serve each task, and every person is rated by all of them. If the source has three raters per task and every future plan also has three, task counts may vary while using `confint()` for G, Phi and their differences. This does not make the raters or tasks fixed. New random tasks and their random raters remain the generalization population.

Let `R` be the balanced source child count per parent. Within-cell means over the children have effective components `vP`, `vParent + vChild/R`, and `vP:Parent + vResidual/R`. Their mean squares are `(MS_P, MS_Parent, MS_P:Parent)/R`, with df `(n-1, k-1, (n-1)*(k-1))`, where `k` is the source parent count. They have the same independent normal-model sampling law as a one-facet G-study. Reuse the F/GCI calculations on this triplet and vary the parent count; no raw-data averaging or refitting is performed by `confint()`. For G, the residual-MS term in Wong and McGraw's (1999, p.277, equation 2.1.3) vanishes when the source/future child-count ratio is one. The GCI route is a reduction to the existing one-facet procedure, not a nested GCI attributed to that paper. G remains exact under the positive-error normal model; Phi retains approximate coverage and Monte Carlo diagnostics.

Changing the child count alters these effective components, so this coefficient reduction no longer applies. Individual SEM intervals still support changed child counts. For SEM differences, plans need only keep the same future child count as each other, which may differ from `R`: their common error SD scales by the inverse square root of the parent count. The base interval may then use signed-sum MLS, retaining its approximate coverage. Comparisons changing both nested counts are not supported by this default. Explicit `method = "mean_square_region"` comparisons permit both counts to change using the enclosure below. `ReferenceCount` in the reduction counts parents; general region comparisons instead retain all reference counts in `attr(result, "reference")$counts`. `nesting_context` retains children per parent and the original source design.

General nested coefficient calculations

Both counts can change with explicit `"approximate_f"` or `"mean_square_region"`. Use the five components in order Person, parent, child(parent), Person:parent, residual. For future child/parent counts `R,T`, relative error weights are `(0,0,0,1/T,1/(R*T))`; absolute error adds `(0,1/T,1/(R*T),0,0)`. The mean-square region has five independent mean squares rather than seven.

For the F calculation let `n,r,t` be source persons, children per parent and parents, and set `m = r/R`, `h = t/T`. The statistic is `F(rho) = (1-rho)*MS_P/D(rho)`. For G, `D = (1-rho+rho*h)*MS_P:parent + rho*h*(m-1)*MS_residual`. For Phi, the coefficients of `MS_parent, MS_child(parent), MS_P:parent, MS_residual` are respectively `rho*h/n`, `rho*h*(m-1)/n`, `1-rho+rho*h*(1-1/n)` and `rho*h*(m-1)*(1-1/n)`. These follow by substituting the expected mean squares into the population coefficient, then isolating `MS_P`.

This agrees with Wong and McGraw's (1999, p.277) relative statistic. Their printed absolute formulas on p.278 omit the `-rho*h/n` term in the Person:parent coefficient and `+rho*h/n` in the residual coefficient. The implementation uses the expected-mean-square derivation above, checked against the component target and the unchanged-child one-facet reduction, rather than transcribing those printed formulas or their divided endpoints. No author-confirmed correction is claimed. Freeze the Satterthwaite df at the unrestricted ANOVA coefficient and invert the F inequalities, as for the crossed case. Signed denominator terms, negative components and invalid moments remain substantive limitations, not reasons to clip estimates.

A bounded normal-model evaluation reused 2,000 mean-square samples for each of three nested sources: 10 persons/2 children/3 parents, 60/4/8, and a 10/2/3 boundary model with only resid-

ual variance. Each had three future plans, with decreased, unchanged and increased child/parent counts. In the first two sources, nominal 95% approximate-F coverage was 93.55–96.45%, with 98.9–100% availability. In the boundary source projected to 4 children and 6 parents, coverage and availability both fell to 62.85% for G and 61.95% for Phi. Coverage retains unavailable replications in its denominator. A separate 200-sample region check gave 99.5–100% availability; some median coefficient widths exceeded .99. These limited results do not qualify either method as a generally useful automatic default.

Mean-square region calculation

For `J` independent normal-model ANOVA strata, each `df[j]*MS[j]/lambda[j]` has a chi-square distribution. Use marginal confidence level `level^(1/J)` for every expected mean square `lambda[j]`. Independence makes the joint rectangular region's coverage `level`. Intersect it with `lambda = L * theta`, where `L` is the expected-mean-square map and the scalar variance components `theta` are nonnegative. This retains the true component vector whenever the original region does.

Minimize and maximize `theta[Person]/(theta[Person] + error' theta)` over that common region for G/Phi. Positive residual limits and positive future counts keep this denominator positive. The change of variables `z = theta/denominator`, `t = 1/denominator` converts each endpoint problem to a linear program. SEM instead uses the extrema of the linear error variance, followed by square roots. The component scaling used internally changes coordinates, not the feasible set or target.

These projections can only increase coverage relative to the joint event for one prespecified score/composite. This does not guarantee narrow intervals. Empty intersections count as noncoverage in repeated sampling; coverage among only the returned intervals is a different quantity. The theoretical statement presumes correct projections; solver failures and finite-precision limitations remain separate availability conditions.

Extremely small confidence levels can make F quantiles or chi-square region limits numerically indistinguishable. The approximate-F calculation requires separation of log-F quantiles greater than `64 * .Machine$double.eps * max(1, abs(log_quantiles))`. A region requires positive bounds with relative width `(upper-lower)/upper` greater than `64 * .Machine$double.eps` for each expected mean square before optimization. Missing limits and `Status` explain failures; `diagnostics` retains F quantiles and resolution, or normalized `mean_square_bounds`, `relative_width`, `region_resolution` and the per-stratum `tail` probability. A SEM transformation that loses a nonzero endpoint or positive width is also unavailable. These checks retain the original point estimates and do not impose a minimum observation count.

Differences from a shared mean-square region

With `reference` and `method = "mean_square_region"`, project the **same** simultaneous source region onto the reference and comparison targets. If their intervals are `[L_a,U_a]` and `[L_b,U_b]`, report the enclosure `[L_b-U_a,U_b-L_a]`. The coverage justification is their common joint event, not independence or marginal 95% coverage. This is generally wider than the tight image of the difference over the region; it is not labeled exact.

Intersect with any sign restriction implied by componentwise ordering of the error weights. For example, if every error weight decreases, G/Phi cannot decrease and SEM cannot increase. For G/Phi, additionally intersect with the analytic population range. For positive reference/comparison errors `ea, eb`, the signed extremum of `U*(ea-eb)/((U+ea)*(U+eb))` occurs at `U=sqrt(ea*eb)`

and equals `tanh(log(ea/eb)/4)`. Across nonnegative components, `ea/eb` lies between the smallest and largest active error-weight ratios `a[j]/b[j]`. Transform those limits and include zero to obtain `FeasibleLower/Upper`. These are parameter-space limits, not extra source information. Their intersection preserves the common-region coverage argument, including when an extremum is approached only as a limit.

Equal error weights give `[0,0]` only when the region projections are available. Preserve missing intervals and diagnostics even for identical plans. The common region event gives coverage of at least `level` jointly over the returned comparisons for one fixed score/composite under positive mean-square expectations and independent normal effects. It gives no distribution-robust guarantee, simultaneous guarantee across selected scores, or tightness guarantee. The returned `individual_intervals` and `reference` retain the projections. Original ANOVA/MINQUE points remain unchanged, even if outside the population range implied by nonnegative components.

Point estimates and intervals at boundaries

`PointStatus` describes the reported point estimate; `Status` describes the interval calculation. A negative ANOVA estimate of Person variance can make the point coefficient unavailable while a confidence interval still exists. Inspect both statuses; do not remove such an interval merely because `Estimate` is `NA`. An empty confidence set and a numerical failure are different outcomes, and neither means zero uncertainty.

A wide interval is retained: it may reflect limited information about the sampled persons, raters or tasks, as well as a conservative method. More numerical draws cannot supply information missing from the rating design. Clipping negative variance estimates or rejecting simulation draws changes a statistical procedure and does not by itself establish valid coverage.

Choosing a two-facet method

The approximate-F, mean-square-region, score and REML alternatives remain explicit choices alongside the default MLS method. Their coverage theory, precision and finite-sample evidence differ; see the method-specific sections. `Available` describes whether endpoints were computed; `Validation` and `NegativeComponent` describe separate limitations. A narrower interval is not automatically better. Select methods before inspecting which is most favorable, and retain unavailable rows when comparing performance.

- Wong and McGraw (1999, pp. 275–276) give approximate F intervals for a fully crossed Person-by-two-facet random design. Both G and Phi use estimated Satterthwaite degrees of freedom. The authors advise against applying the method when variance-component estimates are negative. `method = "approximate_f"` implements the crossed-design calculation and the independently derived nested version above. Calculable intervals with negative components are retained as an experimental extension beyond that advice, with the flag set. Small normal-model evaluations found limitations in coverage, interval width and availability, particularly at zero components and when future facet counts change. See the G-theory vignette for conditions and results.
- `method = "mean_square_region"` projects a simultaneous chi-square confidence region for the seven crossed or five nested independent expected mean squares onto nonnegative variance components. It is a conservative comparison based on normal ANOVA distribution theory, not a method attributed to Wong and McGraw. It can return almost the whole `[0,1]` range. It requires lpSolve; zero mean squares, infeasibility and numerical failures remain explicit.

- `method = "reml_profile"` instead profiles the G/Phi target in a normal restricted likelihood, with approximate chi-square calibration. It also accepts identifiable incomplete/unequal sources. See the REML section above for the separate center, diagnostics and boundary limitations.

Use `plot(result)` for a forest comparison and `plot_data()` for its notes.

- For balanced crossed G, `method = "score"` uses the extended-nuisance score construction above. Its treatment of nuisance component boundaries differs from the nonnegative REML profile; neither method is uniformly superior in the available finite-sample checks. Phi is unsupported.

To show one method along a planning curve, use `plot(d, metric = "Phi", intervals = result)` with matching score/composite selection; see `plot.mfrm_multivariate_d_study()`. Agreement of methods on one dataset does not establish robustness or frequentist coverage. The G-theory vignette includes small evaluation results with their denominators and Monte Carlo uncertainty.

Delta coefficient SEs and intervals

Use `method = "delta"` to quantify uncertainty in the original G/Phi coefficient without fitting a different estimator. Let U be estimated universe variance and $E = a^T\hat{\theta}$ the relative or absolute error variance for the selected plan, where $\hat{\theta}$ contains the original component estimates. For $\hat{\rho} = U/(U+E)$, the gradient is $g = \{(1-\hat{\rho})e_U - \hat{\rho}a\}/(U+E)$, where e_U selects the universe component. The estimated sampling variance is $g^T\hat{C}g$, with $\hat{C}$ from `vcov.mfrm_multivariate_gstudy()`. The full covariance matters: estimated universe and error variances are not independent. Algebra uses normalized component scales to preserve invariance to score units and nonzero multipliers of fixed composite weights.

Balanced scalar ANOVA uses `A %*% diag(2*MS^2/(df+2)) %*% t(A)` for $\hat{C}$. This agrees with the random crossed derivation of Almehrizi (2021, equations 21–25). Nested and incomplete sources use the same differentiable ratio with their own design-specific component covariance. Those extensions follow the delta calculation; they were not evaluated in that paper. Its simulations assess SE accuracy, not Wald interval coverage. The covariance correction does not make a nonlinear coefficient SE unbiased. A normal-reference justification also requires growing information for the components that affect the target; more source Persons alone need not suffice. For example, with a fixed number of random raters and positive rater variance, increasing Persons does not consistently estimate the rater-population variance entering Phi. The rater mean-square degrees of freedom stay fixed. This limitation differs from the exact F result for one-facet G, which eliminates that nuisance component under its stated normal model.

`SE` is coefficient sampling uncertainty, not measurement SEM. Bounds are `Estimate +/- qnorm((1 + level)/2) * SE`. They are returned without clipping to `[0,1]`, even when wide, and summaries, forests and ribbons retain them. `OutsideParameterSpace` and `AtBoundary` identify limitations of the normal approximation; they do not mean the population coefficient can exceed its parameter space. Small facet pools, weak information, boundaries and nonnormal effects can impair coverage. Increasing future plan counts does not increase the source study's information. Intervals are pointwise for prespecified plans and weights, not simultaneous or selection-adjusted.

The original point must be defined. A nonfinite or negative projected sampling variance retains its status and does not supply an SE or interval. Zero first-order variance retains `SE = 0`, but does not justify a degenerate normal-reference interval. Positive SEs are also retained when the requested interval width rounds to zero; the limits are then `NA` with a numeric-resolution status. An indefinite estimated component covariance is flagged; a finite positive projection still returns an explicit

approximation. Nothing is repaired to obtain a preferred interval. For paired two-crossed- facet differences, `mfrm_multivariate_d_compare()` retains its separately documented plug-in covariance, rather than the corrected covariance used here.

A limited normal-model evaluation reuses 2,000 independent samples per source below, with a fixed outcome-independent omission pattern retaining about 8/11 of the cells. Person and residual variances are 1; all other component variances are 0.2. Future plans are complete balanced ratings at the source pool counts; nested child counts are per parent. Entries are percentages. Coverage uses **all 2,000 attempts**, counting unavailable intervals as not covering; return rates are shown separately.

Source persons x measurement counts	G returned	G covered	Phi returned	Phi covered
One facet: 10 x 3	92.00	83.55	92.60	83.60
One facet: 60 x 8	100.00	95.65	100.00	95.25
Crossed: 10 x 2 x 3	78.95	78.30	86.45	82.90
Crossed: 30 x 4 x 5	100.00	93.90	100.00	91.10
Nested: 10 x 2 within 3	97.30	86.40	97.85	87.10
Nested: 40 x 3 within 6	100.00	94.55	100.00	91.75

Coverage Monte Carlo SEs range from 0.46 to 0.92 percentage points. Conditional coverage can differ materially: crossed 10 x 2 x 3 G covers in 99.18% of its 1,579 returned intervals, despite only 78.30% across all attempts. These conditions demonstrate why the approximation is explicit, not a generally calibrated default; they do not establish a sample-size cutoff or robustness to nonnormality or selective omissions.

Other methods and interpretation limits

Data-level bootstrap intervals are not yet options in this function. Incomplete-source G/Phi can use explicit `"delta"` or `"reml_profile"`, with different estimators and approximations. One-facet G also supports explicit `"spectral_f"` when residual degrees of freedom are positive. These routes do not supply a general interval for every original MINQUE component or validate all other incomplete-source targets.

Parameter-distribution Monte Carlo and data-generating bootstrap methods require a specified estimator, joint component covariance or refitting procedure, and handling of boundaries and failed calculations. A result for another estimator or sampling design cannot be transferred solely because its coefficient is also named G or Phi.

A fixed set of tasks scored by every sampled rater is a different target from generalization to new tasks. Representing those tasks as fixed score columns with prespecified weights can use the supported one-random-facet route; see `mfrm_g_study()`. This corresponds to the crossed fixed-facet G construction of Zhou et al. (2011). Merely holding a random task count constant does not make that facet fixed.

An unsupported method/design combination does not imply that the coefficient has no uncertainty or that interval estimation is mathematically impossible. For supported calculations, inspect `Status` and the returned notes: a computable approximation can carry limitations, whereas an undefined ratio, empty confidence set or failed numerical calculation cannot be repaired by adding a warning. Preserve available estimates and diagnostics; do not substitute measurement SEM, combine separate component endpoints, or drop a random facet to obtain an interval. The paired two-facet intervals

from mfrm_multivariate_d_compare() concern differences between plans and have their own approximation limits. They do not supply an individual-plan coefficient interval.

Two-facet calculations

Write the seven ANOVA component estimates as theta_hat = A * MS, in order Person, facet 1, facet 2, Person:1, Person:2, 1:2, residual. For future counts r, t, relative error weights are (0,0,0,1/r,1/t,0,1/(r*t)); absolute error adds (0,1/r,1/t,0,0,1/(r*t),0). The population target is rho = theta_P / (theta_P + e' theta), with nonnegative components and positive denominator. Fixed score weights are applied before this calculation.

For approximate F, let a be the Person row of A and d = e' A. Excluding the Person mean square, set b0 = -a/a_P and b1 = (a+d)/a_P. The denominator is D(rho) = sum((b0+rho*b1)*MS). Estimate its Satterthwaite degrees of freedom at the unrestricted ANOVA rho_hat: nu = D(rho_hat)^2 / sum(((b0+rho_hat*b1)*MS)^2/df). Freeze nu and invert F(rho) = (1-rho)*MS_P/D(rho) between the F quantiles with Person df and nu. Intersect the original linear inequalities, including positive D, with [0,1]. Dividing endpoint formulas and clipping can be wrong when future counts change the divisor sign. diagnostics retains the unrestricted roots and df; roots alone are not confidence limits.

For the mean-square region, let L = solve(A) and lambda = L*theta. Each df*MS/lambda has an independent chi-square distribution. Set each central mean-square interval's coverage to level^(1/K), where K is seven for crossed or five for nested designs, making joint coverage exactly level under positive expectations and normal effects. Intersect this rectangular region for lambda with theta >= 0, then minimize and maximize rho by a Charnes-Cooper linear-fractional transformation. Feasibility and objective residuals are checked after optimization. Its image has coverage at least level, jointly over future plans for this fixed score; it is often conservative. This guarantee does not extend to nonnormal effects or data-selected weights, or establish numerical validity of a failed solve. Negative ANOVA estimates are retained in the source and are not individually clipped before either method.

Calculation of G

Let k be the observed number of facet levels and m a future plan's count. Person and residual mean squares MS_P and MS_E are independent under the stated model, with degrees of freedom n - 1 and (n - 1)*(k - 1). Their expectations are k * sigma_P^2 + sigma_E^2 and sigma_E^2. Thus, writing R = 1 + k * sigma_P^2 / sigma_E^2, (MS_P / MS_E) / R follows that F distribution. Divide the observed ratio by the upper and lower F quantiles to obtain the interval for R. Intersect it with R >= 1, then transform its endpoints using G_m = m * (R - 1) / (m * (R - 1) + k). These steps follow directly from the independent mean-square distributions in Mondal et al. (2025, Section 2, Table 1); the transformation here expresses the one-facet G-theory target at the requested future count. At m = k, the usual estimate is 1 - MS_E / MS_P.

Intersect **before** transforming: outside R >= 1, the transformation can have a pole when m > k. Clipping already transformed endpoints to [0, 1] would therefore be incorrect. RawRatioLower and RawRatioUpper retain unrestricted ratio endpoints; they are not G endpoints.

An interval reaching R = 1 includes G = 0. If the ratio interval is entirely below 1, the valid intersection is empty and both G endpoints are NA, with Status = "Empty parameter-space interval". Negative Person estimates are retained in the source; Estimate and PointStatus follow the existing D-study rules. A point estimate may be unavailable while a nonempty interval still exists. Zero residual mean square leaves the interval unavailable, even if the point estimate is 1: the F pivot's

positive-variance condition cannot be checked from a zero denominator. If a positive width rounds away on the G scale, endpoints are unavailable with a numeric-resolution status. The retained log-ratio bounds can still support representable plan-difference intervals. If the base log-ratio endpoints themselves coincide through rounding (for example at an extremely small `level`), both G and plan-difference intervals are unavailable. This numerical failure differs from a positive-width base set whose valid intersection is the single boundary value G = 0.

All rows use the same ratio interval for the selected score. Under this model their coverage events are identical; they are not independent intervals, and their overlap is not a test of a plan difference. This property does not provide simultaneous inference over different scores, adaptively chosen weights, other designs or other metrics. Existing D-study plots can display saved intervals through their `intervals` argument.

Differences in G between plans

Use `reference = 1` to ask how much G would change from the first plan. The plans use the same estimated components: subtracting independent marginal confidence limits or adding marginal variances is inappropriate. This method instead maps the **entire** confidence set for the common variance ratio to the G difference; it does not use a delta approximation.

Put `t = sigma_P^2 / sigma_E^2 = (R - 1)/k`. For reference count `a` and comparison count `b`, the difference is `h(t) = (b-a)*t / ((1+a*t)*(1+b*t))`. Its derivative has the sign of `(b-a)*(1-a*b*t^2)`. Therefore the range over the nonnegative ratio interval is obtained from both endpoints and, when included, the interior stationary point `t = 1/sqrt(a*b)`. Mapping only the endpoints would miss the extremum. At zero and at the limit `t -> Inf`, the difference is zero. Equal counts give zero throughout a nonempty confidence set. An empty base set or a zero residual mean square retains its unavailable status, including for repeated counts. A nonconstant image whose width or nonzero endpoint is lost to numerical rounding has missing limits with `Status` explaining the loss; its original point remains unchanged. This numerical rule also applies to shared Phi and REML coefficient-set images, with each method's own coverage scope.

Whenever the base F confidence set contains the true variance ratio, all its mapped difference intervals contain the corresponding true differences. Thus coverage is **at least** the requested level under the stated model, including jointly over the returned prespecified counts for this one fixed score/composite. It can be conservative because `h` is not one-to-one. This follows from confidence-set inclusion and the derivative above, not from a new asymptotic distribution. It does not supply inference for other scores, adaptive score weights or other facet designs. Phi uses the separate generalized construction below, with no exact nominal-coverage guarantee. The sign of the change from adding levels already follows from this model; the interval quantifies its size, not whether more levels are cost-effective. A point difference can be unavailable while its interval remains available.

Calculation of Phi

With `n` persons and `k` observed facet levels, write `Q_P`, `Q_F`, `Q_E` for `df * MS / X`, where the three `X` values are independent chi-square draws with degrees of freedom `n-1`, `k-1`, `(n-1)*(k-1)`, respectively. The single-rating agreement-ICC generalized pivot is `max(0, Q_P - Q_E) / (Q_P + (k/n)*Q_F + (k - 1 - k/n)*Q_E)`. This is Equation 11 of Mondal et al. (2025), including their correction to an earlier published denominator. Its central quantiles give the single-rating GCI. For a future count `m`, transform both endpoints by `m * rho / (1 + (m - 1) * rho)`, since `Phi_m = sigma_P^2 / (sigma_P^2 + (sigma_F^2 + sigma_E^2)/m)`. The observed `k`, not future `m`, determines the sampling distributions.

Truncation of the pivot's numerator at zero is part of this GCI method; it does not truncate the source component estimates or change the point estimate. A negative Person estimate can therefore have an unavailable point estimate and an available interval. A zero upper endpoint can also occur; `Status` and the proportion of zero pivotal draws flag this boundary result, which must not be read as proof of zero sampling uncertainty. Any zero mean square makes this implementation return an unavailable interval with `Status = "Degenerate mean squares"`.

The default 100000 draws follow the numerical recommendation discussed in Mondal et al. (Section 3.3). `attr(result, "monte_carlo")` contains draw/seed settings, RNG kind, the zero proportion, and separate 95% Monte Carlo bands for each endpoint, obtained from binomial order-statistic ranks. These bands describe numerical quantile error, not another confidence interval for Phi. They do not include uncertainty from violated model assumptions or approximate frequentist coverage. All future counts share the same sampled interval; rows are dependent and do not test plan differences. Fixed composites are supported, but adaptive weights and simultaneous inference over scores are not.

Differences in Phi between plans

With `parm = "Phi", reference = 1`, ask how much absolute dependability changes from the first plan. Write `t = sigma_P^2/(sigma_F^2+sigma_E^2)`. Then `Phi_m = m*t/(1+m*t)`, so the count-difference function `h(t)` and its interior extremum are the same as for G, with a different variance ratio. Transform the whole single-rating generalized interval by `t = rho/(1-rho)` and take the range of `h`. Zero and unit endpoints correspond to `t = 0` and its infinite limit. This set-image extension follows from the displayed algebra; it is not a new published difference pivot or the quantiles of differences calculated draw by draw.

Whenever the base generalized interval contains the true ratio, its image contains every returned true plan difference for this fixed score. Mapping can increase coverage through conservatism but does not make the base GCI's approximate frequentist coverage exact. Normal effects and prespecified counts/weights remain required. Do not infer the difference interval by subtracting two marginal confidence limits. Use `plot(result)` for a forest view on the difference scale; these results cannot be used as coefficient ribbons in `plot(d)`.

One pivotal sample supplies all comparisons. The `monte_carlo` attribute retains `base_endpoint_bands` and at least 95% **joint numerical** bands for both difference endpoints across the returned plans. Each base endpoint has a Bonferroni-protected 97.5% order-statistic band. Map the outer base interval and their common inner interval to bound the image endpoints; when the inner interval is empty, use the outer image as a coarse bound for each endpoint. These bands concern numerical quantile error, not the probability that a coefficient interval covers its population target. Degenerate/numerically unavailable base intervals remain unavailable even for identical counts. A base `Upper endpoint at zero` status is retained for unequal counts; it is not proof that the difference is precisely zero.

References

E, L., Hannig, J., & Iyer, H. (2008). Fiducial intervals for variance components in an unbalanced two-component normal mixed linear model. *Journal of the American Statistical Association, 103*(482), 854–865. doi:10.1198/016214508000000229. Sections 2 and 2.3, equations 3–7, give the independent quadratic forms and existing F-pivot family used by `"spectral_f"`; the fiducial method developed later is not implemented.

Shedden, M., & Ekvall, K. O. (2026). Score-based confidence intervals for variance-covariance parameters in linear mixed models. Preprint. doi:10.48550/arXiv.2610.04181. The ratio-target adaptation above is not a G-theory coverage result supplied by this paper.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. doi:10.18637/jss.v067.i01. Restricted likelihood computation.

Drton, M. (2009). Likelihood ratio tests and singularities. *The Annals of Statistics, 37*(2), 979–1012. doi:10.1214/07AOS571. General cautions about singular hypotheses; not a calibration theorem for the present crossed REML contrasts.

Crainiceanu, C. M., & Ruppert, D. (2004). Likelihood ratio tests in linear mixed models with one variance component. *Journal of the Royal Statistical Society: Series B, 66*(1), 165–185. doi:10.1111/j.14679868.2004.00438.x. Special-case boundary distributions; not a calibration result for general G/Phi targets.

Betebenner, D. W. (1998). *Improved confidence interval estimation for variance components and error variances in generalizability theory*. Paper presented at AERA, San Diego. ERIC ED423304. https://files.eric.ed.gov/fulltext/ED423304.pdf.

Hess, A., & Iyer, H. (2002). A SAS/IML macro for computation of confidence intervals for variance components of mixed models. *Proceedings of SUGI 27*, Paper 252-27, pp. 2–3. https://support.sas.com/resources/papers/proceedings/proceedings/sugi27/p252-27.pdf.

Ting, N., Burdick, R. K., Graybill, F. A., Jeyaratnam, S., & Lu, T.-F. C. (1990). Confidence intervals on linear combinations of variance components that are unrestricted in sign. *Journal of Statistical Computation and Simulation, 35*(3–4), 135–143. doi:10.1080/00949659008811240. The G/Phi null inversion and conservative completion above are mfrmr's adaptations, with their own limited finite-sample evaluation.

Graybill, F. A., & Wang, C.-M. (1980). Confidence intervals on nonnegative linear combinations of variances. *Journal of the American Statistical Association, 75*(372), 869–873.

Mondal, D., Candel, M. J. J. M., Cassese, A., & Vanbelle, S. (2025). Confidence intervals and sample size for the ICC in two-way ANOVA models. *Statistics in Medicine, 44*(10–12), e70106. doi:10.1002/sim.70106. Section 2 and Table 1 give the independent normal-theory mean squares. Its later comparison of agreement-ICC methods does not establish an exact Phi interval or a multifacet G interval. Section 3.3, Equation 11 and Appendix C give the GCI used for Phi here.

Tian, L., & Cappelleri, J. C. (2004). A new approach for interval estimation and hypothesis testing of a certain intraclass correlation coefficient: The generalized variable method. *Statistics in Medicine, 23*, 2125–2135. doi:10.1002/sim.1782.

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560. G-study and D-study targets.

Wong, S. P., & McGraw, K. O. (1999). Confidence intervals and F tests for intraclass correlations based on three-way random effects models. *Educational and Psychological Measurement, 59*, 270–288. doi:10.1177/00131649921969848. Crossed two-random-facet approximate F method, with the negative-component extension flagged above. Page 277 gives the nested relative-G pivot used in the unchanged-child-count reduction.

Zhou, H., Muellerleile, P., Ingram, D., & Wong, S. P. (2011). Confidence intervals and F tests for intraclass correlation coefficients based on three-way mixed effects models. *Journal of Educational and Behavioral Statistics, 36*, 638–671. doi:10.3102/1076998610381399. The fully crossed

fixed-facet relative-G target agrees with the fixed score-composite, one-random-facet representation described above.

Almehrizi, R. S. (2021). Standard errors of variance components, measurement errors and generalizability coefficients for crossed designs. *Journal of Educational Measurement, 58*, 179–210. doi:10.1111/jedm.12277. Coefficient SEs and their evaluation, not a validation of two-facet coefficient interval coverage.

See Also

mfrm_d_study(), confint.mfrm_multivariate_gstudy(), mfrm_multivariate_d_compare()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("confint.mfrm_multivariate_d_study", run.dontrun = TRUE)
## Not run:
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
d <- mfrm_d_study(g, data.frame(Tasks = c(6, 12)),
  weights = c(V = .5, W = .5))
ci <- confint(d, assumption = "normal")
ci[, c("Tasks", "Estimate", "Lower", "Upper", "Status")]
attr(ci, "notes")
# Optional normal approximation: SE describes the coefficient, not measurement SEM.
delta_ci <- confint(d, method = "delta", assumption = "normal")
delta_ci[, c("Tasks", "Estimate", "SE", "Lower", "Upper", "SEStatus")]
phi_ci <- confint(d, parm = "Phi", assumption = "normal", seed = 1)
phi_ci[, c("Tasks", "Estimate", "Lower", "Upper", "Status")]
attr(phi_ci, "monte_carlo")$endpoint_bands
# How much would G improve by changing from six to twelve tasks?
change <- confint(d, parm = "G", reference = 1, assumption = "normal")
change[, c("Tasks", "ReferenceCount", "Estimate", "Lower", "Upper", "Status")]
# The same planning question for absolute dependability; coverage is approximate.
phi_change <- confint(d, parm = "Phi", reference = 1, assumption = "normal")
phi_change[, c("Tasks", "ReferenceCount", "Estimate", "Lower", "Upper", "Status")]
attr(phi_change, "monte_carlo")$endpoint_bands
# Sampling uncertainty in measurement SEM, rather than in G or Phi.
sem_ci <- confint(d, parm = "AbsoluteSEM", assumption = "normal")
sem_ci[, c("Tasks", "Estimate", "Lower", "Upper", "Method", "Status")]
sem_change <- confint(d, parm = "RelativeSEM", reference = 1,
                      assumption = "normal")
sem_change[, c("Tasks", "Estimate", "Lower", "Upper", "Status")]
sem_plot <- plot(d, type = "sem", metric = "AbsoluteSEM", intervals = TRUE,
                 assumption = "normal", draw = FALSE)
plot_data(sem_plot, "notes")
# Different raters serve each task; retain three raters per task.
data("mfrmr_example_gtheory_nested", package = "mfrmr")
gn <- mfrm_g_study(mfrmr_example_gtheory_nested,
  c("Content", "Organization"), nesting = c(Rater = "Task"))
dn <- mfrm_d_study(gn,
  data.frame(Raters = 3, Tasks = c(6, 12)))
nested_change <- confint(dn, parm = "G", reference = 1, assumption = "normal")
```

```
nested_change[, c("Raters", "Tasks", "ReferenceCount", "Estimate", "Lower", "Upper")]
attr(nested_change, "nesting_context")
# Incomplete one-facet ratings: relative SEM without likelihood fitting.
observed <- tasks[(tasks$Person + tasks$Task) %% 3 != 0, ]
sparse_g <- mfrm_g_study(observed, "V", rater = NULL, method = "minque0")
sparse_d <- mfrm_d_study(sparse_g, data.frame(Tasks = c(6, 12)))
residual_sem <- confint(sparse_d, parm = "RelativeSEM",
  method = "residual_contrast", assumption = "normal")
residual_sem[, c("Tasks", "Estimate", "IntervalCenter", "Lower", "Upper", "Status")]
residual_change <- confint(sparse_d, parm = "RelativeSEM", reference = 1,
  method = "residual_contrast", assumption = "normal")
residual_change[, c("Tasks", "Estimate", "IntervalCenter", "Lower", "Upper")]
# Relative G also uses Person variation; normal Person effects are required.
spectral_g <- confint(sparse_d, method = "spectral_f", assumption = "normal")
spectral_g[, c("Tasks", "Estimate", "Lower", "Upper", "Status")]
spectral_change <- confint(sparse_d, method = "spectral_f", reference = 1,
  assumption = "normal")
spectral_change[, c("Tasks", "Estimate", "Lower", "Upper")]
# A separate experimental REML model can address G/Phi or absolute SEM.
if (requireNamespace("lme4", quietly = TRUE) && requireNamespace("minqa", quietly = TRUE)) {
  profile_ci <- confint(sparse_d, method = "reml_profile", assumption = "normal")
  profile_ci[, c("Tasks", "Estimate", "IntervalCenter", "Lower", "Upper", "Validation")]
  attr(profile_ci, "notes")
  # Uncertainty in absolute measurement SEM, in score units.
  profile_sem <- confint(sparse_d, parm = "AbsoluteSEM",
                         method = "reml_profile", assumption = "normal")
  plot(sparse_d, type = "sem", metric = "AbsoluteSEM", intervals = profile_sem,
       draw = FALSE)
  # How much does doubling the task count change G?
  profile_change <- confint(sparse_d, reference = 1,
    method = "reml_profile", assumption = "normal")
  profile_change[, c("Tasks", "Estimate", "IntervalCenter", "Lower", "Upper")]
  attr(profile_change, "base_interval")
  plot(profile_change, draw = FALSE)
}
# Start with the default two-crossed-facet approximation on bundled data.
data("mfrmr_example_gtheory_crossed", package = "mfrmr")
crossed <- mfrm_g_study(mfrmr_example_gtheory_crossed,
  c("Content", "Organization"))
future <- mfrm_d_study(crossed,
  data.frame(Raters = c(2, 4), Tasks = c(6, 12)),
  weights = c(Content = .5, Organization = .5))
crossed_ci <- confint(future, parm = "Phi", assumption = "normal")
crossed_ci[, c("Raters", "Tasks", "Estimate", "Lower", "Upper",
               "FormulaComplete", "Status")]
attr(crossed_ci, "notes")
# The score method remains an explicit experimental comparison for G.
score_ci <- confint(future, method = "score", assumption = "normal")
score_ci[, c("Scenario", "Estimate", "Lower", "Upper", "Empty", "Status")]
# Vary task counts within each rater count to obtain planning-curve ribbons.
curve_plans <- mfrm_d_study(crossed,
  expand.grid(Raters = c(2, 4), Tasks = c(3, 6, 12)),
  weights = c(Content = .5, Organization = .5))
```

```
score_curve <- plot(curve_plans, metric = "G", intervals = TRUE,
  method = "score", assumption = "normal", draw = FALSE)
plot_data(score_curve, "notes")
if (requireNamespace("lpSolve", quietly = TRUE)) {
  comparison <- confint(future, parm = "Phi", assumption = "normal",
    method = c("mls", "approximate_f", "mean_square_region"))
  comparison[, c("Scenario", "Method", "Lower", "Upper", "Status")]
  figure <- plot(comparison, draw = FALSE)
  plot_data(figure, "notes")
  if (requireNamespace("ggplot2", quietly = TRUE)) as_ggplot(figure)
  # General nested comparison: both counts change, with eighteen ratings.
  general_nested <- mfrm_d_study(gn,
    data.frame(Raters = c(3, 2, 6), Tasks = c(6, 9, 3)))
  nested_region <- confint(general_nested, parm = "Phi", reference = 1,
    method = "mean_square_region", assumption = "normal")
  nested_region[, c("Raters", "Tasks", "Estimate", "Lower", "Upper", "Status")]
  attr(nested_region, "reference")$counts
}

## End(Not run)
```

`confint.mfrm_multivariate_gstudy` *Sampling uncertainty in G-study variance and covariance components*

Description

Add standard errors and confidence intervals to the variance components of one observed score, one prespecified weighted composite, or the covariance between two original scores. The default uses a complete balanced ANOVA G-study without refitting. Explicit experimental REML profiles also accept identifiable incomplete sources, fitting a separate normal model. Explicit residual contrasts provide residual-only intervals from retained used ratings, including incomplete sources, without likelihood fitting.

Usage

```
## S3 method for class 'mfrm_multivariate_gstudy'
confint(
  object,
  parm = NULL,
  level = 0.95,
  score = NULL,
  weights = NULL,
  assumption,
  method = "auto",
  draws = 20000L,
  seed = 20261016L,
  ...
)
```

Arguments

object	A result from mfrm_g_study() fitted with method = "anova". One or two crossed random facets, or the supported balanced nested design, are allowed. With method = "reml_profile", variance and two-score covariance intervals also accept incomplete or unequal MINQUE(0) sources with their raw used ratings retained. method = "residual_contrast" accepts these sources for Residual only.
parm	Component names or integer positions to return; NULL selects all components. Names are those in object$components.
level	Pointwise confidence level strictly between zero and one.
score	One original score name for variances (the default is the first score), or exactly two distinct original names for their covariance. With two names, explicitly select a covariance method below.
weights	Optional named numeric vector specifying one composite. Name every score once, with finite weights and at least one nonzero value. Weights are used as supplied, including signed weights for a difference score. Do not supply both score and weights.
assumption	Required: "normal" asserts independent normal random effects with constant component variances for the selected score or composite. Persons and the selected measurement facets are random. The function does not test these assumptions.
method	"auto" or "mls" for a single score/composite from balanced ANOVA. Explicit "reml_profile" computes approximate component intervals from raw used ratings, for balanced or identifiable incomplete sources; it requires optional lme4 and minqa. It retains original points and reports a separate REML center, but does not supply a sampling SE. Its chi-square calibration is experimental and it is never chosen automatically. For two-score covariance, explicitly choose "gci" (generalized interval with approximate coverage), "regression_region" (conservative normal-model enclosure), or "reml_profile" (experimental joint normal-model profile). The first two require balanced ANOVA. No covariance method is silently selected or substituted after failure. "residual_contrast" supplies only the Residual interval: exact chi-square for one score/composite, or a conservative enclosure for two-score covariance, under the normal residual model. It requires raw used ratings, permits incomplete sources, reports a separate IntervalCenter, and does not supply the original estimator's SE. Other components retain their estimates with unavailable intervals. See *Residual contrasts* below.
draws	Number of pivotal draws for covariance method = "gci"; default 20000. At least 1000 and at least 100 expected draws in each tail are required. Inspect the numerical endpoint bands, not just this minimum.
seed	Nonnegative integer seed for covariance GCI; the caller's random state is restored. draws and seed are rejected for other methods.
...	Reserved; additional arguments are rejected.

Details

For a source-by-target choice table, see *Choose the G-theory target and uncertainty* in mfrmr_methods_reference. Component intervals, G/Phi intervals, measurement SEM and paired plan differences have distinct targets.

A component estimates how much scores vary because of persons, conditions, or their interactions. Its SE describes sampling uncertainty in that estimate. Both the component and its SE have squared score units. This SE is different from a D-study's measurement SEM, which is in score units.

Default balanced intervals use the modified large-sample (MLS) method of Ting et al. (1990) for signed linear combinations of independent ANOVA mean squares, as detailed by Hess and Iyer (2002, pp. 2–3). The residual component, which is a single mean square, has the usual chi-square interval. Other component intervals are approximate, asymmetric and pointwise, not simultaneous. Normal effects, a balanced design and prespecified score weights matter: the method is not robust to arbitrary score distributions or weights chosen after inspecting the data. A numeric ordinal score does not by itself justify the normal model. At very low requested confidence levels, squared chi-square corrections can lose their usual direction; a small `level` does not guarantee narrow or well-calibrated MLS limits.

Negative component estimates are retained. `RawLower` and `RawUpper` are the unrestricted MLS endpoints. `Lower` and `Upper` intersect that interval with the nonnegative variance parameter space. A negative lower endpoint therefore becomes zero, but the point estimate is not changed. If the entire raw interval is below zero, the intersection is empty: both bounds are `NA` with `Status = "Empty nonnegative interval"`, while the raw endpoints remain visible. It is not replaced by an interval of zero width. Inspect the design, assumptions and raw estimates in that case.

For MLS, `Status = "Includes zero"` denotes a nonempty interval containing zero; `"Available"` denotes a strictly positive interval. If all mean squares contributing to a component are zero, its estimated SE is zero but bounds are unavailable (`"Degenerate mean squares"`); this is not evidence of precise population knowledge. `"Numerical interval failure"` retains the estimate and SE when the MLS width cannot be evaluated reliably. A positive interval whose endpoints coincide after rounding instead has `Status = "Variance interval below numeric resolution"` (or the corresponding component-scale status), with missing `Lower`/`Upper`. This can occur at an extremely small requested confidence level. Positive endpoints that overflow or underflow also remain unavailable. Raw endpoints are retained for diagnosis; equal raw endpoints in such a row do not establish zero uncertainty.

This method estimates variances of the selected score, including all covariances needed for a fixed composite. The separate two-score covariance calculation is described below. Neither supplies G/Phi or plan-difference intervals. Explicit REML profiles and residual contrasts accept incomplete sources, with the different scopes described below. `mfrm_multivariate_d_compare()` provides a separate approximation for certain paired plan differences. Use `plot(g, intervals = TRUE, assumption = "normal")` to calculate and show these component intervals directly, with `score` or `weights` selecting the desired score. Its returned data retain the complete interval tables.

Value

A data frame with `Source`, selected `Score`, raw `Estimate`, `SE`, nonnegative-space `Lower` and `Upper`, unrestricted `RawLower` and `RawUpper`, `Level`, `Method`, and `Status`. Attributes `weights`, `kind`, `assumption`, `source_design`, `source_rows`, and `notes` retain context. Read `attr(result, "notes")` and save the full result with `saveRDS()`; a CSV alone does not retain weights and assumptions. The original G-study is unchanged. Stored mean products, degrees of freedom and design metadata are

required for MLS; raw ratings are not needed for that calculation. REML profiles instead require raw used ratings and retain `PointStatus`, `IntervalCenter`, `SEStatus`, `CandidateLower/Upper`, `MethodCode`, `Available`, `Validation`, and portable `reml_source` and `diagnostics` attributes. Diagnostics are named in the selected component order, in standardized-score units; multiply by `reml_source$score_scale` twice for a variance or by the two separate scale entries for a covariance. No live fitting environment is saved. `PointStatus` is for scalar variances. Two-score results additionally include `Score1`, `Score2`, `SEStatus`, `score_pair` and method-specific `diagnostics`. Balanced GCI/region results retain normalized `mean_products` and optional `monte_carlo` attributes; their `RawLower/RawUpper` equal the signed `Lower/Upper`. REML results instead retain the separate center, candidates and profiles. Covariance intervals have no nonnegative-space intersection. Residual-contrast results retain `IntervalCenter`, `SEStatus`, `MethodCode`, `Available` and `Validation`, plus a `residual_projection` attribute with residual degrees of freedom, rank, scale and numerical checks. They require retained ratings but save no live fitting environment or projection matrix.

Residual contrasts

Use `method = "residual_contrast"` to ask how uncertain the combined residual variance or covariance is. It supports the same crossed and nested source designs, including identifiable incomplete MINQUE(0) sources, when the observed assignment leaves residual degrees of freedom. It does not supply intervals for Person, facet or lower-order interaction components or G/Phi. For a one-facet D-study, `confint.mfrm_multivariate_d_study()` can map this interval to relative SEM and its plan differences; the component result itself remains in squared score units. With one rating per cell, the residual combines the highest interaction and observation error; it cannot separate them.

Let `Z` contain all nonresidual incidence columns, including interactions and the mean. With `nu = n - rank(Z) > 0`, choose an orthonormal basis `H` satisfying `H'Z = 0`. Under independent homoskedastic normal residuals, `H'Y = H'E` and `Y'HH'Y` has a Wishart distribution with `nu` degrees of freedom and residual covariance `Sigma_E`. Thus for prespecified weights `w`, the residual sum of squares divided by `w'Sigma_E w` is chi-square with `nu` degrees of freedom. Its quantiles give exact pointwise limits for a positive residual variance. The calculation uses QR residuals, checks rank against singular values and verifies orthogonality, without forming an observation-by-observation projection matrix. This applies the standard multivariate regression residual law to the G-study incidence model.

For two scores, the same Wishart block feeds the conservative regression enclosure described in *Between-score covariance elements* below. One-degree-of-freedom or singular pairs instead use marginal variance bounds and Cauchy-Schwarz. Signed covariance bounds are retained. Coverage is conservative under the model, not exact equal-tail coverage, and the enclosure can be very wide. A sparse crossed arrangement can leave few residual contrasts despite many ratings. Inspect `attr(ci, "residual_projection")$df` and the limits.

The assignment is treated as fixed; omissions must preserve the residual model. This does not correct selective missingness, heteroskedasticity or dependence between remaining residuals. Normal residuals are essential. Claims are pointwise for prespecified scores/weights, not simultaneous or adjusted for selecting a result after looking at the data.

`Estimate` retains the original ANOVA/MINQUE component. `IntervalCenter` is the separate residual-contrast estimate; the two agree in balanced ANOVA but can differ for incomplete sources. The original point can lie outside the interval. `SE = NA` does not mean zero uncertainty: this method does not calculate the original estimator's SE. The scalar chi-square calculation shares the

numerical-width checks above; an unresolved interval retains its original point and projected center. Use vcov.mfrm_multivariate_gstudy() for its separately qualified sampling covariance. Both the component and these limits have squared score units. Zero residual degrees of freedom, numerically unresolved residual variation or failed projection checks give unavailable limits with explicit Status. Unsupported component rows remain visible. No unavailable row is a zero-width interval. Raw ratings and stored components are checked for agreement; recreating a modified G-study is preferable to editing its fields.

REML component profiles

For example, uncertainty in the Task variance asks how strongly randomly sampled tasks differ in their average scores. method = "reml_profile" fits a separate normal random-effects model to the observed assignment: one common mean and independent Person, facet, interaction and residual effects. Omissions must be noninformative conditional on that assignment. It does not correct selective missingness or replace the original MINQUE estimator. Estimate retains the original component, including negative values; IntervalCenter is the REML estimate of the same population variance. The original point can be outside the interval. Neither component matrices nor original estimates are clipped or repaired.

Write theta = s2 * v, with residual ratio one. For target component j, constrain tau = theta[j] by setting s2 = tau / v[j]. Restore residual scale to the profiled restricted criterion using D(v,s2) = D_profile(v) + (n-1)*(log(s2/s2_hat(v)) + s2_hat(v)/s2 - 1). Optimize the remaining ratios. At a zero nonresidual target, set that component exactly to zero and profile the others; the residual remains strictly positive. The interval uses D - D_min <= qchisq(level, 1). See the REML section of confint.mfrm_multivariate_d_study() for the underlying normal model, numerical checks and relationship to ordinary ML profile intervals. Bates et al. supply the REML criterion, not a finite-sample validation of this particular G-theory procedure.

Search outward from the REML center and refine the first cutoff crossing in each direction. This returns a numerical interval around that center; it does not enumerate remote disconnected accepted regions. Multiple starts, a second optimizer, and finite-difference checks including zero boundaries do not certify global optima or rule out narrower undetected crossings. Unresolved required evaluations, roots or tails leave accepted bounds unavailable, retaining any resolved CandidateLower/Upper separately. Reaching a search limit does not justify an infinite bound. A constant selected score is degenerate, not evidence of zero-width uncertainty. Weighted scores are scaled together with their weights before fitting; reciprocal changes of score units and weights preserve the selected composite. At very small confidence levels, the likelihood-ratio cutoff may be too small to distinguish from numerical error. Such intervals have missing bounds and an explicit Status, retaining the original estimate and representable REML center. For each component, diagnostics[[i]]$cutoff records the cutoff, endpoint tolerance and deviance resolution. Nonzero bounds lost when restoring score units are also unavailable; normalized profile diagnostics remain available. These checks also apply to joint covariance profiles. They are numerical restrictions, not minimum sample sizes or evidence of zero uncertainty.

Available describes numerical completion, not statistical coverage. A limited check reused 40 samples per condition: component true-value acceptance was 92.5–100% with positive components and 90–100% for checked zero components, counting numerical failures as misses. The Monte Carlo SE can reach 4.7 percentage points. These are coarse feasibility results, not coverage of full endpoint searches or support for a generally recommended method. The chi-square cutoff is asymptotic; a universal boundary-mixture correction is not assumed. Intervals are pointwise for prespecified scores and components, neither simultaneous nor distribution robust.

SE = NA and SEStatus = "Not supplied by profile inversion" mean that this method did not calculate a sampling SE. The interval width must not be converted into one by dividing by a normal critical value. RawLower and RawUpper are inapplicable MLS fields and remain NA; they are not failed or unrestricted profile endpoints. All reported estimates and bounds use squared score units. Scalar component intervals cannot be added to obtain SEM, G/Phi or covariance-element intervals. Use vcov.mfrm_multivariate_gstudy() separately for sampling SEs of the original ANOVA/MINQUE component estimates. Those SEs use normal product moments and do not describe the separate REML estimator or calibrate its interval.

Use plot(g, intervals = TRUE, method = "reml_profile", assumption = "normal") to compute the intervals and draw the original points in one call. Its returned data retain the full interval table, including REML centers and diagnostics. Save that plot object with saveRDS() and use as_ggplot() after reloading to render it without another fit.

Calculation of the SE

Let MS be the independent ANOVA mean squares, df their degrees of freedom, and A the linear map from mean squares to raw components. Under the normal model, df[i] * MS[i] / E(MS[i]) is chi-square. The sampling covariance is estimated by A %*% diag(2 * MS^2 / (df + 2)) %*% t(A); SE is the square root of its diagonal. The squared SE is unbiased for the sampling variance under this model; the SE itself need not be unbiased. This follows from E(MS[i]^2) = E(MS[i])^2 * (df[i] + 2) / df[i] and matches the urGENOVA SEVC convention. The existing paired D-study comparison instead uses the plug-in denominator df. Its method is unchanged. MLS intervals are calculated from the mean squares and chi-square/F quantiles, not by adding and subtracting a normal critical value times this SE.

Between-score covariance elements

For example, score = c("V", "W") estimates how the V and W scores vary together for each source. A negative covariance can be meaningful; its interval is signed and is never intersected with nonnegative variances. Covariance units are the product of the two score units. These are not correlations, and large covariance alone does not establish a halo effect. Select the pair before examining its results. Raw negative estimates and non-positive-semidefinite component matrices are retained without repair.

For each independent mean-product matrix S with df nu, nu*S has a Wishart distribution under the balanced normal model. Write its entries as x = S[1,1], c = S[1,2], y = S[2,2]. For nu > 1, an unbiased estimator of the sampling variance of c is (nu*x*y + (nu-2)*c^2)/((nu-1)*(nu+2)). Sum these estimates with squared ANOVA-map coefficients to obtain the component's squared SE. Its square root need not be unbiased. If a contributing block has df one, this SE formula is unavailable; SEStatus says so independently of interval status. Computation scales the scores separately, so a change in their relative units does not erase the smaller marginal variance. The mean_products attribute retains normalized entries (x, c, y), the two score_scale divisors and their product scale; multiply the normalized covariance column by scale to restore covariance units (use the respective squared divisors for the two variance columns). A squared SE can be outside floating-point range even when its square root is representable; the calculation retains that representable SE. Unrepresentable endpoints or a positive interval width lost to rounding produce missing limits with an explicit Status, retaining the original estimate and any calculable SE. This also applies to residual-contrast covariance intervals, which retain their separate IntervalCenter.

Generalized interval. Krishnamoorthy and Xia (2007, Section 2) construct a covariance-matrix pivot `C %*% solve(V)`, where `C` is the lower Cholesky factor of the observed sum-of-products matrix and `V` has independent Bartlett entries. Its covariance element simplifies to `nu * (c - sqrt(x*y-c^2)*Z/sqrt(U2)) / U1`, with independent `U1 ~ chi-square(nu)`, `U2 ~ chi-square(nu-1)` and `Z ~ N(0,1)`. Our extension generates independent pivots for the ANOVA blocks, applies the signed component map, and takes type-1 empirical quantiles. It does not filter negative draws or impose component-matrix constraints. Each contributing block must have df greater than one and a nonsingular pair. The paper's favorable results for a simple correlation do not establish exact frequentist coverage for these G-theory covariance components.

The `monte_carlo` attribute gives 95% joint numerical bands for the two endpoints of each component interval, using binomial order-statistic bounds. These bands concern simulation error in the quantiles. They do not measure source-study uncertainty or establish sampling coverage. More draws reduce numerical error; they do not resolve insufficient facet levels. At a very small requested confidence level, the two quantile probabilities or empirical ranks may coincide. In that case the interval is unavailable and `Status` distinguishes numerical probability resolution from an unresolved interval at the current draw count. Increasing `draws` may resolve ranks but cannot restore distinct probabilities lost to rounding.

Conservative enclosure. For each component with `K` contributing independent mean-product blocks, give each block coverage `gamma = level^(1/K)`. When `nu > 1` and the pair is nonsingular, the regression-slope t pivot (df `nu-1`) is independent of the predictor sum-of-squares chi-square pivot (df `nu`). Give each pivot coverage `sqrt(gamma)`. The covariance is the slope times the predictor variance; enclose their product over that rectangle. In terms of `S`, take all products of `c +/- t_quantile*sqrt((x*y-c^2)/(nu-1))` and the two ratios `nu / chi_square_quantile`, with equal tails `(1-sqrt(gamma))/2`. This expression is symmetric in the scores and respects changes of units.

For a one-df or singular block, use two one-sided marginal variance upper limits, each with tail `(1-gamma)/2`, and Cauchy-Schwarz to bound covariance between plus/minus the square root of their product. Bonferroni protects this block. Propagate block intervals through the signed ANOVA map. The independent block events give at least `level` coverage for the component under positive marginal variances and normal effects. This is our conservative construction from standard pivots, not a published G-theory formula attributed to Krishnamoorthy and Xia. It is pointwise, not simultaneous over components or score pairs. It can be extremely wide: enclosing nuisance regions adds conservatism beyond the source design's limited information.

A limited normal-model check covered six small/larger one-facet, crossed and nested sources. A 200-replication GCI screen was followed by four exploratory checks using 2000 saved samples and 8000 pivotal draws each. Their nominal 95% coverages were 94.65%, 94.80%, 99.65% and 96.50% (Monte Carlo SE 0.13–0.50 percentage points). These concern respectively one-facet residual (10 persons, 2 tasks), crossed residual and Task (10 persons, 2 raters, 3 tasks), and crossed Rater (60,4,8) covariances. The small Task interval remained very wide; covariance GCIs involving a one-df block were unavailable. This check does not establish universal coverage or qualify a generally preferred covariance method. Neither method is distribution robust or applies to incomplete MINQUE(0) sources.

`plot(g, type = "covariance", score = c("V", "W"), intervals = TRUE, method = "gci", assumption = "normal")` displays one signed interval plot across sources. Without intervals, `type = "covariance"` retains its matrix heatmaps. Saved plot data retain the full interval table and numerical bands.

Joint REML covariance profiles

For incomplete sources, explicitly select `method = "reml_profile"` with two original scores. This fits both scores jointly, with one mean per score, independent bivariate normal random effects, a positive-semidefinite matrix for each component and a positive-definite residual matrix. Omissions must be noninformative conditional on the observed assignment. This is a different inference procedure from MINQUE: the original covariance remains `Estimate`, while `IntervalCenter` is the separate joint REML covariance. Both target the same population quantity. The original point can lie outside the interval.

With score-major observations, let `V = sum(Sigma[k] %x% K[k])`, where each `K[k]` records shared levels, and `X = I[2] %x% rep(1,n)`. The restricted deviance is `log|V| + log|X'V^-1 X| + y'Py` plus a constant, with `P = V^-1 - V^-1 X (X'V^-1 X)^-1 X'V^-1`. Fix one off-diagonal component element, optimize all other variances and covariances, and invert `D - D_min <= qchisq(level, 1)`. No marginal interval endpoints are combined. Internally, residual whitening permits sparse lme4 evaluation after restoring residual scale and the transformation determinant. Independent direct-matrix calculations verify that criterion.

Zero covariance is permitted exactly; it does not imply zero marginal variances. Signed matrix factors preserve positive semidefiniteness during optimization. Covariance-coordinate derivatives check boundary optimality as well as ordinary parameter gradients. This numerical parameterization does not remove the statistical boundary problem (Wu & Neale, 2013). Multiple starts and these necessary local checks do not prove global optima. If the initially selected candidate fails these checks, already computed converged candidates within `1e-7` deviance of the smallest recorded value are checked in objective order. The first candidate passing every unchanged gradient, covariance-cone and target-constraint check is used. More distant local solutions are not substituted. Selected indices and inspected checks are retained in each fit's `selection`; no further optimization is triggered.

The returned interval surrounds the REML center and ends at the first checked cutoff crossing in each direction; it does not enumerate remote accepted regions. Unresolved source/constraint fits, roots or tails leave accepted bounds unavailable, preserving numerical candidates separately. A search limit is not an infinite endpoint. Constant or numerically collinear score pairs are degenerate, not evidence of zero uncertainty.

This method is experimental and never selected automatically. Limited balanced normal-model checks used 40 samples per condition and found true-value LR inclusion of 82.5–100%, counting numerical failures as misses (Monte Carlo SE up to 6.0 percentage points). The lowest result concerned Rater covariance with 60 persons, four raters and eight tasks. More persons alone do not increase the number of independent rater effects. These checks do not establish full endpoint-search coverage; incomplete-source coverage is not established. Do not interpret numerical availability as a coverage guarantee or replace the existing balanced methods merely because this method returns a narrower interval. No universal boundary-mixture correction is assumed. The intervals are pointwise and not distribution robust.

`SE` is `NA`: profile inversion does not supply sampling SEs for either the original point or REML center. `RawLower/RawUpper` are inapplicable here; `CandidateLower/Upper` retain exploratory endpoints. All reported covariance estimates and bounds use products of the score units, retaining negative signs. Saved diagnostics use standardized units; multiply by both entries of `attr(result, "reml_source")$score_scale` to restore score-product units.

References

Stanford University, STATS 305C. Multivariate regression, Distribution results. Normal-model residual Wishart law; the G-study projection is described above.

Wu, H., & Neale, M. C. (2013). On the likelihood ratio tests in bivariate ACDE models. *Psychometrika, 78*, 441–463. doi:10.1007/s1133601293042. Boundary-inference cautions, not a calibration formula for these G-theory covariance-element intervals.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. doi:10.18637/jss.v067.i01. Restricted likelihood and sparse computation.

Krishnamoorthy, K., & Xia, Y. (2007). Inferences on correlation coefficients: One-sample, independent and correlated cases. *Journal of Statistical Planning and Inference*, 137, 2362–2379. doi:10.1016/j.jspi.2006.08.002. Section 2 supplies the Wishart matrix pivot; component propagation is the extension documented above, with separate finite-sample evaluation.

R Core Team. Random Wishart distributed matrices. Normal sampling representation and elementwise sampling variance.

Ting, N., Burdick, R. K., Graybill, F. A., Jeyaratnam, S., & Lu, T.-F. C. (1990). Confidence intervals on linear combinations of variance components that are unrestricted in sign. *Journal of Statistical Computation and Simulation, 35*(3–4), 135–143. doi:10.1080/00949659008811240.

Hess, A., & Iyer, H. (2002). A SAS/IML macro for computation of confidence intervals for variance components of mixed models. *Proceedings of SUGI 27*, Paper 252-27, pp. 2–3. https://support.sas.com/resources/papers/proceedings/proceedings/sugi27/p252-27.pdf.

Brennan, R. L. (2001). *Manual for urGENOVA, Version 2.1*. SEVC and interval definitions, pp. 11–12; Appendix C numerical example, p. 26. Available from CASMA.

See Also

mfrm_g_study(), plot.mfrm_multivariate_gstudy(), mfrm_multivariate_d_compare(), mfrmr_methods_reference

Examples

```
# Long-running workflow; run explicitly when needed.
# example("confint.mfrm_multivariate_gstudy", run.dontrun = TRUE)
## Not run:
# Published synthetic data: 10 persons, 6 common tasks, scores V and W.
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
ci <- confint(g, weights = c(V = .5, W = .5), assumption = "normal")
ci[, c("Source", "Estimate", "SE", "Lower", "Upper", "Status")]
attr(ci, "notes")
# One original score and one component; no new G-study is fitted.
confint(g, parm = "Person", score = "V", assumption = "normal", level = .90)
# Covariance between the two observed scores, retaining its sign.
covariance_ci <- confint(g, score = c("V", "W"), method = "gci",
  assumption = "normal")
covariance_ci[, c("Source", "Estimate", "SE", "Lower", "Upper", "Status")]
attr(covariance_ci, "monte_carlo")$endpoint_bands
# Incomplete ratings: a residual-only interval without likelihood fitting.
observed <- tasks[(tasks$Person + tasks$Task) %% 3 != 0, ]
```

```
sparse_g <- mfrm_g_study(observed, c("V", "W"), rater = NULL,
                                  method = "minque0")
residual_ci <- confint(sparse_g, parm = "Residual", score = "V",
  method = "residual_contrast", assumption = "normal")
residual_ci[, c("Estimate", "IntervalCenter", "Lower", "Upper", "Status")]
attr(residual_ci, "residual_projection")$df
# A separate experimental REML model can address other components.
if (requireNamespace("lme4", quietly = TRUE) && requireNamespace("minqa", quietly = TRUE)) {
  profile_ci <- confint(sparse_g, score = "V", method = "reml_profile",
                        assumption = "normal")
  profile_ci[, c("Source", "Estimate", "IntervalCenter", "Lower", "Upper", "SEStatus")]
  attr(profile_ci, "notes")
  # Two scores require a joint model; this is covariance, not correlation.
  joint_ci <- confint(sparse_g, parm = "Person", score = c("V", "W"),
                      method = "reml_profile", assumption = "normal")
  joint_ci[, c("Source", "Estimate", "IntervalCenter", "Lower", "Upper", "Validation")]
}

## End(Not run)
```

`confint.mfrm_random_rater` *Population-SD profiles and explicit shared-rater model intervals*

Description

Population-SD profiles and explicit shared-rater model intervals

Usage

```
## S3 method for class 'mfrm_random_rater'
confint(object, parm = "rater_sd", level = 0.95, ...)
```

Arguments

`object`	A result from `fit_mfrm_random_rater()`. Profiling requires estimated rater SD.
`parm`	`"rater_sd"` (default) profiles population variation. `"raters"` explicitly requests first-order normal prediction intervals for all observed raters. These are not supplied automatically because their nominal coverage is not established. `"calibration"` explicitly requests observed-information normal approximations for fixed-facet levels and steps, excluding population SDs.
`level`	Confidence level, between zero and one; default 0.95.
`...`	Unused.

Details

For `parm = "rater_sd"`, fixed facets, steps and any estimated ability SD are refitted at each candidate rater SD, including zero. A specified known ability SD stays fixed. Earlier saved fits retain their known N(0,1) population. The search brackets a lower and upper crossing of twice the log-likelihood loss with `qchisq(level, df = 1)`, around the fitted SD. It uses the same approximate marginal likelihood as the fit. The lower bound is zero when the zero-variance submodel belongs to the likelihood-ratio region. An estimated boundary can therefore have a positive upper limit even though ordinary Wald intervals are unavailable. Each nonzero endpoint must attain the likelihood-ratio cutoff to `1e-4 * min(1, cutoff)` absolute accuracy. Insufficient floating-point resolution stops the search instead of reporting coincident limits. Profile refits require positive nuisance information. With known ability SD, nuisance curvature is checked at both integration orders. These local numerical checks do not prove a global optimum or exclude disconnected likelihood-ratio regions between sampled candidates.

This is an asymptotic likelihood-ratio interval, not a finite-sample coverage guarantee. The chi-square reference is nonregular at a true zero variance; the usual one-degree-of-freedom cutoff is conservative under the standard single-variance boundary asymptotics. Few raters, Laplace error and design misspecification can alter coverage. The numerical checks do not establish those asymptotic conditions. The calculation stops if the fitted or profiled ability variance is an estimated zero boundary; that additional nuisance boundary requires a different reference distribution for inference. No ability-SD interval is supplied. This interval does not quantify the predictive distribution of a replacement rater, whose variation is a different target.

The separate `parm = "raters"` calculation uses the conditional mode plus or minus `qnorm((1 + level)/2) * PredictionSE`. It targets realized, uncentered rater effects relative to the population mean. First-order calibration uncertainty does not ensure nominal coverage. In particular, few-rater coverage remains unresolved; see `vignette("mfrmr-random-raters")`. Requesting this approximation does not make it qualified for classification or exclusion of raters. It is not a profile interval, a bootstrap or an interval for a difference of raters.

Value

A one-row matrix with `Lower` and `Upper`, and attributes `level`, `method`, `profile` and `note`. The profile retains every evaluated SD, refitted likelihood, `LR` (twice the loss from the source fit), information checks and `Detail`. Unresolved numerical checks or endpoint searches stop with an `mfrm_profile_error`; its `profile` field retains the evaluations. A failed search does not establish that a statistical interval is absent. With `parm = "raters"`, one row per rater, computed from saved estimates and `PredictionSE` without refitting or RTMB. Attributes retain the method, target, level and interpretation. Unresolved numerical/information checks, estimated variance boundaries and unavailable SEs give missing bounds. With `parm = "calibration"`, one row per fixed-facet level or step, with the same level/method/target/note attributes and availability guards. Default calibration tables omit bounds; this explicit calculation uses saved SEs without refitting and does not establish finite-sample coverage. Both normal-interval routes retain row-level `IntervalStatus`, `CIEligible` and plain-language `InferenceReview` in `attr(ci, "diagnostics")`, together with saved estimates/SEs, and source checks in `attr(ci, "checks")`. Export the diagnostics table when saving CSV; a two-column numeric matrix alone loses the explanation. See `confint.mfrm_testlet()` for the status meanings. A missing interval does not turn a retained point into a reliable estimate, and more iterations cannot by themselves remove an estimated-variance boundary restriction.

References

Self, S. G. and Liang, K.-Y. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association*, 82, 605–610. doi:10.1080/01621459.1987.10478472.

`confint.mfrm_testlet` *Explicit normal-approximation intervals for testlet calibration*

Description

Explicit normal-approximation intervals for testlet calibration

Usage

```
## S3 method for class 'mfrm_testlet'
confint(object, parm = "calibration", level = 0.95, ...)
```

Arguments

`object`	A result from `fit_mfrm_testlet()`.
`parm`	`"calibration"` returns all fixed-facet and step intervals. Variance components and Person abilities are not included.
`level`	Nominal confidence level between zero and one; default 0.95.
`...`	Unused.

Details

The explicit request computes estimate plus/minus a normal quantile times the saved observed-information SE. It needs no fitting, scoring or live optimizer and does not change the source fit. Nominal finite-sample coverage is not established. Estimated variance boundaries, unresolved numerical/information checks and invalid SEs retain missing bounds. No regular variance interval, simultaneous comparison or automatic rater classification is supplied. Default fit tables, summaries and plots omit these bounds. Use `summary(fit, calibration_intervals = "normal")`, `plot(fit, intervals = "normal")`, or `mfrm_results(fit, calibration_intervals = "normal")` to select the same approximation in those outputs; their level arguments retain its nominal interpretation. `predict.mfrm_testlet()` supplies the separate conditional Person-scoring intervals.

Value

A matrix with `Lower` and `Upper`, one row per fixed-facet level or step, and attributes `level`, `method`, `target`, `note`, `checks` and `diagnostics`. `attr(ci, "diagnostics")` retains each estimate, SE, `IntervalStatus`, `CIEligible` and plain-language `InferenceReview`; `checks` retains the source checks without recomputing them. Save the whole object with `saveRDS()`; for CSV, export the diagnostics table to retain reasons.

Reading missing bounds

estimated_variance_boundary means an estimated variance is at zero; regular normal inference is withheld even when a point is retained. numerical_review_failed means numerical readiness or positive information has not been established; InferenceReview retains all detected reasons, including any simultaneous boundary. Increasing iterations alone does not repair a variance-boundary restriction. Inspect attr(ci, "checks") first. invalid_estimate_or_se and nonfinite_bounds identify row-specific calculation failures. Missing bounds are neither zero uncertainty nor infinite endpoints. Retaining a point does not certify its reliability. Default calibration tables also distinguish not_requested from unsupported_target (population SDs/variances). CIEligible is true only for a returned normal interval; it is not a coverage guarantee.

See Also

confint.mfrm_random_rater(), mfrmr_interval_guide()

---

data_quality_report *Build a data quality summary report (preferred alias)*

Description

Build a data quality summary report (preferred alias)

Usage

```
data_quality_report(
  fit,
  data = NULL,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  min_category_count = 10,
  dominant_category_cutoff = 0.95,
  include_fixed = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
data	Optional raw data frame used for row-level review.
person	Optional person column name in data.
facets	Optional facet column names in data.
score	Optional score column name in data.
weight	Optional weight column name in data.

`min_category_count`
: Minimum raw or weighted count used to label a non-zero facet-level score category as sparse. Default `10`.

`dominant_category_cutoff`
: Proportion in `(0, 1]` used to flag a facet level whose responses are dominated by one score category. Default `0.95`.

`include_fixed`
: If `TRUE`, include a legacy-compatible fixed-width text block.

Details

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_data_quality` (`type = "dashboard"`, `"quality_flags"`, `"row_review"`, `"category_counts"`, `"score_support"`, `"facet_category_usage"`, `"facet_response_patterns"`, `"score_map"`, `"missing_rows"`).

Value

A named list with data-quality report components. Class: `mfrm_data_quality`.

Interpreting output

- `summary`: retained/dropped row overview.
- `quality_overview`: area-level QC status for rows, score support, facet-category use, and design matching.
- `quality_flags`: prioritized QC flags with counts and recommended next actions. This is not an item/person/rater table.
- `row_review`: reason-level breakdown for data issues.
- `category_counts`: post-filter category usage, including retained zero-count score-support categories.
- `score_support_review`: quick view of zero-count boundary/intermediate categories and their threshold-functioning caveats.
- `category_usage_by_facet`: facet-level category counts over the retained score support.
- `category_usage_summary`: per-facet-level zero/sparse category summary.
- `facet_response_patterns`: facet-level response-pattern summaries, including single-category and dominant-category use.
- `caveats`: user-facing score-support warnings, including cases where non-consecutive original labels such as `1, 2, 4, 5` were recoded because `keep_original = FALSE`.
- `score_map`: original-to-internal score mapping used when labels are recoded.
- `unknown_elements`: facet levels in raw data but not in fitted design.

Local category avoidance is not automatically a GPCM problem

Suppose a declared 1–10 scale is observed globally, but one rater uses only scores 3–8. If other raters use 1, 2, 9, and 10, the shared score support is not globally missing. The affected rater instead has local zero-category use: inspect `category_usage_by_facet`, `category_usage_summary`, and

`plot(out, type = "facet_category_usage")`. This pattern can reflect range restriction, assignment/case mix, sparse information, or operational scoring practice; the table alone does not identify the cause.

Do not select `GPCM` merely to absorb category avoidance. `GPCM` changes the designated step/slope owner's threshold and discrimination structure; it does not repair a rater's restricted use of the rubric. When the affected rater is itself the `step_facet`/`slope_facet` owner, local category gaps can also weaken owner-specific PCM/GPCM parameter support and the fit-readiness result must be checked. Under RSM, or when another facet owns the steps, the same local pattern remains important response-use evidence without automatically becoming a distinct response model.

Typical workflow

1. Run `data_quality_report(...)` with raw data.
2. Check `summary(out)` and `plot(out, type = "dashboard")`, then inspect `quality_flags`, score-support, score-map, facet-response-pattern, and missing/unknown element sections as needed.
3. Resolve missing values, score-support gaps, and sparse categories before final estimation/reporting.

See Also

`fit_mfrm()`, `describe_mfrm_data()`, `specifications_report()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Review the input-data checks recorded with this fit
quality <- data_quality_report(fit)
summary(quality)

# Draw the data-quality overview
plot(quality)
```

---

describe_mfrm_data *Check rating data before fitting an MFRM*

Description

Inspect how many rating rows can be used, how often each score category occurs, and whether facet levels connect through shared persons. This function prepares descriptive checks; it does not fit a model or change the data object supplied by the caller. Each row should represent one rating, and `person`, `facets`, and `score` name its columns.

Usage

```
describe_mfrm_data(
  data,
  person,
  facets,
  score,
  weight = NULL,
  rating_min = NULL,
  rating_max = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  include_person_facet = FALSE,
  include_agreement = TRUE,
  rater_facet = NULL,
  context_facets = NULL,
  agreement_top_n = NULL,
  expected_design = NULL,
  min_linking_persons = 2L,
  category_policy = NULL,
  repeated_ratings = c("review", "independent"),
  comparisons = NULL
)

## S3 method for class 'mfrm_data_description'
summary(object, digits = 3, top_n = 10, ...)

## S3 method for class 'mfrm_data_description'
plot(
  x,
  y = NULL,
  type = c("score_distribution", "facet_levels", "missing"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

data	A data.frame in long format (one row per rating event). Grouped or rowwise tibbles are treated as one table; only the selected model columns enter row filtering and the design review.
person	Column name for person IDs.
facets	Character vector of facet column names. The reserved data and parameter names listed in `fit_mfrm()` cannot name facets; rename such columns before analysis.
score	Column name for observed score. The category-code and near-integer normalization rules are the same as in `fit_mfrm()`.

`weight` Optional weight/frequency column name, with the numeric precision and finite-sum rules of `fit_mfrm()`.

`rating_min` Optional minimum category value. Supply with `rating_max` to retain unused boundary categories in the intended score support.

`rating_max` Optional maximum category value. Supply with `rating_min` to retain unused boundary categories in the intended score support.

`keep_original` Keep original category values. Use this with `rating_min` / `rating_max` when the intended scale has unused intermediate categories such as `1, 2, 4, 5` on a 1-5 scale. New code can instead use `category_policy = "preserve"`.

`missing_codes` Optional. `NULL` (default) is a no-op; `TRUE` or `"default"` activates the FACETS / SPSS / SAS convention (`c("99", "999", "-1", "N", "NA", "n/a", ".", "")`) for the score column while preserving person/facet identifiers; supply a character vector to apply a custom code set across all model columns. Replacement counts are returned in the `missing_recoding` component when supported by the calling helper. See `recode_missing_codes()` for the standalone version.

`include_person_facet`
If `TRUE`, include person-level rows in `facet_level_summary`.

`include_agreement`
If `TRUE`, include an observed-score agreement bundle (summary/pairs/settings) for a selected non-person facet.

`rater_facet` Optional facet name used to identify repeated scorers for agreement summaries. If `NULL`, a rater-like name such as `Rater`, `Judge`, or `Scorer` is inferred. No agreement analysis is run when such a name is absent; set this argument explicitly only when another facet genuinely represents repeated scorers.

`context_facets` Optional facets used to define matched contexts for agreement. If `NULL`, all remaining facets (including `Person`) are used.

`agreement_top_n`
Optional maximum number of agreement pair rows.

`expected_design`
Optional data frame declaring the planned assignment roster. It must contain the columns named by `person` and `facets`, with one row per planned Person x facet cell. Extra columns are ignored. When supplied, observed cells are compared with the roster so planned omissions can be distinguished from cells that were never assigned.

`min_linking_persons`
Positive integer used as a descriptive sparse-link flag. A facet level observed for fewer than this many distinct persons is counted in `linkage_summary$SparseLevels`. This is a review threshold, not a model-acceptance rule.

`category_policy`
Optional explicit category choice: `"collapse"` maps gaps in the observed categories to consecutive scores; `"preserve"` keeps the intended ladder, declared with `rating_min` and `rating_max`. This changes the fitted category steps, not just labels. `NULL` (default) uses `keep_original`, whose default is `FALSE` (`"collapse"`). Supplying both choices is allowed only when they agree. Preservation does not estimate unsupported steps: fitting stops if a retained internal category has no observations. Use the same policy in `describe_mfrm_data()` and `review_mfrm_anchors()`.

`repeated_ratings`
: Treatment of repeated Person-by-facet cells. `"review"` (default) retains the duplicate-cell warning and ordinary MML/JML input-review state. Use `"independent"` only to declare that the rows are distinct, conditionally independent rating events under the specified model. Every row and its weight are retained; this argument does not aggregate scores, add a facet, change the likelihood or establish independence from the data. Other input, numerical and inference checks still apply. The declaration is saved in preparation notes, configuration and replay settings. Corrected JML already requires conditionally independent repeated ratings; the argument records that assumption explicitly there. It is not supported by the two-family GPCM route.

`comparisons`
: Optional named specification for a prospective facet-by-origin interaction review. See "Comparison-specific review". Default `NULL` performs no model/target review or fitting.

`object`
: Output from `describe_mfrm_data()`.

`digits`
: Number of digits for numeric rounding.

`top_n`
: Maximum rows shown in preview blocks.

`...`
: Reserved for generic compatibility.

`x`
: Output from `describe_mfrm_data()`.

`y`
: Reserved for generic compatibility.

`type`
: Plot type: `"score_distribution"`, `"facet_levels"`, or `"missing"`.

`main`
: Optional title override.

`palette`
: Optional named colors (`score`, `facet`, `missing`).

`label_angle`
: X-axis label angle for bar plots.

`draw`
: If `TRUE`, draw using base graphics.

Details

Set `rating_min` and `rating_max` from the rubric, including categories nobody received. Use `keep_original = TRUE` to preserve its category structure in the review. Numeric descriptives of score and weight use `psych::describe()`.

Key data-quality checks to perform before fitting:

- *Sparse categories*: review categories with little weighted support because their threshold estimates may be imprecise. Do not collapse categories solely from a package warning; also consider the rubric, intended score interpretation, and category diagnostics after fitting.
- *Unlinked elements*: inspect `design_connectivity` for the observed Person-facet graph. More than one component means that the levels of that facet are not connected through shared persons. This facet-specific check is conservative and does not by itself prove full model identification.
- *Extreme scores*: MML uses a person distribution to obtain posterior person scores, including for persons with all-minimum or all-maximum scores. Non-person facets remain fixed effects: an extreme rater or criterion is not given a prior or automatically shrunk by choosing MML. Review the fitted boundary and precision evidence before interpretation.

summary() method: Summarize a data-description object

Read a compact summary of the checks from describe_mfrm_data() before fitting a model. Save it with review <- summary(data_review) and select the tables you need with $, as in the example.

For summary(): data_review holds the complete data checks; review holds summary tables and notes. Neither object contains model estimates. The default top_n = 10 limits the missing-column and score-distribution previews; use data_review$missing_by_column and data_review$score_distribution to see the complete tables.

plot() method: Plot a data-description object

For plot(): This method draws quick pre-fit quality views from describe_mfrm_data():

- score distribution balance
- facet-level structure size
- missingness by selected columns

Value

A list of class mfrm_data_description with:

- overview: one-row run-level summary including CategoryPolicy and ScoreRecoded. The former records the selected category handling; the latter indicates whether original score values actually changed. A "collapse" policy can leave a contiguous scale unchanged. Inspect score_support$score_map for the mapping.
- missing_by_column: missing counts in selected input columns
- missing_rate_summary: per-column missingness rate summary (one row per input column, with raw and proportion-of-N columns)
- score_descriptives: output from psych::describe() for score
- weight_descriptives: output from psych::describe() for weight
- score_distribution: weighted and raw score frequencies over the prepared score support. Unused boundary categories are retained when the rating range was supplied explicitly; unused intermediate categories require keep_original = TRUE.
- facet_level_summary: per-level usage and score summaries
- facet_crosstabs: pairwise observation-count crosstabs between non-person facets (named list keyed "facetA__facetB") for optional downstream coverage displays
- linkage_summary: person-facet connectivity diagnostics
- structural_missingness: declared-design comparison bundle containing a one-row summary, missing expected cells, unexpected observed cells, per-facet level coverage, and settings
- design_connectivity: component counts for each observed Person-facet graph and, when declared, each expected Person-facet graph
- design_components: component-level counts and facet-level labels; person labels are included only when include_person_facet = TRUE
- duplicate_cell_summary: counts of duplicate Person x facet cells
- duplicate_cell_detail: duplicate-cell keys and row counts
- agreement: observed-score agreement bundle for the selected scorer facet

- row_retention: row counts before and after preparation filters
- preparation_notes: structured notes for row drops, ID trimming, and design conditions detected during preparation
- missing_recoding: per-column counts of declared missing-code values replaced with NA before row filtering
- score_support: minimal prepared score-support metadata used by summary(ds)$caveats
- comparison_review: optional prospective interaction review, including all requested rows and their explicit weights; also retained by summary().

For summary(): An object of class summary.mfrm_data_description.

- overview: design/sample counts
- missing: top columns by missingness
- score_distribution: compact score-usage table, including zero-count categories retained by the prepared score support
- facet_overview: facet-level coverage summary
- structural_missingness: declared assignment coverage summary; status is "not_declared" when no assignment roster was supplied
- structural_level_coverage: expected versus observed level counts
- design_connectivity: Person-facet component counts for observed and declared-expected designs
- design_components: component sizes and facet-level labels; person labels are suppressed unless explicitly requested in describe_mfrm_data()
- linkage_summary: sparse-support and shared-person counts by facet
- duplicate_cell_summary: aggregate duplicate-cell counts
- agreement: selected-facet agreement summary when available
- agreement_settings: selected scorer facet, matching context, and status
- row_retention: row counts before and after preparation filters
- preparation_notes: structured preparation notes retained from describe_mfrm_data()
- reporting_map: manuscript-oriented guide to what is covered here versus which companion outputs should be consulted
- caveats: structured warning/review rows for score-support issues; print(summary(ds)) shows a compact Caveats block when rows are present
- notes: plain-language explanations of missingness, preparation, score support, and design-review findings

For plot(): A plotting-data object of class mfrm_plot_data.

Comparison-specific review

Use `comparisons = list(facet = ..., group = ..., groups = c("A", "B"), contrasts = W, model = "RSM", method = "JML")` to review explicitly declared interaction comparisons before estimation. Both `facet` and `group` must occur in `facets`; each Person/recording must keep one origin group. `groups` gives the subtraction order, including when additional neutral origins are present. Unit weights and preserved categories are required. `W` is a numeric matrix with unique comparison row names and every reviewed facet level as a named column, including zero weights. Each nonzero row must sum to zero. For four raters, weights `c(.75, -.25, -.25, -.25)` compare the first rater's A-minus-B interaction with the equal-weight panel; `c(1, -1, 0, 0)` compares the first two raters' A-minus-B interactions. These are different targets.

The prospective model has centered measurement facets, a zero-valued group main facet, and the named facet-by-group interaction with zero marginal sums. JML has free Person locations. Choose `model = "PCM"` with an explicit `step_facet` for facet-owned steps. Optional `positive_facets` records the location orientation. Anchors, extra interactions and nonunit weights are outside this review. GPCM interaction requests return `unsupported`, not an additive-model substitute.

`method = "MML"` uses a fixed common standard-normal population unless `population_formula` is supplied (an intercept or the origin group only). `ConditionalState` refers to the linear response map; `FreePersonState` reviews the same response structure with unrestricted Persons. A same-law alias remains unidentified after mixing. A positive conditional rank alone does not certify the marginal model: those MML rows have `State = "not_evaluated"`. No fit or numerical marginal-rank calculation is performed. JML rows distinguish `all_parameters_identified`, `target_identified_only`, and `target_unidentified`. Numerically sensitive or oversized dense maps remain `not_evaluated`.

`IdentificationBasis` interprets these checks for each requested target: `free_person_response_model` means the declared response map identifies it even with unrestricted Person locations. `population_restriction` means an unrestricted-Person alias is removed only after imposing the declared population restrictions in the conditional map. This does **not** certify marginal identification, and the corresponding MML `State` stays `not_evaluated`. `conditional_support` identifies a CML target from response patterns at the observed totals. `unidentified`, `not_evaluated` and `unsupported` remain separate. `IdentificationDetail` explains the scope; `Basis` retains the broader estimator/model basis.

A fixed common N(0,1) population or estimated common normal population (`~1`) restricts origin groups to the same ability distribution. This is substantive, not arbitrary scale coding. Origin regression relaxes only the means allowed by its design and retains a common residual SD. Interpret the zero origin main effect jointly with those restrictions. A contrast identified in a free-Person response map does not make its MML estimator distribution-free. All conclusions still depend on the declared response model, including equal slopes and shared/facet-owned step restrictions.

`method = "CML"` instead reviews conditional support at the actual Person totals and rating rosters. A dynamic program spans differences between all possible patterns at those totals, so extreme scores and unavailable category combinations do not manufacture information. `State` and `ConditionalState` distinguish full identification, target-only identification and unidentified targets; `FreePersonState` is the separate unrestricted-Person response-map review. Person-constant main effects disappear, and no population distribution or SD is estimated. Population formulas are not accepted for CML.

Read `$comparison_review$table`, `$weights`, `$cells` and `$notes` together. The table preserves group direction, model, step owner and counts; counts cover nonzero-weight facet levels, while `cells` includes empty cells. `$fit_arguments` records the prospective structural settings; use

matching input preparation when fitting. An identified target does not permit fitting a rejected full parameterization. Estimate stays missing and formal inference is unavailable. Save the review with its retained response rows. Pass it as comparison_review to analyze_dff() to preserve this separate prospective question alongside residual/refit effects; it does not certify those different effects. See vignette("mfrmr-linking-and-dff") for a school-assignment example and mfrmr_linking_and_dff for interpretation. A matching, fully supported joint fit can return descriptive contrasts via interaction_effect_table() or summary(fit, comparison_review = review). These explicit calls add retained-point numerical checks, not partial estimation or formal uncertainty. To fit selected CML contrasts and obtain conditional-information Wald intervals, use fit_mfrm(fit_scope = "comparisons", comparison_review = review) with the CML review's $fit_arguments; see the comparison-only section of fit_mfrm().

Interpreting output

Recommended order:

- overview: confirms retained ratings (Observations), persons, facets, and category span. Use row_retention to compare input and retained Rows; DroppedRows counts exclusions during preparation.
- missing_by_column: counts NA values in the input columns. A missing score or required identifier excludes that rating row, not automatically the person's other ratings. The package does not fill missing ratings. When missing_codes is supplied, these counts still describe the original input; inspect missing_recoding and preparation_notes as well.
- structural_missingness: compares observed rating cells with expected_design, when supplied. Without a declared roster, structural missingness is reported as not assessed rather than assumed to be zero.
- score_distribution: checks sparse/unused score categories. Skew can be substantively expected, but weakly supported or unused categories need explicit interpretation.
- facet_level_summary and linkage_summary: checks per-level support, shared-person counts, and sparse levels. Use design_connectivity for the separate graph-component result.
- agreement: optional observed agreement summary for the selected scorer facet (exact agreement, correlation, and mean differences per pair).

data_review <- describe_mfrm_data(...) saves all these checks. review <- summary(data_review) provides a compact view; its missingness table is named review$missing, while the original full table is data_review$missing_by_column. Summary previews use top_n = 10 by default. Use the original tables to inspect all rows or categories.

For summary():

Recommended read order:

- overview: retained ratings (Observations), persons, facets, and categories. Compare input and retained Rows in row_retention and investigate unexpected DroppedRows. CategoryPolicy and ScoreRecoded distinguish the selected policy from actual changes to score values. Older results without the policy record report "not_recorded"; an absent map gives NA for recoding.
- missing: input NA counts by column. This table is named missing_by_column in the original data_review object. Declared missing-code replacements and invalid score text can cause additional row loss; inspect data_review$missing_recoding and preparation_notes.

- score_distribution: category usage balance.
- notes / printed Caveats: retained zero-count score categories and related score-support caveats. With keep_original = TRUE, a retained unused internal category stops fitting; review the data and rubric first.
- facet_overview: coverage per facet (minimum/maximum weighted counts).
- agreement: observed-score agreement for the selected scorer facet (when available).
- design_connectivity: check for more than one observed component before comparing facet levels. structural_missingness reports planned omissions only when an assignment roster was supplied; not_declared does not mean that no ratings are missing.

Very low MinWeightedN in facet_overview is a practical warning for unstable downstream facet estimates.

For plot():

- "score_distribution": bar chart of weighted observation counts per score category. Y-axis is WeightedN (sum of weights for each category). Categories with very few observations (< 10) may produce unstable threshold estimates. A roughly uniform or unimodal distribution is ideal; heavy floor/ceiling effects compress the measurement range.
- "facet_levels": bar chart showing the number of distinct levels per facet. Useful for verifying that the design structure matches expectations (e.g., expected number of raters or criteria). Very large numbers of levels increase computation time and may require higher maxit in fit_mfrm().
- "missing": bar chart of missing-value counts per input column. Columns with non-zero counts should be investigated before fitting—rows with missing scores, persons, or facet IDs are dropped during estimation.

If the input needs attention

- **Column name not found:** run names(ratings) and match spelling, spaces, and case in person, facets, and score.
- **Unexpected row loss:** inspect row_retention, missing_by_column, and preparation_notes. Resolve unintended missing IDs and invalid score text in the input data. For documented score markers such as 99 or ., use recode_missing_codes() with explicit columns and codes.
- **Repeated person-by-facet cells:** inspect duplicate_cell_detail. Correct accidental duplicates; declare repeated_ratings = "independent" for distinct conditionally independent events, or model dependence. Include a task or occasion facet when ratings represent distinct events in the design.
- **Unused category or disconnected design:** inspect score_distribution and design_connectivity. Review the rubric and assignments before changing the model. A retained internal zero-count category stops fitting; extra optimizer iterations cannot supply the missing category information.

After editing or recoding ratings, rerun this review on the corrected data and pass that same data to fit_mfrm(), using the same columns, score bounds, and keep_original setting. If you use missing_codes within the review instead of recoding first, supply the same option to the fit: reviewing does not modify the original data. For CSV import, column mapping, and a complete worked example, see vignette("mfrmr-workflow", package = "mfrmr").

Typical workflow

1. Run `data_review <- describe_mfrm_data(...)` on the rating data.
2. Inspect row retention, category counts, and design connectivity.
3. Correct input issues, repeat the review, and fit the reviewed data with `fit_mfrm()`.

For `summary()`:

1. Run `describe_mfrm_data()` on raw long-format data.
2. Inspect `review <- summary(data_review)` before model fitting.
3. Correct input issues and repeat the review, then pass the corrected rating data to `fit_mfrm()` with the same preparation settings.

For `plot()`:

1. Run `describe_mfrm_data()` before fitting.
2. Inspect `summary(ds)` and `plot(ds, type = "missing")`.
3. Check category/facet balance with other plot types.
4. Fit model after resolving obvious data issues.

See Also

`fit_mfrm()`, `summary.mfrm_data_description()`, `recode_missing_codes()`, mfrmr_workflow_methods, `review_mfrm_anchors()`

`describe_mfrm_data()`, `summary.mfrm_fit()`

`describe_mfrm_data()`, `plot()`

Examples

```
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")
head(ratings)

# Check the data before fitting; the intended score categories are 1 to 4
data_review <- describe_mfrm_data(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1,
  rating_max = 4,
  category_policy = "preserve"
)
data_review$row_retention       # Input and retained rows; check DroppedRows
data_review$missing_by_column   # Missing input values in each model column
data_review$score_distribution  # RawN is the number of ratings per category
data_review$design_connectivity # Components = 1 means connected for that facet
# Here all 282 rows are retained, and all four categories have observations

# Save a compact summary when you want the overview and review notes
```

```
review <- summary(data_review)
review$overview
review$notes
# For the next fit, use data = ratings; data_review is a set of checks, not ratings
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_data_description", run.dontrun = TRUE)
## Not run:
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Demonstrate two missing scores in a copy of the example data
ratings$Score[1:2] <- NA
data_review <- describe_mfrm_data(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1,
  rating_max = 4,
  category_policy = "preserve"
)
review <- summary(data_review)
review$row_retention # 282 input rows, 280 retained rows
review$missing       # Score has 2 missing input values
review$overview      # Counts describe the retained ratings
review$notes         # Explanations to read before fitting

# The original description retains the full missingness table
data_review$missing_by_column
# Investigate missingness before using ratings in fit_mfrm()

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_data_description", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
ds <- describe_mfrm_data(toy, "Person", c("Rater", "Criterion"), "Score")
p <- plot(ds, draw = FALSE)

## End(Not run)
```

`detect_anchor_drift` *Detect anchor drift across multiple calibrations*

Description

Compares facet estimates across two or more calibration waves to identify elements whose difficulty/severity has shifted beyond supplied screening thresholds. Useful for monitoring rater drift over time or checking the stability of item banks.

Usage

```
detect_anchor_drift(
  fits,
  facets = NULL,
  drift_threshold = 0.5,
  flag_se_ratio = 2,
  reference = 1L,
  include_person = FALSE
)

## S3 method for class 'mfrm_anchor_drift'
print(x, ...)

## S3 method for class 'mfrm_anchor_drift'
summary(object, ...)

## S3 method for class 'summary.mfrm_anchor_drift'
print(x, ...)

## S3 method for class 'mfrm_anchor_drift'
plot(x, type = c("drift", "heatmap", "forest"), ...)
```

Arguments

fits	Named list of mfrm_fit objects (e.g., list(Year1 = fit1, Year2 = fit2)).
facets	Character vector of facets to compare (default: all non-Person facets).
drift_threshold	Absolute screening threshold for flagging (logits, default 0.5). A non-negative scalar; Inf disables magnitude screening.
flag_se_ratio	Drift/SE ratio screening threshold for flagging (default 2.0). A non-negative scalar; Inf disables SE-ratio flagging.
reference	Index or name of the reference fit (default: first).
include_person	Include person estimates in comparison. Use only when the person identifiers refer to the same persons across fitted waves.
object	An mfrm_anchor_drift object (for summary).
...	For print.mfrm_anchor_drift(): Ignored. For plot(): facet optionally selects facets; ci_level = 0.95 controls forest whiskers only; preset selects "standard", "publication", "compact", or "monochrome"; draw = FALSE returns data without drawing. Additional base-graphics arguments depend on the selected view.
x	For print.mfrm_anchor_drift(): An mfrm_anchor_drift object. For plot(): A saved result from detect_anchor_drift().
type	"drift" for relative changes with screening flags, "heatmap" for wave-by-element patterns, or "forest" for per-wave estimates with approximate normal whiskers. A linking chain is a different result: use plot(chain, type = "chain") after build_equating_chain().

Details

For each non-reference wave, the function extracts facet-level estimates using `make_anchor_table()` and computes the element-by-element difference against the reference wave. Standard errors are obtained from `diagnose_mfrm()` applied to each fit. Only elements common to both the reference and a comparison wave are included. Before reporting drift, the function removes the weighted common-element link offset between the two waves so that `Drift` represents residual instability rather than the overall shift between calibrations. The function also records how many common elements survive the screening step within each linking facet and treats fewer than 5 retained common elements per facet as thin support. This count is a package screening convention, not a universal adequacy threshold.

One `LinkOffset` is pooled across all selected common elements and facets. If at least one pair has finite positive SEs, only pairs with such SEs contribute to the inverse-variance weighted offset; rows without usable SEs can still appear in retained/support counts. If preliminary residual screening would retain no row, the current implementation falls back to all finite differences. Use a single substantively coherent facet when a common shift across facet blocks is not defensible, and inspect `drift_table` rather than treating `LinkSupportAdequate` as sufficient evidence. The fallback is recorded as `ScreeningFallback`, and `Contributing` / `N_Contributing` identify the elements actually used in the final offset. Retention and contribution are different: a retained row with a missing SE does not enter a weighted offset when other rows have usable SEs.

`OffsetStatus` and `OffsetReason` distinguish computed offsets, absent or nonfinite common differences, and numerical failure. Missing residuals stay missing. `SERatioStatus` explains unavailable SE ratios; `FlagBasis` identifies whether the flag used both criteria, only residual magnitude, or neither. In the magnitude-only case, `Flag = FALSE` is not a negative SE-ratio screen. Summaries report unavailable counts, and withhold the complete flagged count when any residual flag is unavailable.

Matching `Facet` and `Level` labels are treated as common elements; the helper cannot verify that they have invariant meaning across waves. The input fits must therefore use compatible model, score, orientation, and population conventions and must be independently reviewed for fit readiness. Offset removal aligns coordinates conditionally on those assumptions; it is not proof of scale equivalence.

This is a relative drift screen. For example, if the selected judges' estimates all differ from the reference by +0.40 logits, `LinkOffset` is +0.40 and each residual `Drift` is zero. This does not establish panel stability. The offset itself also cannot distinguish real panel change from a change of coordinates. Common recorded performances can provide a separate reference, but a complete recording-based annual analysis and its formal difference intervals are not yet verified here. See the annual assessment example in mfrmr_linking_and_dff.

An element is **flagged** when either condition is met:

$$|\Delta_e| > \texttt{drift_threshold}$$

$$|\Delta_e / SE_{\Delta_e}| > \texttt{flag_se_ratio}$$

The two criteria are joined by `OR`: a large absolute residual is flagged even when imprecise, while a smaller residual can be flagged by its SE ratio. The ratio uses only the two element SEs and omits uncertainty in the estimated `LinkOffset` and any cross-fit covariance. It is therefore a descriptive screen, not a formal z test with a calibrated error rate. Residuals share an estimated offset, and contributing elements also enter that offset. Offset covariance can either reduce or increase residual uncertainty relative to the displayed denominator. The ratio is not guaranteed to be conservative.

When facets is NULL, all non-Person facets are compared. Providing a subset (e.g., facets = "Criterion") restricts comparison to those facets only.

plot() method: Plot a saved anchor-drift review

Inspect relative changes after the common-element offset used by detect_anchor_drift(). Read summary(x) and link support alongside the plot.

Value

Object of class mfrm_anchor_drift with components:

drift_table Tibble of element-level drift statistics.

summary Drift summary aggregated by facet and wave.

common_elements Tibble of pairwise common-element counts.

common_vs_reference Tibble of common-element counts between each wave and the reference wave (i.e., which labels remain available as candidate common elements).

n_common_all_waves Integer count of elements that are common across every wave; used by summary() to gauge how robust the chain is to chained linking error.

common_by_facet Tibble of retained common-element counts by facet.

config List of analysis configuration.

For plot(): An mfrm_plot_data object, or NULL when no drift rows remain. Drift/heatmap data use data$table (and data$matrix for a heatmap); forest rows use data$data, including CI_Lower and CI_Upper.

Which function should I use?

- Use anchor_to_baseline() when your starting point is raw new data plus a single baseline fit.
- Use detect_anchor_drift() when you already have multiple fitted waves and want a reference-versus-wave comparison.
- Use build_equating_chain() when the waves form a sequence and you need cumulative linking offsets.

Interpreting output

- $drift_table: one row per element x wave combination, with columns Facet, Level, Wave, Ref_Est, Wave_Est, LinkOffset, Drift, SE_Ref, SE_Wave, SE, Drift_SE_Ratio, LinkSupportAdequate, and Flag. Large drift signals instability after alignment to the common-element link.
- $summary: aggregated statistics by facet and wave: number of elements, mean/max absolute drift, and count of flagged elements.
- $common_elements: pairwise common-element counts in tidy table form. Small overlap weakens the comparison and results should be interpreted cautiously.
- $common_by_facet: retained common-element counts by linking facet for each reference-vs-wave comparison. LinkSupportAdequate = FALSE means the link rests on fewer than 5 retained common elements in at least one facet.

- `$config`: records the analysis parameters for reproducibility.
- A practical reading order is `summary(drift)` first, then `drift$drift_table`, then `drift$common_by_facet` if overlap looks thin.

Typical workflow

1. Fit separate models for each administration wave.
2. Combine into a named list: `fits <- list(Spring = fit_s, Fall = fit_f)`.
3. Call `drift <- detect_anchor_drift(fits)`.
4. Review `summary(drift)` and `plot(drift)`.
5. Flagged elements may need to be removed from anchor sets or investigated for substantive causes (e.g., rater re-training).

Interpretation

For `plot()`:

Drift and heatmap views show the saved relative drift. Flags use the screening rules chosen during analysis; changing the display does not recalibrate these rules. Subtracting the common-element offset can remove a panel-wide shift, so unflagged elements do not establish absence of annual drift.

Forest whiskers describe individual wave estimates, not uncertainty in their differences. They do not include linking-offset uncertainty or cross-wave covariance, and their overlap is not a test of change. Missing standard errors produce no whisker, not a zero-width interval.

See Also

`anchor_to_baseline()`, `build_equating_chain()`, `make_anchor_table()`, plot(x), mfrmr_linking_and_dff

`detect_anchor_drift()`, mfrmr_plot_migration

Examples

```
# Long-running workflow; run explicitly when needed.
# example("detect_anchor_drift", run.dontrun = TRUE)
## Not run:
# Deliberately linked teaching waves: common labels below represent the
# same rater and criterion identities by construction.
toy <- load_mfrmr_data("example_core")
people <- unique(toy$Person)
# Two balanced 24-Person waves retain every Rater and Criterion.
d1 <- toy[toy$Person %in% people[1:24], , drop = FALSE]
d2 <- toy[toy$Person %in% people[25:48], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "MML", quad_points = 7, maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "MML", quad_points = 7, maxit = 30)
drift <- detect_anchor_drift(list(Wave1 = fit1, Wave2 = fit2))
summary(drift)
head(drift$drift_table[, c("Facet", "Level", "Wave", "Drift", "Flag")])
```

```
drift$common_elements

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_anchor_drift", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
people <- unique(ratings$Person)
d1 <- ratings[ratings$Person %in% people[1:12], , drop = FALSE]
d2 <- ratings[ratings$Person %in% people[13:24], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML")
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML")
# Review convergence and common-element support before interpreting changes.
summary(fit1)
summary(fit2)
drift <- detect_anchor_drift(list(W1 = fit1, W2 = fit2))
drift_plot <- plot(drift, type = "drift", draw = FALSE)
class(drift_plot)
names(drift_plot$data)
chain <- build_equating_chain(list(F1 = fit1, F2 = fit2))
chain_plot <- plot(chain, type = "chain", draw = FALSE)
head(chain_plot$data$table)
if (interactive()) {
  plot(drift, type = "heatmap", preset = "publication")
}

## End(Not run)
```

`detect_facet_nesting` *Detect nesting structure between facets*

Description

Classifies both directions of each pair of facets (optionally including `Person`) as crossed, partially nested, near-perfectly nested, or fully nested, based on a conditional-entropy index:

$$\text{nesting_index}(A \to B) = 1 - H(B \mid A)/H(B).$$

An index near 1 means that knowing the level of `A` essentially determines the level of `B` (A is nested in B).

Usage

```
detect_facet_nesting(data, facets, person = NULL, weight_col = NULL)
```

Arguments

data	Data frame in long format (one row per rating).
facets	Character vector of facet column names.
person	Optional name of the person column (adds Person to the nesting matrix if supplied).
weight_col	Optional name of a numeric column of finite, nonnegative weights. Zero-weight rows are excluded; at least one weight must be positive. Weighted frequencies are used directly, without rounding or replicating rows.

Details

This is a pure descriptive review of the observed design. It does not affect estimation; fit_mfrm() continues to treat all facets as fixed effects.

Value

A list of class `mfrm_facet_nesting` with:

- `pairwise_table`: one row per facet pair, reporting both directions with `NestingIndex_AinB`, `NestingIndex_BinA`, classification strings, `Direction`, `UsedRows` and `ExcludedRows`.
- `summary`: a one-line summary table with facet counts and whether any non-crossed structure was detected.
- `facets`: the facet vector that was reviewed.

Classification bands

- `"Fully nested"`: nesting index >= 0.99.
- `"Near-perfectly nested"`: 0.95 <= index < 0.99.
- `"Partially nested"`: 0.50 <= index < 0.95.
- `"Crossed"`: index < 0.50.

The direction column records which facet is nested in which, or `"crossed"` when neither direction is above 0.95.

Missing values and weights

Each pair uses rows observed on both columns and having positive weight. Both marginal entropies and both conditional entropies use those same rows. `UsedRows` and `ExcludedRows` report this pair-specific scope; they count original rows, not the sum of weights. Different pairs may use different rows. Multiplying all weights by a positive constant leaves the indices unchanged. Weights are descriptive frequency weights, not a survey-variance correction. The index is undefined (`NA`) when its denominator entropy is zero, including pairs with no usable rows. Missing-value exclusion does not establish that the observed pattern represents the missing ratings.

Interpreting output

A `Direction` value of `"Rater nested in Region"` means that every rater appears in exactly one region (or very close to it). For additive fixed-effects MFRM, this is a concern: the severity of a rater is confounded with region-level variance that the model cannot partition. Consider reporting the nesting direction explicitly and, when relevant, refitting without the nested facet or moving to a hierarchical estimation tool (e.g. `lme4::lmer`, `brms`, `TAM`) to separate the variance components.

`Direction = "crossed"` is the most common reading when both nesting indices are below 0.5; the two facets largely co-occur at multiple combinations, which is the setting Linacre (1989) assumed.

Typical workflow

1. Call `detect_facet_nesting(data, facets)` before fitting.
2. If any pair is flagged as nested or partially nested, review the numeric index and the `LevelsA`/`LevelsB` counts.
3. For downstream reporting, use `analyze_hierarchical_structure()` to bundle this output with ICC and design-effect summaries, which `reporting_checklist()` can record as available report material.

References

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379-423, 623-656. https://web.mit.edu/6.976/www/handout/shannon.pdf

McEwen, M. R. (2018). *The effects of incomplete rating designs on results from many-facets-Rasch model analyses* (Doctoral thesis, Brigham Young University). https://scholarsarchive.byu.edu/etd/6689/

Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.

See Also

`facet_small_sample_review()`, `analyze_hierarchical_structure()`, `compute_facet_icc()`, `compute_facet_design_effect()`, `fit_mfrm()` (see "Fixed effects assumption" in its details).

Examples

```
toy <- load_mfrmr_data("example_core")
nesting <- detect_facet_nesting(toy, c("Rater", "Criterion"))
summary(nesting)

# Synthetic example: raters fully nested within regions.
d <- data.frame(
  Person = rep(paste0("P", formatC(1:20, width = 2, flag = "0")),
               each = 6),
  Rater  = rep(paste0("R", 1:6), 20),
  Region = rep(rep(c("A", "A", "B", "B", "C", "C"), 20)),
  Score  = sample(0:4, 120, replace = TRUE),
  stringsAsFactors = FALSE
)
nest <- detect_facet_nesting(d, c("Rater", "Region"))
```

```
nest$pairwise_table[, c("FacetA", "FacetB",
                        "NestingIndex_AinB", "Direction")]
```

`diagnose_mfrm` *Compute diagnostics for an* `mfrm_fit` *object*

Description

Compute diagnostics for an `mfrm_fit` object

Usage

```
diagnose_mfrm(
  fit,
  interaction_pairs = NULL,
  top_n_interactions = 20,
  whexact = FALSE,
  fit_df_method = "facets",
  diagnostic_mode = c("both", "legacy", "marginal_fit"),
  residual_pca = c("none", "overall", "facet", "both"),
  pca_max_factors = 10L
)

## S3 method for class 'mfrm_diagnostics'
plot(
  x,
  type = c("marginal_fit", "marginal_pairwise", "bubble", "residual_pca"),
  view = NULL,
  ...
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`interaction_pairs`	Optional list of facet pairs.
`top_n_interactions`	Number of top interactions.
`whexact`	ZSTD transformation. `FALSE` (default) uses Wilson-Hilferty with the q/3 term (FACETS `WHEXACT=YES`); `TRUE` omits that term (FACETS `WHEXACT=NO`), where q = sqrt(2/df). The historical logical names are retained. `"legacy_linear"` explicitly requests the former TRUE calculation, (MnSq - 1) * sqrt(df/2), which is not FACETS WHEXACT=NO.
`fit_df_method`	Degrees-of-freedom convention used for fit ZSTD. `"facets"` (default) uses the Wright-Masters fourth-moment approximation `df = 2 / q^2` and caps reported ZSTD at +/-9. `"engine"` retains `DF_Infit = sum(Var * Weight)` and `DF_Outfit`

= sum(Weight). "both" uses engine primary columns and adds *_FACETS companions. Engine transformations require df >= 1; the FACETS route admits df > 0.

diagnostic_mode
: Diagnostic basis to compute: "both" (the current default) computes both the residual/EAP-based stack and the strict latent-integrated first-order marginal-fit companion; "legacy" keeps the residual/EAP-based stack only; "marginal_fit" returns only the marginal-fit companion. The "both" path adds a posterior-integrated pass that typically doubles to quintuples wall-clock time relative to "legacy"; pass "legacy" explicitly when iterating on large designs and only the residual stack is needed. Use "both" for RSM/PCM reporting fits because it enables plot(x, type = "marginal_fit") and plot(x, type = "marginal_pairwise") follow-up.

residual_pca
: Residual PCA mode: "none", "overall", "facet", or "both".

pca_max_factors
: Maximum number of PCA factors to retain per matrix.

x
: An mfrm_diagnostics result from diagnose_mfrm().

type
: "marginal_fit", "marginal_pairwise", "bubble", or "residual_pca".

view
: For marginal fit, "std_residual" (default) or "prop_diff"; for bubbles, "measure" (default) or "infit_outfit"; for PCA, "scree", "parallel_scree", "parallel_excess", or "loadings". Omit it for "marginal_pairwise" and use metric instead.

...
: All views accept draw = FALSE, top_n, and preset. Marginal views accept facet; pairwise plots additionally accept metric = "exact" or "adjacent". Bubbles accept fit_stat ("Infit" or "Outfit"), bubble_size ("SE", "N", or "equal"), facets, include_person = FALSE, and the heuristic fit_range = c(0.5, 1.5). PCA accepts mode, facet, and component, as described in plot.mfrm_residual_pca(). Controls for other views are rejected.

Details

This function computes a diagnostic bundle used by downstream reporting. It calculates element-level fit statistics, approximate facet separation/reliability summaries, residual-based QC diagnostics, and optionally residual PCA for exploratory residual-structure screening.

diagnostic_mode keeps the legacy residual fit path explicit rather than silently replacing it. The legacy path is a compatibility-oriented residual/EAP stack, whereas the strict marginal path targets latent-integrated first-order category counts. When diagnostic_mode = "both", the output includes a diagnostic_basis guide so downstream tables and summaries can distinguish these targets.

Legacy response calculations. Saved MML EAPs are matched to responses by Person identifier. JML retains optimizer-coordinate probabilities for compatibility diagnostics. In obs, PersonMeasure records the ability used, PrimaryPersonMeasure records the primary estimate (possibly infinite), and PersonMeasureBasis identifies the calculation. A finite JML optimizer value is not a finite primary estimate at a boundary. For supported ordinary fits, fitted() instead uses primary JML limits. The posterior/corrected-JML targets have their own documented routes in mfrm_response_diagnostics().

obs$ConditionalVariance retains the actual conditional variance. Legacy Var, StdResidual and StdSq are withheld when that variance is at or below 1e-10. A positive value below this cutoff is

a diagnostic screening convention, not a mathematically undefined Pearson residual; residuals(fit, "pearson") retains representable positive variances within its supported scope. A zero variance makes the Pearson residual undefined. Expected ratings and raw residuals can remain available in either case. `PearsonAvailable`, `ResponseStatus` and `ResponseReason` explain each row, and are retained in score/residual file tables.

Overall and element-level mean squares require every contributing row's standardized residual, variance and positive weight. If any is unavailable, the aggregate and its degrees of freedom remain `NA`; missing residuals are not counted as zero misfit or silently excluded. `FitResponseRows`, `FitAvailableRows` and `FitUnavailableRows` count rows (not weight totals), while `FitStatus` and `FitReason` explain availability. These fields accompany `overall_fit`, `fit` and `measures`. FACETS-style degrees of freedom also require complete fourth moments. Regenerate older diagnostics and exports from the saved fit to obtain these explanations; model refitting is not required.

Marginal expected counts integrate over each Person's posterior conditioned on the same observed responses, holding fitted calibration fixed. They are not expectations from an independent replication or a prior-only population margin. First-order residual scales use `sum(w^2 * p * (1-p))`; pairwise scales use the analogous formula with products of row weights. These scales omit cross-response/opportunity covariance and calibration-parameter uncertainty. They are descriptive screens, not calibrated residual tests.

Missing/invalid contributing probabilities, scores or weights withhold the affected complete-scope aggregate rather than selecting usable rows. Missing standardized residuals and flags remain `NA`. A known cutoff crossing stays flagged when a companion rule is unavailable. `marginal_fit$coverage` records classified/unclassified cells, groups and level pairs. Row/opportunity counts remain visible; zero-weight opportunities contribute no information. Available maxima are accompanied by classification/residual counts; they are not maxima over unavailable values. RMSDs require the complete scope. In PCM and GPCM, step-group summaries retain the declared `step_facet`. Regenerate older marginal diagnostics and downstream summaries/plots/exports from the existing fit and original diagnostic settings; no model refit is required.

Choosing `diagnostic_mode`:

- `"legacy"`: use when continuity with historical residual-based workflows is the priority.
- `"marginal_fit"`: use when you want the strict latent-integrated screen without the extra legacy bundle.
- `"both"`: recommended when you want continuity with the legacy residual stack while making the strict marginal path explicit for `RSM`, `PCM`, and `GPCM` fits.

For `GPCM`, the same generalized partial credit kernel now drives both the residual/probability tables and the strict marginal category-fit companion. Residual-based MnSq summaries should still be read as exploratory screening tools rather than strict Rasch-style invariance tests because discrimination is free, and the strict marginal companion should likewise be treated as a slope-aware screen rather than a finalized inferential test family.

Key fit statistics computed for each element:

- **Infit MnSq**: information-weighted mean-square residual; sensitive to on-target misfitting patterns. Expected value = 1.0.
- **Outfit MnSq**: unweighted mean-square residual; sensitive to off-target outliers. Expected value = 1.0.

- **ZSTD**: Wilson-Hilferty cube-root transformation of MnSq to an approximate standard normal deviate.
- **PTMEA**: within-element point-measure correlation between observed scores and the oriented sum of model measures (`obs$MeasureSum`), including Person rows. A positive value is directionally consistent with the fitted orientation; it is not a confirmatory test.

The MnSq values and the ZSTD values should be read separately. `mfrmr` keeps the fourth-moment df convention by default. The former engine convention remains explicit. This changes some ZSTD screening decisions, not the underlying MnSq. MML residual diagnostics remain EAP plug-in quantities; this standardization does not make them posterior-integrated fit tests. Use `fit_df_method = "both"` to review these two standardization conventions side by side, using engine primary columns and FACETS companion columns.

Residual basis under MML. For `method = "MML"` fits, residuals, MnSq, and ZSTD are computed at the EAP person measures from the marginal model. EAP measures are shrunken toward the population mean, so expected scores – and therefore fit statistics – differ systematically from JMLE-based engines such as FACETS, especially for persons with extreme raw scores. The df conventions above do not remove this difference: it is a residual-basis difference, not a standardization difference. Refit with `method = "JML"` when an external FACETS fit comparison requires a JMLE-style residual basis (see `facets_fit_review()`). ConQuest MML parameter fit also has a different target: residuals are aggregated through the response design for each parameter, then averaged over posterior ability distributions. Its Infit averages a ratio, rather than dividing separately averaged residual sums and variances. Posterior integration of row residuals alone does not reproduce that statistic. Native marginal category-count diagnostics are another target. See the ConQuest technical manual.

Heuristic misfit-screening guidelines (Bond & Fox, 2015):

- MnSq < 0.5: overfit (too predictable; may inflate reliability)
- MnSq 0.5–1.5: productive for measurement
- MnSq > 1.5: underfit (noise degrades measurement)
- $|\mathrm{ZSTD}| \geq 2$: package convention for the approximate-normal review flag; not a calibrated 5\ and repeated screening across elements

When Infit and Outfit disagree, Infit is generally more informative because it downweights extreme observations. Large Outfit with acceptable Infit typically indicates a few outlying responses rather than systematic misfit.

`interaction_pairs` controls which facet interactions are summarized. Each element can be:

- a length-2 character vector such as `c("Rater", "Criterion")`, or
- omitted (`NULL`) to let the function select top interactions automatically.

Residual PCA behavior:

- `"none"`: skip PCA (fastest; recommended for initial exploration)
- `"overall"`: compute overall residual PCA across all facets
- `"facet"`: compute facet-specific residual PCA for each facet
- `"both"`: compute both overall and facet-specific PCA

Overall PCA examines the person × combined-facet residual matrix; facet-specific PCA examines person × facet-level matrices. These summaries are exploratory screens for residual structure, not standalone proofs for or against unidimensionality. Facet-specific PCA can help localise where a stronger residual signal is concentrated.

These residual-PCA summaries are not a DIMTEST/UNIDIM implementation. DIMTEST-style essential-unidimensionality tests work at an item-response layer and require an explicit decision about how many-facet rating data are collapsed, conditioned, or adjusted for rater/task/facet effects. For manuscripts, combine global/element fit, residual PCA, and local-dependence screens, and use limited wording such as "evidence consistent with essential unidimensionality under the specified facet structure" rather than "unidimensionality was established."

`plot()` method: Plot saved ordinary-model diagnostics

Inspect an existing diagnostic result to locate cells or elements needing follow-up. The selected display retains unavailable results and screening limits.

Value

An object of class `mfrm_diagnostics` including:

- `obs`: observed/expected/residual-level table
- `measures`: facet/person fit table (`Infit`, `Outfit`, `ZSTD`, `PTMEA`, `ModelSE`, `RealSE`, `CI_Lower`, `CI_Upper`, `CI_Level`, `CI_Method`). `Fixed = TRUE` identifies location values fixed by anchors or identification constraints. Their sampling SEs and normal intervals are `NA` (not estimated, not a numerical failure). Free levels retain their estimator-specific SEs. Aggregate reliability/separation and facet variability tests are unavailable for facets containing fixed values; response-fit diagnostics remain available. This convention differs from `mfrm_facet_intervals()`, which represents a fixed target's conditional covariance as zero, also without an interval. Neither route propagates uncertainty in supplied anchor values.
- `overall_fit`: overall fit summary
- `fit`: element-level fit diagnostics
- `reliability`: facet-level model/real separation and reliability
- `precision_profile`: one-row summary of the active precision tier and its recommended use
- `precision_review`: package-native checks for SE, CI, and reliability
- `parameter_uncertainty`: MML observed-information uncertainty for structural parameters when available (`steps`, and `GPCM` `slopes` on both log and positive scales), plus covariance status metadata. Step `CIEligible` and `CIUse` retain the source fit's restrictions; non-unit observation-weight bands are diagnostic only. GPCM MML relative slopes use the separate `confint.mfrm_fit()` checks and pointwise log-Wald approximation; inspect `CIEligible`, `CIUse` and `InferenceReview`.
- `facet_precision`: facet-level precision summary by distribution basis and SE mode
- `facets_chisq`: fixed/random facet variability summary
- `interactions`: top interaction diagnostics
- `interrater`: inter-rater agreement bundle (`summary`, `pairs`) including agreement and rater-severity spread indices
- `unexpected`: unexpected-response bundle

- fair_average: adjusted-score reference bundle (reported as unavailable for GPCM)
- displacement: displacement diagnostics bundle
- approximation_notes: method notes for SE/CI/reliability summaries
- diagnostic_basis: guide to the statistical target of each diagnostic path
- fit_standardization: guide to the df convention behind fit ZSTD values
- fit_readiness, fit_readiness_components, and fit_readiness_parameters: the source fit's versioned readiness decision; diagnostic computation does not override global fit readiness. GPCM slope parameter rows additionally retain the output-specific interval decision
- marginal_fit: optional strict marginal-fit companion based on posterior-expected first-order category counts, with classification coverage
- residual_pca_overall: optional overall PCA object
- residual_pca_by_facet: optional facet PCA objects
- replay_inputs: diagnostic settings retained for reproducible export, including fit standardization, interaction selection, and PCA limits

For plot(): An mfrm_plot_data object. Use plot_data() to inspect its tables and notes. The plot does not add unavailable diagnostics or qualify intervals.

Saved diagnostic definitions

New diagnostics carry calculation_version = 2L. Recreate older diagnostics with diagnose_mfrm(fit) before using current reports; no refit is needed. For historical ZSTD use fit_df_method = "engine" and, if applicable, whexact = "legacy_linear". PTMEA, displacement, eligible chi-square df, fair scores and unexpected-response defaults use the corrected definitions.

Reading key components

Practical interpretation often starts with:

- overall_fit: global infit/outfit and degrees of freedom.
- reliability: facet-level model/real separation and reliability. MML uses model-based ModelSE values where available; JML keeps these quantities as exploratory approximations.
- fit: element-level misfit scan (Infit, Outfit, ZSTD).
- unexpected, fair_average, displacement: targeted QC bundles. For GPCM, fair_average is retained with an unavailable status because that compatibility calculation is outside the documented generalized-model contract.
- approximation_notes: method notes for SE/CI/reliability summaries.

Interpreting output

Testlet and shared-rater fits use mfrm_response_diagnostics() instead of this ordinary-model route. Their same-data posterior predictive residual summaries do not inherit ordinary fit cutoffs, ZSTD or p-values.

Start with overall_fit and reliability, then move to element-level diagnostics (fit) and targeted bundles (unexpected, displacement, interrater, facets_chisq). Treat fair_average as available only for the RSM / PCM branch.

Consistent signals across multiple components are typically more robust than a single isolated warning. For example, an element flagged for both high Outfit and high displacement is more concerning than one flagged on a single criterion.

`SE` is kept as a compatibility alias for `ModelSE`. `RealSE` is a fit-adjusted companion defined as `ModelSE * sqrt(max(Infit, 1))`. Reliability tables report model and fit-adjusted indices from observed variance minus mean squared SE, truncated at zero. Fit-adjusted values are not confidence bounds; `JML` entries remain exploratory. Separation, strata, and reliability follow the Wright & Masters (1982) conventions: $G = \mathrm{TrueSD}/\mathrm{RMSE}$, $R = G^2/(1+G^2)$, and $H = (4G+1)/3$.

Tables record finite-estimate counts and the available SEs on those same levels. Non-finite estimates are excluded from the spread and SE summaries. If any finite estimate lacks a valid SE, reliability, separation, strata and error-adjusted spread are unavailable, rather than combining different sets of levels. Excluded levels or incomplete uncertainty prevent a facet summary from supporting formal reporting. A facet containing fixed values also has no aggregate reliability, separation or strata: a fixed anchor's zero conditional variance is not evidence of perfect measurement. Fixing a group mean alone does not fix each group member. For EAP Persons, this separation-based index is distinct from posterior-variance EAP reliability. High rater separation means distinguishable rater measures, not high rater agreement.

Facet SEs from a regularized information matrix or an observation-table fallback remain diagnostic approximations. They are labelled explicitly and cannot authorize ordinary confidence-interval reporting. Numerical convergence is reviewed separately from support for inference; switching from JML to MML does not by itself establish valid SEs or intervals. Recompute older diagnostic objects with `diagnose_mfrm(fit)` before reporting; the existing fit can be used without refitting the model.

Typical workflow

1. Start with `diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")`.
2. Inspect `summary(diag)` and use `diagnostic_basis` to separate legacy residual evidence from strict marginal evidence.
3. If needed, rerun with residual PCA (`"overall"` or `"both"`).

References

- Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press. (G/R/H separation, reliability, and strata formulas summarized in `s_diag$reliability` follow this convention.)
- Wright, B. D., & Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions, 8*(3), 370. (Source for the 0.5-1.5 Infit / Outfit heuristic review interval that `s_diag$key_warnings` and `misfit_thresholds` apply.)
- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press. (FACETS Tables 6 + 7 correspond to the per-facet element measures, fit, and chi-square heterogeneity screen exposed via `s_diag$reliability` and `s_diag$facets_chisq`.)
- Bond, T. G., & Fox, C. M. (2015). *Applying the Rasch model: Fundamental measurement in the human sciences* (3rd ed.). Routledge. (Reference text for the Rasch-family fit conventions exposed by this helper.)
- Linacre, J. M. (2002). What do Infit and Outfit, Mean-square and Standardized mean? *Rasch Measurement Transactions, 16*(2), 878.

- Linacre, J. M. (2026). *A user's guide to Facets Rasch-model computer programs*. Winsteps.com. (WHEXACT / FACETS standardized fit df notes.)

Choosing a view

For `plot()`:

Marginal fit and pairwise plots require strict marginal diagnostics from `diagnose_mfrm()` with `diagnostic_mode = "both"` or `"marginal_fit"` for supported MML RSM/PCM models. Positive residuals indicate more observed category usage or agreement than expected; negative values indicate less. Returned `full_table` and `retention` distinguish omitted and unavailable rows. A flag is a descriptive review rule, not a calibrated test.

Bubble plots place measures against mean-square fit, or Infit against Outfit. Bubble size can represent SE or observation count; neither a review band nor bubble size alone justifies deleting an element.

The PCA convenience route can derive PCA from the diagnostic result. For repeated PCA displays, first save `analyze_residual_pca()` and use its `plot()` method. Parallel-reference generation remains an analysis step.

Expected point-measure correlation under ordinary JML

For ordinary additive RSM/PCM JML without group-mean anchors, `measures$PTMEAExpected` adds a conditional reference beside observed `PTMEA`. Let `x` be the oriented generating measure, `e` the expected score, `v` its conditional model variance, and `N` the sum of frequency weights within the element. The reference is

$$\frac{\mathrm{Cov}_w(x, e)}{\sqrt{\mathrm{Var}_w(x)\{\mathrm{Var}_w(e) + (N-2)\bar{v}_w/N\}}}.$$

Weighted moments use the same observations and normalized frequency weights. `PTMEAExpectedAsymptotic` omits the `(N-2)/N` adjustment. The correction is Linacre's published finite-count reporting approximation; it is not an exact expectation of the sample correlation, a significance test or a confidence interval. Calibration is held fixed. Different administered PCM scales are retained when calculating each observation's expectation and variance.

`PTMEAExpectedN`, `PTMEAExpectedFactor`, `PTMEAExpectedStatus` and `PTMEAExpectedReason` identify the calculation and unavailable states. At `N = 2` the variance term vanishes; this does not establish perfect measurement or fit. Below `N = 2`, the corrected reference is unavailable (FACETS 4.5.0 can instead display a boundary value). The asymptotic reference remains available if its moments exist. Arbitrarily normalizing weights changes the corrected reference: use replication semantics for comparison. Constant predictors, invalid moments, or observations involving unanchored extreme main effects withhold the complete element's expected reference; they are not silently dropped. Fixed finite anchors remain usable.

FACETS 4.5.0 fixed-calibration PCM checks cover missing assignments and unit, integer, fractional and unequal replication weights. These checks establish a reporting convention within that scope, not general diagnostic calibration. Observed `PTMEA` retains its existing definition. MML, corrected JML, GPCM, fitted interactions and group-mean anchors do not receive the new expected-correlation columns. Recreate older diagnostics from the saved fit to obtain them; old saved results are not relabeled or silently recalculated. See Linacre (2008), *The Expected Value of a Point-Biserial (or Similar) Correlation*, Rasch Measurement Transactions, 22:1, 1154, including its later finite-count note: https://www.rasch.org/rmt/rmt221e.htm.

See Also

fit_mfrm(), analyze_residual_pca(), build_visual_summaries(), mfrmr_visual_diagnostics, mfrmr_reporting_and_apa

diagnose_mfrm(), plot.mfrm_residual_pca(), mfrmr_plot_migration

Examples

```
# Long-running workflow; run explicitly when needed.
# example("diagnose_mfrm", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Fit the model
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

# Check model fit and the support for standard errors and intervals
diagnostics <- diagnose_mfrm(fit)
# Check each residual's basis and the coverage of aggregate fit statistics
head(diagnostics$obs[, c("PersonMeasureBasis", "ConditionalVariance",
  "PearsonAvailable", "ResponseReason")])
diagnostics$overall_fit[, c("Infit", "Outfit", "FitUnavailableRows", "FitReason")]
diagnostic_summary <- summary(diagnostics)
diagnostic_summary$decision

diagnostic_summary$key_warnings # Issues to investigate, if present
diagnostic_summary$top_fit      # Most unusual residual-based fit statistics

# Distinguish residual-based checks from marginal model checks
diagnostic_summary$diagnostic_basis[, c("DiagnosticPath", "Status", "Basis")]

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("diagnose_mfrm", run.dontrun = TRUE)
## Not run:
# Each row is one bundled example rating; use ordinary JML for this reference
ratings <- load_mfrmr_data("example_core")
jml_fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                    method = "JML", model = "RSM")
jml_diagnostics <- diagnose_mfrm(jml_fit, residual_pca = "none",
                                 diagnostic_mode = "legacy")
head(jml_diagnostics$measures[, c("Facet", "Level", "PTMEA",
     "PTMEAExpected", "PTMEAExpectedN", "PTMEAExpectedStatus")])
```

```
# Compare observed PTMEA with its conditional reference; neither is a test

## End(Not run)
```

`dif_interaction_table` *Compute interaction table between a facet and a grouping variable*

Description

Produces a cell-level interaction table showing Obs-Exp differences, scaled residuals, and absolute residual mean comparisons for each facet-level x group-value cell.

Usage

```
dif_interaction_table(
  fit,
  diagnostics,
  facet,
  group,
  data = NULL,
  min_obs = 10,
  p_adjust = "holm",
  abs_t_warn = 2,
  abs_bias_warn = 0.5
)

## S3 method for class 'mfrm_dif_interaction'
plot(x, type = "heatmap", ...)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`.
`facet`	Character scalar naming the facet.
`group`	Character scalar naming the grouping column.
`data`	Optional data frame with the group column. If `NULL` (default), the data stored in `fit$prep$data` is used, but it must contain the `group` column. All fitted response columns, including the weight column if used, are required. Subsets and repeated responses follow `analyze_dff()`'s response-matching contract; unmatched rows remain visible in `$cases`.
`min_obs`	Minimum observations per cell. Cells with fewer than this many observations are marked sparse; scaled residuals and magnitude flags are `NA`. Observed and expected score summaries remain available.
`p_adjust`	Retained for compatibility; unused because no p-values are reported. Default `"holm"`.

abs_t_warn	Retained for compatibility; unused because scaled residuals are not t statistics. flag_t is NA.
abs_bias_warn	Threshold for marking the absolute observed-minus-expected average, in score units. Default 0.5. This is a descriptive magnitude rule.
x	A result from dif_interaction_table().
type	The sole view, "heatmap".
...	Heatmap controls below, plus draw = FALSE to return data without drawing. This input accepts metric = "obs_exp" or "t"; "contrast" requires a result from analyze_dff().

Details

This function uses observation-level residuals computed from the fitted model rather than re-estimating it. For each facet-level x group-value cell, it computes:

- N: number of observations in the cell
- NPersons: number of distinct Persons in the cell, not an independence certificate
- ObsScore: sum of observed scores
- ExpScore: sum of expected scores
- ObsExpAvg: mean observed-minus-expected difference
- Var_sum: sum of model variances
- StdResidual: (ObsScore - ExpScore) / sqrt(Var_sum)
- t, df, p_value, p_adjusted, flag_t: NA, retained for compatibility

plot() method: Plot a differential-functioning interaction grid

Display saved residual means by facet level and group. This is a descriptive screen for follow-up; residual differences do not isolate differential functioning.

Value

Object of class mfrm_dif_interaction with:

- table: tibble with per-cell statistics and flags.
- cases: requested-row matching and exclusion audit.
- summary: tibble summarizing flagged and sparse cell counts.
- gpcm_boundary: for GPCM fits, a capability-boundary table.
- config: list of analysis parameters.

For plot(): An mfrm_plot_data object with data$matrix, data$pairs, the selected metric and retained interpretation notes. Use plot_data(p, "matrix") to extract the matrix from a saved plot p.

When to use this instead of analyze_dff()

Use dif_interaction_table() when you want cell-level screening for a single facet-by-group table. Use analyze_dff() when you want group-pair comparisons. Neither residual output tests differential functioning.

Further guidance

For plot selection and follow-up diagnostics, see mfrmr_visual_diagnostics.

Interpreting output

- `$table`: the full interaction table with one row per cell.
- `$cases`: requested-row matching and exclusion audit, also kept by `summary()`.
- `$summary`: overview counts of flagged and sparse cells.
- `$config`: analysis configuration parameters.
- `$gpcm_boundary`: for `GPCM` fits, a capability-boundary table marking the table as caveated DFF screening evidence.
- `flag_bias` records `|ObsExpAvg| > abs_bias_warn` in score units. It does not establish differential functioning. `flag_t` is unavailable.
- Sparse cells (N < min_obs) have `sparse = TRUE` and unavailable scaled residuals and magnitude flags. The score means and counts are retained.

GPCM boundary

For `GPCM`, the interaction table uses the fitted slope-aware expected-score/residual scale and should be reported as screening evidence, not as a standalone fairness, invariance, or operational subgroup decision.

Residual means can differ even when response parameters are the same between groups. These summaries provide no p-values or t-based decisions. Recompute older saved tables with this function using the fitted model; no refit is needed.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Run `dif_interaction_table(fit, diag, facet = "Rater", group = "Gender", data = df)`.
3. Inspect `$table` for flagged cells.
4. Visualize with plot(x, type = "heatmap").

Interpretation

For `plot()`:

Positive `"obs_exp"` values mean observed scores exceed model expectations; negative values mean they fall below expectations. These values are in score units, not logits. The `"t"` metric is a scaled residual without a calibrated test or p-value. Color and optional threshold outlines are descriptive; an unavailable cell is not a zero residual.

Heatmap appearance

- `show_values = TRUE` labels cells; `value_digits = 2` controls rounding.
- `flag_threshold = NULL` adds no flags. A non-negative value outlines cells at or above that absolute value; `flag_color = "black"` sets the border.
- `scale_limit = NULL` uses the observed range. A positive value fixes symmetric color limits for comparing plots. It does not change saved values.

See Also

analyze_dff(), analyze_dif(), plot(x, type = "heatmap"), dif_report(), estimate_bias()

dif_interaction_table(), plot.mfrm_dif(), mfrmr_plot_migration

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("dif_interaction_table", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 300)
diag <- diagnose_mfrm(fit, residual_pca = "none")
int <- dif_interaction_table(fit, diag, facet = "Rater",
                             group = "Group", data = toy, min_obs = 2)
int$summary
head(int$table[, c("Level", "GroupValue", "ObsExpAvg", "flag_bias")])

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_dif_interaction", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 300)
diag <- diagnose_mfrm(fit, residual_pca = "none")
int <- dif_interaction_table(fit, diag, facet = "Rater",
                             group = "Group", data = ratings, min_obs = 2)
heat <- plot(int, type = "heatmap", metric = "obs_exp", draw = FALSE)
dim(heat$data$matrix)
# Obs-Exp is in score units, not logits. A scaled residual
# (metric = "t") is descriptive, not a calibrated t-test.

## End(Not run)
```

---

dif_report *Generate a differential-functioning interpretation report*

Description

Produces APA-style narrative text interpreting the results of a differential- functioning analysis or interaction table. For `method = "refit"`, the report summarises linked screening contrasts and whether conditional plug-in uncertainty was available. For `method = "residual"`, it summarises the availability and meaning of residual comparisons, without testing differential functioning or classifying comparisons as positive or negative.

Usage

```
dif_report(dif_result, ...)
```

Arguments

dif_result	Output from analyze_dff() / analyze_dif() (class mfrm_dff with compatibility class mfrm_dif) or dif_interaction_table() (class mfrm_dif_interaction).
...	Reserved for generic compatibility.

Details

When dif_result is an mfrm_dff/mfrm_dif object, the report is based on the pairwise differential-functioning contrasts in $dif_table. When it is an mfrm_dif_interaction object, the report uses the cell-level statistics and flags from $table.

Refit differences are descriptive on a linked logit scale when subgroup calibrations retain the required anchors. Their separate-subgroup plug-in standard errors condition on those anchors and omit baseline-anchor uncertainty and cross-refit covariance, so the report does not assign ETS labels or present refit rows as formal inference. The residual method also returns descriptive group differences without tests or binary classifications.

Value

Object of class mfrm_dif_report with narrative, counts, large_dif, gpcm_boundary, and config.

Interpreting output

- $narrative: character scalar with the full narrative text.
- $counts: named integer vector of method-appropriate counts.
- $large_dif: an empty compatibility table for current refit output, or an empty table for residual comparisons. Interaction reports include only cells above the requested absolute residual mean threshold, in score units.
- $gpcm_boundary: for GPCM inputs, a capability-boundary table marking the narrative as caveated DFF screening output.
- $config: analysis configuration inherited from the input.
- $comparison_review: when supplied to analyze_dff(), its separate prospective interaction review, retained without treating its states as classifications of the residual/refit contrasts. Also retained by summary().

GPCM boundary

If the input comes from a GPCM fit, the narrative includes a GPCM note and the returned report carries gpcm_boundary. Treat the text as slope-aware screening/reporting support, not as a standalone fairness, invariance, or operational subgroup decision.

Typical workflow

1. Run `analyze_dff()` / `analyze_dif()` or `dif_interaction_table()`.
2. Pass the result to `dif_report()`.
3. Print the report or extract `$narrative` for inclusion in a manuscript.

References

The narrative caveat about distinguishing construct-relevant variation from unwanted measurement bias is grounded in:

- Eckes, T. (2011). *Introduction to Many-Facet Rasch Measurement: Analyzing and Evaluating Rater-Mediated Assessments*. Frankfurt am Main: Peter Lang. ISBN 978-3-631-61350-4.
- McNamara, T., & Knoch, U. (2012). The Rasch wars: The emergence of Rasch measurement in language testing. *Language Testing*, 29(4), 555–576. doi:10.1177/0265532211430367

See Also

`analyze_dff()`, `analyze_dif()`, `dif_interaction_table()`, plot(x, type = "heatmap"), `build_apa_outputs()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("dif_report", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
  method = "JML", model = "RSM", maxit = 300
)
diag <- diagnose_mfrm(fit, residual_pca = "none")
dif <- analyze_dff(fit, diag, facet = "Rater", group = "Group", data = toy)
rpt <- dif_report(dif)
cat(rpt$narrative)

## End(Not run)
```

`displacement_table` *Compute displacement diagnostics for facet levels*

Description

Compute displacement diagnostics for facet levels

Usage

```
displacement_table(
  fit,
  diagnostics = NULL,
  facets = NULL,
  anchored_only = FALSE,
  abs_displacement_warn = 0.5,
  abs_t_warn = 2,
  top_n = NULL
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
facets	Optional subset of facets.
anchored_only	If TRUE, keep only directly/group anchored levels.
abs_displacement_warn	Absolute displacement warning threshold.
abs_t_warn	Absolute displacement t-value warning threshold.
top_n	Optional maximum number of rows to keep after sorting.

Details

Displacement is computed as a one-step Newton update: sum(residual) / sum(information) for each facet level. This approximates how much a level would move if constraints were relaxed.

Value

A named list with:

- table: displacement diagnostics by level
- summary: one-row summary
- thresholds: applied thresholds

Interpreting output

- table: level-wise displacement and flag indicators.
- summary: count/share of flagged levels.
- thresholds: displacement and t-value cutoffs.

Large absolute displacement in anchored levels suggests potential instability in anchor assumptions.

Typical workflow

1. Run `displacement_table(fit, anchored_only = TRUE)` for anchor checks.
2. Inspect `summary(disp)` then detailed rows.
3. Visualize with plot(x, type = "displacement").

Output columns

The `table` data.frame contains:

Facet, Level Facet name and element label.

Displacement One-step Newton displacement estimate (logits).

DisplacementSE Standard error of the displacement.

DisplacementT Displacement / SE ratio.

Estimate, SE Current measure estimate and its standard error.

N Number of observations involving this level.

AnchorValue, AnchorStatus, AnchorType Anchor metadata.

Flag Logical; `TRUE` when displacement exceeds thresholds.

Saved-result reuse

The returned `fit_identity` records the fitted analysis for reuse with reports and exports. Recreate older results from the saved fit and matching diagnostics using the original screening options; no model refit is needed. See mfrmr_compatibility_layer for migration instructions.

See Also

`diagnose_mfrm()`, `unexpected_response_table()`, `fair_average_table()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("displacement_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
disp <- displacement_table(fit, anchored_only = FALSE)
summary(disp)
p_disp <- plot(disp, draw = FALSE)
p_disp$data$plot

## End(Not run)
```

`draw_mfrm_resamples` *Draw observed-data MFRM resamples*

Description

Draw observed-data MFRM resamples

Usage

```
draw_mfrm_resamples(spec, keep_data = TRUE)
```

Arguments

`spec`	Output from `build_mfrm_resampling_spec()`.
`keep_data`	Logical; if `TRUE`, return a list of replicate data frames. If `FALSE`, return manifest tables only.

Details

See `build_mfrm_resampling_spec()` for sampling assumptions, top-up, label handling and saved-specification compatibility. Sampling keeps each selected Person's entire rating cluster and re-keys every instance when drawing with replacement. Other facet labels are retained.

Value

An object of class `mfrm_resamples` with:

- `samples`: replicate data frames with original Person, draw-unit, stratum and top-up metadata, or an empty list when `keep_data = FALSE`.
- `manifest`: replicate seeds, sampling scheme, base/added/final cluster counts, original Person counts, rating-row counts and coverage checks.
- `stratum_manifest`: `BasePersonClusters` before top-up, `TopupPersonClusters`, final `SelectedPersonClusters` and `SelectedOriginalPersons`, and final `Rows` for each stratum.
- `preserve_manifest`: observed coverage of requested facet levels.
- `spec` and `terminology`: inputs, sampling assumptions and interpretation. `summary()` retains full manifests as well as compact tables. For older results with top-up, pre-top-up stratum counts are retained as base counts; unrecorded final cluster counts are `NA`, with a note, while final row counts remain available. This does not redraw old samples.

See Also

`build_mfrm_resampling_spec()`

Examples

```
# Same-size Person-cluster bootstrap within the observed Group strata.
ratings <- load_mfrmr_data("example_operational")
spec <- build_mfrm_resampling_spec(
  ratings, person = "Person", facets = c("Rater", "Criterion"),
  score = "Score", strata = "Group", reps = 2,
  design = "stratified_bootstrap", sample_fraction = 1,
  topup_preserve_facets = FALSE, seed = 99
)
draws <- draw_mfrm_resamples(spec, keep_data = FALSE)
summary(draws)$overview
```

`ej2021_data` *Legacy synthetic MFRM datasets inspired by Eckes and Jin (2021)*

Description

Synthetic many-facet rating datasets in long format. All datasets include one row per observed rating.

Format

A data.frame with 5 columns:

Study Study label (`"Study1"` or `"Study2"`).

Person Person/respondent identifier.

Rater Rater identifier.

Criterion Criterion facet label.

Score Observed category score.

Details

Available data objects:

- `ej2021_study1`
- `ej2021_study2`
- `ej2021_combined`
- `ej2021_study1_itercal`
- `ej2021_study2_itercal`
- `ej2021_combined_itercal`

Naming convention:

- `study1` / `study2`: separate simulation studies
- `combined`: row-bind of study1 and study2
- `_itercal`: legacy synthetic sensitivity variant. These objects can differ in observed rows as well as scores and should not be interpreted as a controlled one-parameter recalibration.

Use `load_mfrmr_data()` for programmatic selection by key.

Data dimensions

Dataset	Rows	Persons	Raters	Criteria
study1	1842	307	18	3
study2	3287	206	12	9
combined	5129	307	18	12
study1_itercal	1842	307	18	3
study2_itercal	3341	206	12	9
combined_itercal	5183	307	18	12

Score range: 1–4 (four-category rating scale).

For the combined rows, `Persons` and `Raters` count unique raw labels. Treating the two Study labels as distinct namespaces would instead give 513 person labels and 30 rater labels, but would leave two unlinked components.

Provenance and limits

These are legacy synthetic datasets whose stored responses reproduce the dimensions described above. The exact response-generation code and random seed are not available, so the objects must not be used as parameter-recovery evidence or as evidence for a particular generating distribution. The separately stored `_itercal` objects can differ in observed rows as well as scores; they are legacy variants, not empirical calibration standards or a controlled one-parameter recalibration.

Combined-data caution

The `combined` objects reuse `P001`–`P206` and `R01`–`R12` across the two study labels. A combined analysis is meaningful only when those labels encode an intended cross-study identity and an explicit anchor or linking design establishes a common scale. Prefixing `Person` and `Rater` by `Study` removes accidental label collisions, but it creates disconnected study components and does not by itself establish a common scale. Analyze the studies separately unless the linking design has been specified and reviewed. The combined datasets are not beginner workflow examples or direct-fit examples.

Interpreting output

The study-specific datasets are in long format and can be passed to `fit_mfrm()` after confirming column-role mapping. The `combined` objects require a reviewed identity and anchor/linking design before a joint fit; row-binding or prefixing identifiers alone does not create a common scale.

Typical workflow

1. Inspect available datasets with `list_mfrmr_data()`.
2. Load one dataset using `load_mfrmr_data()`.
3. Fit and diagnose with `fit_mfrm()` and `diagnose_mfrm()`.

Source

Simulated for this package with design settings informed by Eckes and Jin (2021). The Eckes & Jin (2021) Method section reports the following design parameters that motivated the synthetic versions shipped here: Study 1 had 307 examinees (149 males, 158 females), 18 raters (4 males, 14 females), and 3 criteria (global impression, task fulfillment, linguistic realization) on a 4-category rating scale (TDN levels rescored 1-4); Study 2 had 206 examinees (66 males, 140 females), 12 raters (1 male, 11 females), and 9 criteria on the same 4-category scale. The packaged datasets reproduce these (examinees, raters, criteria, categories) shapes but use simulated responses, so they are not the real TestDaF data.

References

Eckes, T., & Jin, K.-Y. (2021). Measuring rater centrality effects in writing assessment: A Bayesian facets modeling approach. *Psychological Test and Assessment Modeling, 63*(1), 65–94.

Examples

```
data("ej2021_study1", package = "mfrmr")
head(ej2021_study1)
table(ej2021_study1$Study)
```

`estimate_all_bias` *Estimate bias across multiple facet pairs*

Description

Estimate bias across multiple facet pairs

Usage

```
estimate_all_bias(
  fit,
  diagnostics = NULL,
  pairs = NULL,
  include_person = FALSE,
  drop_empty = TRUE,
  keep_errors = TRUE,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001
)

## S3 method for class 'mfrm_bias_collection'
summary(object, ...)

## S3 method for class 'mfrm_bias_collection'
```

```
print(x, digits = 3, ...)

## S3 method for class 'summary.mfrm_bias_collection'
print(x, digits = 3, ...)

## S3 method for class 'mfrm_bias_collection'
plot(
  x,
  pair = NULL,
 type = c("overview", "scatter", "ranked", "heatmap", "abs_t_hist", "facet_profile"),
  draw = TRUE,
  preset = "standard",
  ...
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are computed with residual_pca = "none". Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
pairs	Optional list of facet specifications. Each element should be a character vector of length 2 or more, for example list(c("Rater", "Criterion"), c("Task", "Criterion")). Explicit specifications may include canonical "Person" regardless of include_person. When NULL, all 2-way combinations of modeled facets are used. Each combination may appear only once, regardless of facet order.
include_person	If TRUE and pairs = NULL, include "Person" in the automatically generated pair set. "Person" denotes the source column supplied as person to fit_mfrm() and remains opt-in because these cells condition on fitted JML person measures or MML EAP scores.
drop_empty	If TRUE, omit empty bias tables from by_pair while still recording them in the summary table.
keep_errors	If TRUE, also retain per-pair error rows in the returned errors table. Either way, errors remain in summary$Error and do not stop the batch.
max_abs	Passed to estimate_bias().
omit_extreme	Passed to estimate_bias().
max_iter	Passed to estimate_bias().
tol	Passed to estimate_bias().
object	A result from estimate_all_bias().
...	For plot(): With pair, additional arguments to plot.mfrm_bias(), such as top_n or sort_by. The overview rejects additional arguments. For summary(): Reserved; additional arguments are rejected.
x	A result from estimate_all_bias().

digits	Number of significant digits used for printing; calculations and returned tables retain their precision.
pair	An exact Interaction label from summary(x)$overview, or its row number. NULL (default) shows all requested pairs.
type	"overview" shows signed cell estimates grouped by pair. With pair supplied, the default becomes "heatmap"; "scatter", "ranked", "abs_t_hist" and "facet_profile" select the corresponding plot.mfrm_bias() view.
draw	Draw the figure. FALSE returns its data without opening a device.
preset	A package plot preset, for example "standard" or "monochrome".

Details

This function orchestrates repeated calls to estimate_bias() across multiple facet pairs and returns a consolidated bundle.

Bias/interaction in MFRM refers to a systematic departure from the additive model for a specific combination of facet elements (e.g., a particular rater is unexpectedly harsh on a particular criterion). See estimate_bias() for the mathematical formulation.

When pairs = NULL, the function builds all 2-way combinations of modelled facets automatically. For a model with facets Rater, Criterion, and Task, this yields Rater×Criterion, Rater×Task, and Criterion×Task. With include_person = TRUE, Person×facet pairs are prepended in the fit's facet order. They are conditional plug-in likelihood screens, not jointly fitted Person×facet terms. A single warning records this boundary for the batch.

The summary table aggregates results across pairs:

- Rows: number of interaction cells estimated
- Points: cells with an available bias estimate (including bounded optima; inspect BiasStatus in the combined cells)
- ScreenAvailable, ScreenUnavailable: cells with/without a usable conditional screening statistic
- ScreenPositive: count of available cells with $|t| \geq 2$; NA when no screening statistic is available, including failed pairs
- Significant: backward-compatible alias for ScreenPositive
- MeanAbsBias: average absolute available bias magnitude (native ability units)
- Status: available, partial, unavailable, empty, or failed; available refers to conditional screening, not formal inference
- FormalInferenceEligible / PrimaryReportingEligible: always FALSE; these conditional plug-in interaction statistics are screening evidence

Per-pair failures (e.g., insufficient data for a sparse pair) are captured in errors rather than stopping the entire batch. There is no collection-wide multiplicity correction. Cells and pairs reuse the same observations and fitted calibration. Adjusting their screening tail areas would not establish formal family-wise error or false-discovery control. A screen-positive result is a prompt for review, not proof of unfairness; no screen-positive cells does not establish absence of interaction. Search-boundary points remain visible, with missing screening statistics and the ScreenStatus/ScreenReason returned by estimate_bias().

summary() method: Summarize all retained bias screens

For summary(): No model, diagnostics or cell calculation is repeated. The overview counts finite bias estimates from successful cell optimizations separately from cells with a finite screening statistic and positive finite conditional SE. ScreenPositive is missing when no statistic can be screened, rather than reporting zero. Failed pairs remain visible even when the original batch used keep_errors = FALSE. Older collections without retained tables are labeled not_retained; they are not recalculated.

A positive bias shift increases expected coded scores conditional on the fitted model. It does not necessarily mean better performance or unfair leniency. Baseline calibration and Person uncertainty are held fixed. Different pairs use overlapping data, so do not add their observed counts to obtain a study sample size. The overview's mean absolute bias describes magnitude only; use the signed cell estimates to study direction. Counts cover retained estimate_bias() cells: unobserved combinations and cells omitted by the chosen screening options are not reconstructed. No multiplicity-adjusted confirmatory decisions are supplied.

plot() method: Explore bias across facet pairs

For plot(): The overview plots every finite retained bias estimate, without ranking or a significance cutoff. Positive shifts increase expected coded scores under the fitted model; score meaning depends on the rubric. Vertical spreading only separates cells visually and has no statistical meaning.

Filled circles have a usable conditional screening statistic; open circles retain a point without one. These symbols indicate availability, not whether a cell is flagged. Pair labels report usable statistics / retained cells and the range of observed row counts per cell (n), not independent Person counts. Pairs share observations. Empty, failed and unavailable pairs remain labeled, without substituting a point at zero. The vertical zero line means no conditional shift, not a threshold for practical importance.

No model or diagnostics are recalculated. The overview supplies no intervals, multiplicity correction or formal bias test. Review the individual cell counts, optimizer status and subject-matter explanations; selecting a pair after this overview does not make its subsequent analysis confirmatory. Titles and interpretation notes are returned outside the figure.

Value

A named list with class mfrm_bias_collection.

For summary(): summary() returns a summary.mfrm_bias_collection list: overview has one row per requested interaction, cells combines retained cell tables with an Interaction identifier and explicit PointAvailable, ScreenAvailable and ScreenPositive fields, errors lists failed pairs, and optimization_failures retains cell-level optimizer records. Original cell columns, including facet levels, observed counts, signed Bias Size, conditional S.E., t and any GPCM profile status, are preserved. settings, fit_identity and notes retain provenance and interpretation. Print methods return their input invisibly.

For plot(): Invisibly, an mfrm_plot_data object. For the overview, plot_data() provides table (all combined cells, including unavailable rows), overview, errors, optimization_failures, fit_identity, settings, axis labels and notes. as_ggplot() reproduces the overview. A selected pair returns the ordinary bias plot with collection_pair and fit_identity provenance added.

Output

The returned object is a bundle-like list with class `mfrm_bias_collection` and components such as:

- `summary`: one row per requested interaction
- `by_pair`: named list of successful `estimate_bias()` outputs
- `errors`: per-pair error log
- `settings`: resolved execution settings
- `primary`: first successful bias bundle, useful for downstream helpers

Typical workflow

1. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`. For `RSM` / `PCM` reporting runs, prefer `method = "MML"` plus `diagnostic_mode = "both"` in the diagnostics call.
2. Run `estimate_all_bias()` to compute the requested multi-pair interactions.
3. Inspect `summary(bias_all)$overview` for availability and magnitude, then `summary(bias_all)$cells` for all retained cells and their observed counts. Use `plot.mfrm_bias_collection()` for the overview or a named pair.
4. Use `report <- mfrm_report(bias_all)` to retain the full search, failed pairs and all cells in one report; `export_summary_appendix()` saves its tables. Supply `study_context` to state rubric endpoints and score direction. The `by_pair` list also feeds `reporting_checklist()` or `facet_quality_dashboard()`.
5. For a substantively justified joint interaction model, use `facet_interactions` in `fit_mfrm()` and `compare_mfrm()` on compatible fits. Selecting terms from these screens makes same-data comparisons exploratory; it does not create a prespecified confirmatory test.

Saved-result reuse

The returned `fit_identity` records the fitted analysis for reuse with reports and exports. Recreate older results from the saved fit and matching diagnostics using the original screening options; no model refit is needed. See mfrmr_compatibility_layer for migration instructions.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`estimate_bias()`, `mfrm_report()`, `compare_mfrm()`, `reporting_checklist()`, `facet_quality_dashboard()`

`estimate_all_bias()`, `plot.mfrm_bias_collection()`, `mfrm_report()`, `estimate_bias()`

`estimate_all_bias()`, `summary.mfrm_bias_collection()`, `plot.mfrm_bias()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("estimate_all_bias", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML")
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
bias_all <- estimate_all_bias(fit, diagnostics = diag)
summary(bias_all)
head(summary(bias_all)$cells)
report <- mfrm_report(bias_all, study_context = c(
  score_direction = "Higher coded scores indicate better performance"))
report$tables$overview
head(report$tables$bias_cells)

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("summary.mfrm_bias_collection", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "MML")
screens <- estimate_all_bias(fit)
review <- summary(screens)
review$overview
head(review$cells[, c("Interaction", "Bias Size", "ObsN", "ScreenAvailable")])

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_bias_collection", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "MML")
```

```
screens <- estimate_all_bias(fit)
p <- plot(screens)
plot_data(p, "overview")
plot(screens, pair = "Rater x Criterion", type = "ranked")

## End(Not run)
```

`estimate_bias` *Estimate bias and interaction screening terms*

Description

Estimate bias and interaction screening terms

Usage

```
estimate_bias(
  fit,
  diagnostics,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  xtreme = 0
)

## S3 method for class 'mfrm_bias'
summary(object, digits = 3, top_n = 10, p_cut = 0.05, ...)

## S3 method for class 'mfrm_bias'
plot(
  x,
  type = c("scatter", "ranked", "heatmap", "abs_t_hist", "facet_profile"),
  ...
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
`facet_a`	First facet name. Provide together with `facet_b` for the classic pairwise 2-way interaction. Ignored when `interaction_facets` is supplied. The canonical `"Person"` role is accepted explicitly.

facet_b	Second facet name. See facet_a.
interaction_facets	Character vector of two or more facets to model as one interaction effect. When supplied, this takes precedence over facet_a/facet_b. Use this form (rather than facet_a/facet_b) whenever you want 3+ way interactions, since facet_a/facet_b is restricted to the pairwise case. The canonical "Person" role may be supplied explicitly; see the Person-screening boundary below.
max_abs	Bound for absolute bias size.
omit_extreme	Omit extreme-only elements. In ordinary additive RSM/PCM JML, directly fixed finite anchors are retained even with all-low/all-high responses. Their cell bias may still have no finite unconstrained optimum.
max_iter	Maximum number of repeated cell-estimation passes recorded in iteration; the baseline calibration stays fixed throughout.
tol	Between-pass change tolerance for the iteration log. This does not control the tolerance of the bounded scalar likelihood optimizer.
xtreme	Default 0 retains bounded extreme-cell estimates with SEs withheld. A positive correction (usually .5 for FACETS bias reporting) requests finite total-score extrapolation for ordinary additive RSM/PCM JML without group-mean anchors; it must be smaller than the score range.
object	Output from estimate_bias().
digits	Number of digits for printed numeric values.
top_n	Number of strongest bias rows to keep.
p_cut	Tail-area cutoff used for counting screen-positive rows.
...	For plot(): Named controls below. draw = FALSE returns the plot data without drawing. Controls do not change the fitted calibration. For summary(): Reserved for generic compatibility.
x	A result from estimate_bias().
type	"scatter", "ranked", "heatmap", "abs_t_hist", or "facet_profile".

Details

This helper supports RSM, PCM, and one-slope-family GPCM fits. Two-slope-family GPCM fits are not currently supported. It estimates an additional cell shift relative to the fitted baseline, holding that baseline fixed. For joint estimation of main effects and a two-facet interaction in RSM / PCM, use the facet_interactions argument of fit_mfrm(). For the supported fixed-population RSM/PCM MML route, mfrm_facet_intervals() with interaction = "FacetA:FacetB" returns joint-information cell intervals. They estimate a different target and do not replace or relabel this fixed-baseline screening result.

Bias (interaction) in an additive MFRM refers to a systematic departure from the main-effects model: a specific rater-criterion (or higher-order) combination produces scores that are consistently higher or lower than predicted by the main effects alone. For example, Rater A might be unexpectedly harsh on Criterion 2 despite being lenient overall.

Mathematically, the bias term b_{jc} for rater j on criterion c modifies the linear predictor:

$$\eta_{njc} = \theta_n - \delta_j - \beta_c + b_{jc}$$

Here the example uses negatively oriented rater and criterion facets. The returned `Bias Size` is always an additive shift in the score predictor: positive values increase expected scores; negative values decrease them. If the fitted model already contains `facet_interactions`, those fitted interactions remain part of the fixed baseline. The new shift is a residual departure from that model, not a re-estimate of its interaction. Each cell's shift is estimated by maximizing its conditional likelihood within `[-max_abs, max_abs]`, holding fitted person measures, facet values, steps and slopes fixed. For `RSM` / `PCM`, an interior solution satisfies the weighted score equation:

$$\sum_n w_{njc}\{X_{njc} - E_{njc}(b_{jc})\} = 0.$$

With unit weights, $w_{njc} = 1$. The conditional information gives $\mathrm{SE}(\hat{b}) = [\sum_n w_{njc}\mathrm{Var}_{njc}(\hat{b})]^{-1/2}$. For `GPCM`, the score and information include the observation's slope and squared slope, respectively, and conditional profile-likelihood columns are also returned. `Bias Size` is in the model's native ability units: in `RSM` / `PCM` it equals the change in adjacent-category log odds, whereas in `GPCM` that change is ab for an observation with slope a. The same bias value can therefore imply different probability changes across observations. All these quantities are plug-in screening evidence: JML person measures or MML EAP scores and the calibration parameters are treated as known. Their estimation uncertainty, including uncertainty in supplied anchors, is not propagated. A fixed rater severity can therefore have a nonzero cell bias and positive bias SE: the fixed severity and the cell's additional shift are different quantities. `BiasStatus` identifies an interior solution or an optimum at either search bound. A bounded point remains available, but its `S.E.`, `t`, `Prob.` and GPCM profile-inference columns are withheld at a boundary. Inspect `ScreenStatus` and `ScreenReason`: a bound can mean an insufficient search range or an all-low/all-high cell with no finite unconstrained optimum. Increasing the range does not resolve the latter.

With `xtreme > 0`, an all-low/all-high cell's weighted total is moved inward by `xtreme` times its smallest positive weight. A root strictly inside `[-max_abs, max_abs]` yields `BiasStatus = "extreme_adjusted"`, finite `Bias Size` and conditional information SE; `PrimaryBias` retains the infinite cell MLE and `ExtremeAdjustment` records the total correction. Otherwise the boundary remains unavailable for screening. Observations involving any unanchored extreme main-effect element are excluded in this route even when `omit_extreme = FALSE`; finite fixed anchors remain usable. This prevents main-effect display replacements from becoming calibration. The default `xtreme = 0` also retains directly fixed finite anchors in ordinary additive RSM/PCM JML; extreme cells then keep bounded points with screening SEs and tail areas withheld. Group-mean constraints are not direct anchors. The native FACETS `Xtreme = main,bias` has two separate corrections: this function's `xtreme` corresponds to the second value (normally `.5`), not the first (normally `.3`). Executed FACETS 4.5.0 Person-by-Rater examples retain fixed extreme Persons and agree with the conditional cell calculation on the same main calibration. This does not establish main-calibration parity. If a low cell's baseline expectation is already below the inward correction, its adjusted bias can be positive despite a negative observed-minus-expected score difference (and conversely at the upper endpoint). Inspect `PrimaryBias`, `ExtremeAdjustment` and `Obs-Exp Average` together; the adjusted sign describes the amended total, not a reversal of the observed departure. Extrapolated tail areas, summaries and pairwise contrasts are reporting conventions, not regular-MLE inference, coverage guarantees or fairness tests.

ConditionalInformation uses every cell row, including small positive conditional variances; the residual-screening cutoff does not remove their information. Incomplete/nonpositive information withholds the SE and screening statistic while retaining a computed point. Infit/Outfit retain the legacy variance cutoff (1e-10) and require all contributing rows; FitResponseRows, FitAvailableRows, FitUnavailableRows, FitStatus and FitReason explain withheld complete-cell fit summaries.

- For two-way mode, use facet_a and facet_b (or interaction_facets with length 2).
- For higher-order mode, provide interaction_facets with length >= 3.
- A Person-involving result is a conditional plug-in likelihood screen. It holds fitted JML person measures or MML EAP scores and the other fitted parameters fixed; it is not a jointly fitted Person-by-facet interaction and does not support formal inference or a cell-level fairness decision. The function emits a classed warning when "Person" is requested.

summary() method: Summarize an mfrm_bias object in a user-friendly format

For summary(): This method returns a compact interaction-bias summary:

- interaction facets/order and analyzed cell counts
- effect-size profile (|bias| mean/max, screen-positive cell count)
- fixed-effect chi-square block
- iteration-end convergence indicators
- top rows ranked by absolute t

plot() method: Plot saved bias-interaction results

Inspect the pattern of conditional interaction estimates from estimate_bias(). Start with ranked interactions, then use other views to understand the pattern.

For plot(): Visualization front-end for bias_interaction_report() with multiple views. With draw = FALSE, the returned plot data include plot_long, plot_annotations, flag_summary, and plot_settings in addition to the view-specific ranked_table, scatter_data, facet_profile, and heatmap components. Use these fields when rebuilding the same screening view in ggplot2, plotly, Quarto, or a dashboard. Saved report thresholds determine flags, reference lines and returned notes; recreate bias_interaction_report() from the saved bias object to change them. Scatter and ranked views support as_ggplot(); the other views require their base renderer or custom graphics. Conditional bounds and their status remain separate from calibration-inclusive uncertainty.

Value

An object of class mfrm_bias with:

- table: interaction rows with effect size, SE, screening t/p metadata, reporting-use flags, fit columns, and GPCM profile-likelihood columns when available
- summary: compact summary statistics
- chi_sq: fixed-effect chi-square style screening summary
- facet_a, facet_b: first two analyzed facet names (legacy compatibility)
- interaction_facets, interaction_order, interaction_mode: full interaction metadata

- person_interaction, person_source_column, person_interaction_note: Person-screening provenance and interpretation metadata
- iteration: iteration history/metadata
- orientation_review: facet-orientation sign-consistency review table
- mixed_sign: logical flag indicating whether bias-size signs flip across facets in a way that complicates direction interpretation
- direction_note: one-line interpretive note describing the dominant bias direction (empty when not applicable)
- recommended_action: one-line recommended-action label routing the user to the appropriate follow-up helper
- inference_tier: summary label indicating that the bias rows are intended for screening and follow-up review
- optimization_failures: per-cell record of any inner-loop optimizer failures encountered while estimating the bias parameters; empty when every cell converged cleanly

For summary(): An object of class summary.mfrm_bias with:

- overview: interaction facets/order, cell counts, and effect-size profile
- chi_sq: fixed-effect chi-square block
- final_iteration: end-of-iteration status row
- top_rows: highest-|t| interaction rows
- notes: short interpretation notes

For plot(): An mfrm_plot_data object with view-specific tables, retained screening rules and interpretation notes. Plot agreement across views is not independent evidence of bias. Use print(p) or plot_data(p, "notes") for the interaction definition, score-direction caveat, magnitude-only facet-profile interpretation and conditional uncertainty qualifications outside the figure.

What this screening means

estimate_bias() summarizes additional cell departures from the fitted MFRM. It is best read as a targeted screening tool for potentially noteworthy cells or facet combinations that may merit substantive review.

What this screening does not justify

- t and Prob. are screening metrics, not formal inferential quantities.
- A flagged interaction cell is not, by itself, proof of rater bias or construct-irrelevant variance.
- Non-flagged cells should not be over-read as evidence that interaction effects are absent.

Interpreting output

Use summary for global magnitude, then inspect table for cell-level interaction effects.

Prioritize rows with:

- larger |Bias Size|, assessed against a substantively chosen threshold in native ability units and its effect on expected scores, rather than a universal small/large-effect rule

- larger |t| among the screening metrics (the default review rule is $|t| \geq 2$, not a validated fairness decision)
- smaller Prob. among the screening metrics

Use Bias Size for the direction of the departure from the baseline model. Expctd Score and Obs-Exp Average use the main-analysis expectations, before adding the cell shift. AfterBiasExpectedScore and AfterBiasObsExpAverage retain the remaining discrepancy after fitting it. Cell t reference df uses weighted count minus 1. This matches the executed FACETS 4.5.0 equal-count binary example; a manual count-minus-2 description is a distinct convention and does not establish version-wide equivalence. DFUnrounded retains count minus one, including nonpositive values. A nonpositive value withholds d.f., DFDisplay and Prob.; a finite point and conditional information SE can remain available. For positive df, DFDisplay rounds to the nearest integer (halves upward) for RSM/PCM display comparison only. d.f. and Prob. use the continuous reference. ProbabilityReference, ProbabilityStatus and ProbabilityReason identify the reference distribution and distinguish available conditional or extreme-adjusted tails from unavailable df, statistics or numerical results. These fields describe Prob., not the separate GPCM likelihood-ratio tail.

Checked FACETS 4.5.0 fractional-count tails are reproduced by extending an integer-df finite-sum formula to fractional df. That extension is not the continuous-df Student distribution and can change a 0.05 screening flag. mfrmr retains the continuous Student tail, independently checked by density integration. This explains the checked output behavior without claiming access to the executable's implementation or general finite-sample validity. The all-zero chi_sq screen sums squared bias/SE ratios without centering; its df is the number of cells. It requires at least two cells and complete cell estimates and SEs. Otherwise its statistic, degrees of freedom and tail area are missing, with available-cell counts and a reason.

summary() reports available and unavailable screening tail-area counts. With none available, screen-positive counts are NA, not zero. Bonferroni/Holm summaries use the full number of retained cells in this interaction, including unavailable cells; they do not include omitted or unobserved cells or other interactions from estimate_all_bias(). Adjustment does not validate the underlying plug-in tail areas or establish formal family-wise error control.

iteration records changes between repeated estimates of the same conditional cell problems. These passes do not refit person or facet parameters, and increasing max_iter does not turn this screen into a jointly fitted interaction model.

For summary():

- overview: interaction order, analyzed cells, and effect-size profile.
- chi_sq: fixed-effect test block.
- final_iteration: end-of-loop status from the bias routine.
- top_rows: strongest bias contrasts by |t|; GPCM summaries also retain the profile-likelihood review columns when present.

Start with "scatter" or "ranked" for triage, then confirm pattern shape using "abs_t_hist" and "facet_profile".

These views reuse the same estimates and flags. Agreement between views is not independent corroborating evidence of bias. Review exposure, calibration uncertainty and substantive explanations before drawing conclusions about a rater.

Typical workflow

1. Fit and diagnose model.
2. Run `estimate_bias(...)` for target interaction facets.
3. Review `summary(bias)` and `bias$table`.
4. Visualize/report via plot(x, type = "bias") and `build_fixed_reports()`.

For `summary()`:

1. Estimate interactions with `estimate_bias()`.
2. Check `summary(bias)` for screen-positive and unstable cells.
3. Use `bias_interaction_report()` or plot(x, type = "bias") for details.

Interpreting key output columns

In `bias$table`, the most-used columns are:

- `Bias Size`: estimated additional cell shift b_{jc} in native ability units; in `GPCM`, adjacent log odds change by ab_{jc}
- `t` and `Prob.`: screening metrics, not formal inferential quantities
- `S.E.`: conditional information-based SE of the cell shift, with the calibration held fixed
- `Obs-Exp Average`: observed-minus-baseline-expected average on the raw-score metric
- for `GPCM`, `LR ChiSq`, `LR Prob.`, and `Profile CI Lower` / `Profile CI Upper`: conditional profile-likelihood checks for a single additive bias shift, holding the fitted person, facet, step, and slope estimates fixed. Calibration uncertainty is not propagated. Check `Profile CI Status`: `limited by search range` marks a retained search limit, not a computed likelihood-ratio endpoint. These are screening approximations, not formal fairness tests.

The `chi_sq` element screens whether all interaction cells equal zero, using diagonal conditional SEs. It is not a full covariance test. Recreate older bias results with current diagnostics and `estimate_bias()`; saved old expected-score and chi-square definitions are not relabeled.

Recommended next step

Use plot(x, type = "bias") to inspect the flagged cells visually, then integrate the result with DFF, linking, or substantive scoring review before making formal claims about fairness or invariance. Save the `mfrm_bias` object for repeat use. `bias_interaction_report()` adds explicit screening rules without refitting; plotting the saved report keeps those rules. `mfrm_results()` has a separate bias-screen collection route; saving a standalone bias result does not automatically attach it there.

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press. (FACETS Table 13 corresponds to the bias / interaction estimation that this helper implements.)
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly, 2*(3), 197-221.
- Eckes, T. (2015). *Introduction to many-facet Rasch measurement: Analyzing and evaluating rater-mediated assessments* (2nd ed.). Peter Lang.

- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement, 4*(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement, 5*(2), 189-227.

Saved-result reuse

The returned `fit_identity` records the fitted analysis for reuse with reports and exports. Recreate older results from the saved fit and matching diagnostics using the original screening options; no model refit is needed. See mfrmr_compatibility_layer for migration instructions.

Controls

For `plot()`:

- `facet_a`, `facet_b`, or `interaction_facets` identify the interaction; `top_n = 40` and `sort_by = "abs_t"` (`"abs_bias"` or `"prob"` also accepted) select ranked interactions.
- `abs_t_warn = 2`, `abs_bias_warn = 0.5`, and `p_max = 0.05` set screening rules when preparing a report from the bias result. They are review aids, not guarantees of a calibrated error rate.
- `show_ci = FALSE` omits whiskers; `ci_level = 0.95` chooses their level for scatter/ranked views. GPCM rows retain conditional profile bounds, with dashed whiskers for search-limited or unverified bounds. Other rows use conditional per-cell SEs. Missing profile bounds are not replaced with another method. These intervals omit calibration uncertainty; endpoint completion is not evidence of coverage.
- `preset` selects `"standard"`, `"publication"`, `"compact"`, or `"monochrome"`; `palette` overrides named colors (`normal`, `flag`, `hist`, `profile`); `label_angle` controls label rotation. `title` sets title metadata (`main` is a compatibility alias; supply only one).

Plot types

`"scatter"` **(default)** Observed-minus-expected score average (x) versus conditional bias size in logits (y). Points use the saved flag status; horizontal guides show the saved absolute-bias threshold and zero, with a vertical guide at zero score difference.

`"ranked"` Conditional bias size (x) for up to `top_n` saved ranked interactions (y). Points are colored by flag status; optional whiskers retain the conditional uncertainty and incomplete-bound status. A saved report retains the ranking chosen when it was constructed.

`"heatmap"` Facet A by facet B matrix of signed bias size. Cells retain reusable matrix and flag tables for dashboards. This is a Table 13 follow-up display: it supports pattern recognition but does not turn screening rows into confirmatory tests.

`"abs_t_hist"` Histogram of absolute screening t-statistics across all interaction cells. Dashed reference line at `abs_t_warn`. Use for assessing the overall distribution of interaction effect sizes.

`"facet_profile"` Per-facet-level aggregation showing mean absolute bias and flag rate. Useful for identifying which individual facet levels drive systematic interaction patterns.

See Also

interaction_effect_table(), build_fixed_reports(), build_apa_outputs()

estimate_bias(), bias_interaction_report()

estimate_bias(), bias_interaction_report(), mfrmr_plot_migration, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("estimate_bias", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
s_bias <- summary(bias)
s_bias$overview
# This example uses the default RSM, so Bias Size is also in logit units.
# `Significant` counts cells crossing the screening rule, not proven bias.
# The `BonferroniSignificant` /
#   `HolmSignificant` columns count cells that survive multiple-
#   testing adjustment to screening tail areas. Zero flags do not
#   establish absence of bias or equivalence.
s_bias$top_rows
# An illustrative review rule is |t| > 2 and |Bias Size| > 0.5 logits.
# Choose the effect threshold substantively; these cutoffs do not
#   establish statistical reliability;
#   inspect cell counts, uncertainty and substantive consequences.
p_bias <- plot(bias, type = "bias", draw = FALSE)
p_bias$data$plot
# Optional finite reporting for extreme cells; this does not refit calibration
extreme_bias <- estimate_bias(fit, diag, "Rater", "Criterion", xtreme = .5)
head(extreme_bias$table[, c("Bias Size", "PrimaryBias", "BiasStatus", "SEBasis")])


## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_bias", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_bias")
toy <- toy[toy$Person %in% unique(toy$Person)[1:8], ]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 1)
summary(bias)

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_bias", run.dontrun = TRUE)
## Not run:
```

```
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diagnostics <- example_fit$diagnostics
bias <- estimate_bias(fit, diagnostics = diagnostics,
  facet_a = "Rater", facet_b = "Criterion")
p <- plot(bias, type = "ranked", preset = "publication", draw = FALSE)
head(plot_data(p, "ranked_table"))

## End(Not run)
```

`estimation_iteration_report` *Build an estimation-iteration report (preferred alias)*

Description

Build an estimation-iteration report (preferred alias)

Usage

```
estimation_iteration_report(
  fit,
  max_iter = 20,
  reltol = NULL,
  include_prox = TRUE,
  include_fixed = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`max_iter`	Maximum replay iterations (excluding optional initial row).
`reltol`	Stopping tolerance for replayed max-logit change.
`include_prox`	If `TRUE`, include an initial pseudo-row labeled `PROX`.
`include_fixed`	If `TRUE`, include a legacy-compatible fixed-width text block.

Details

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_iteration_report` (`type = "residual"`, `"logit_change"`, `"objective"`).

Value

A named list with iteration-report components and, for `GPCM`, a `gpcm_boundary` table. Class: `mfrm_iteration_report`.

Interpreting output

- `iterations`: trajectory of convergence indicators by iteration.
- `summary`: final status and stopping diagnostics.
- optional `PROX` row: pseudo-initial reference point when enabled.

For `GPCM`, this helper replays slope-aware optimization steps from a reconstructed starting state. It is not the exact optimizer history from the fitted object and is not an additional convergence test. Use `summary(fit, profile = "fit", detail = "brief")` for the recorded convergence result, and read the returned `gpcm_boundary` before reporting the replay.

Typical workflow

1. Run `estimation_iteration_report(fit)`.
2. Inspect plateau/stability patterns in summary/plot.
3. Adjust optimization settings if convergence looks weak.

See Also

`fit_mfrm()`, `specifications_report()`, `data_quality_report()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("estimation_iteration_report", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 7, maxit = 30
)
out <- estimation_iteration_report(fit, max_iter = 5)
summary(out)
p_iter <- plot(out, draw = FALSE)
p_iter$data$plot

## End(Not run)
```

---

evaluate_mfrm_design *Evaluate MFRM design conditions by repeated simulation*

Description

Evaluate MFRM design conditions by repeated simulation

Usage

```
evaluate_mfrm_design(
  n_person = c(30, 50, 100),
  n_rater = c(3, 5),
  n_criterion = c(3, 5),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  fit_method = c("JML", "MML"),
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  seed = NULL,
  progress = interactive(),
  parallel = c("no", "future"),
  file = NULL,
  file_refit = c("never", "on_change", "always")
)
```

Arguments

`n_person`	Vector of person counts to evaluate.
`n_rater`	Vector of rater counts to evaluate.
`n_criterion`	Vector of criterion counts to evaluate.
`raters_per_person`	Vector of rater assignments per person.
`design`	Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (`n_person`, `n_rater`, `n_criterion`, `raters_per_person`), current public aliases implied by `sim_spec` (for example `n_judge`, `n_task`, `judge_per_person`), or role keywords (`person`, `rater`, `criterion`, `assignment`). Values may be vectors. The nested named-facet form `design$facets = c(person = ..., judge = ..., task = ...)` is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both `design` and the scalar design-grid arguments.

reps	Number of replications per design condition.
score_levels	Number of ordered score categories.
theta_sd	Standard deviation of simulated person measures.
rater_sd	Standard deviation of simulated rater severities.
criterion_sd	Standard deviation of simulated criterion difficulties.
noise_sd	Optional observation-level noise added to the linear predictor.
step_span	Spread of step thresholds on the logit scale.
fit_method	Estimation method passed to fit_mfrm().
model	Measurement model passed to fit_mfrm(). RSM and PCM use the documented Rasch-family design-planning layer. GPCM is available as a caveated simulation/refit operating-characteristic route.
step_facet	Step facet passed to fit_mfrm() when model = "PCM" or model = "GPCM". When left NULL, the function inherits the generator step facet from sim_spec when available and otherwise defaults to "Criterion".
slope_facet	Slope facet passed to fit_mfrm() when model = "GPCM". The current bounded branch requires slope_facet == step_facet.
slopes	Optional GPCM generator slopes used when sim_spec = NULL. See build_mfrm_sim_spec() for accepted formats.
assignment	Optional assignment design used when sim_spec = NULL. "sparse_linked" activates planned-missing sparse rating designs; use sparse_controls to specify the linking set.
sparse_controls	Optional named list used when assignment = "sparse_linked" and sim_spec = NULL, or retained from a sparse linked sim_spec. See build_mfrm_sim_spec().
maxit	Maximum iterations passed to fit_mfrm().
quad_points	Quadrature points for fit_method = "MML".
residual_pca	Residual PCA mode passed to diagnose_mfrm().
sim_spec	Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec() used as the base data-generating mechanism. When supplied, the design grid still varies n_person, n_rater, n_criterion, and raters_per_person, but latent-spread assumptions, thresholds, and other generator settings come from sim_spec. If sim_spec contains step-facet-specific thresholds, the design grid may not vary the number of levels for that step facet away from the specification. If sim_spec stores an active latent-regression population generator, this helper currently requires fit_method = "MML" so each replication can refit the population model.
seed	Optional seed for reproducible replications.
progress	Logical. Whether to show a progress bar across design-by-replication cells. Defaults to interactive(), so interactive exploratory runs show progress while non-interactive tests, scripts, and report rendering stay quiet. Set TRUE or FALSE explicitly to override.

`parallel` Parallelisation strategy for the rep loop within each design row. `"no"` (default) runs serially; `"future"` uses `future.apply::future_lapply` and respects whatever `future::plan()` is currently active. The Suggests package `future.apply` must be installed for the parallel path to activate; otherwise the call falls back to serial execution with a single message. A fixed explicit `seed` preallocates the same per-cell simulation seeds before either route, so serial and future execution receive identical stochastic inputs for the same ordered design grid. Parallel dispatch applies to replications within each design row; actual worker concurrency depends on the active future plan.

`file` Optional RDS path for saving and reusing the simulation. A missing `.rds` extension is appended. Its parent directory must already exist. `NULL` (default) performs no file reads or writes.

`file_refit` File reuse policy. `"never"` (default) loads a matching completed result or resumes matching saved trials; incompatible files produce an error. `"on_change"` restarts when inputs or implementation/ environment differ. `"always"` replaces the file with a new run. Unlike `brms::brm(file_refit = "never")`, this function checks compatibility even under `"never"`; it does not silently return a different analysis.

Details

This helper runs a compact Monte Carlo design study for common rater-by-item many-facet settings.

For each design condition, the function:

1. generates synthetic data with `simulate_mfrm_data()`
2. fits the requested MFRM with `fit_mfrm()`
3. computes diagnostics with `diagnose_mfrm()`
4. stores recovery and precision summaries by facet

Severity recovery compares relative locations after aligning group means within each trial. PCM/GPCM generating step means are absorbed into the owning facet truth before comparison, consistently with the fitted identification. `MeanSeverityRMSE` is the mean of available trial RMSEs, with MCSE across trials; it differs from a pooled-row RMSE. `SeverityBias` averages to zero by construction and cannot demonstrate unbiased individual estimates. `RecoveryAvailableReps` counts finite trial recovery metrics; review it alongside all-trial convergence and failures.

Older PCM/GPCM results without this truth-coordinate review retain their historical tables. New summaries and recommendations withhold their recovery metrics with `RecoveryBasis = "legacy_step_location_truth_`
Re-summarizing cannot reconstruct the missing trial-level truth/estimates; rerun in a new file for corrected recovery, preserving the old evidence. Other saved diagnostics remain available. Numerical agreement and successful examples do not establish finite-sample performance or population forecasting.

The result is intended for planning questions such as:

- how many raters are needed for stable rater separation?
- how does `raters_per_person` affect severity recovery?
- when do category counts become too sparse for comfortable interpretation?

This is a **parametric simulation study**. It does not take one observed design (for example, 4 raters x 30 persons x 3 criteria) and analytically extrapolate what would happen under a different design (for example, 2 raters x 40 persons x 5 criteria). Instead, you specify a design grid and data-generating assumptions (latent spread, facet spread, thresholds, noise, and scoring structure), and the function repeatedly generates synthetic data under those assumptions.

When you want the simulated conditions to resemble an existing study, use substantive knowledge or estimates from that study to choose `theta_sd`, `rater_sd`, `criterion_sd`, `score_levels`, and related settings before running the design evaluation.

When `sim_spec` is supplied, the function uses it as the explicit data-generating mechanism. This is the recommended route when you want a design study to stay close to a previously fitted run while still varying the candidate sample sizes or rater-assignment counts. The specification supplies the generator, not the analysis settings: explicitly match `fit_method`, `model`, `step_facet`, `maxit`, and `quad_points` to the intended analysis. The defaults (`JML`, 25 iterations; seven quadrature points if `MML` is selected) are small exploratory settings, not an accuracy recommendation. This helper does not automatically inherit anchors, interactions, adaptive integration, or optimizer controls from a source fit. If those are essential to the analysis, generate data with `simulate_mfrm_data()` and use the intended `fit_mfrm()` call in each replicate.

Sparse linked simulation specifications and direct `assignment = "sparse_linked"` calls are carried into the design-evaluation output. The resulting sparse-design columns report planned-missingness and rater-link diagnostics (for example design density and rater-pair common persons). They are design diagnostics, not fit statistics or universal adequacy thresholds.

If that specification also stores a latent-regression population generator, each replication carries forward the simulated one-row-per-person background data and refits the MML population-model branch. This remains a scenario study under explicit assumptions; it is not a closed-form predictive distribution for one future administration.

`GPCM` design evaluation is available with caveats. It repeatedly generates data from the supplied or fit-derived slope-aware specification, refits `GPCM`, and summarizes facet-level operating characteristics. The route remains a role-based person x rater-like x criterion-like planner: it does not validate diagnostic-screening or signal-detection rules, does not provide a fully arbitrary-facet planner, and does not replace `simulate_mfrm_recovery()` for slope-recovery adequacy review.

Recovery metrics are reported only when the generator and fitted model target the same facet-parameter contract. In practice this means the same `model`, and for `PCM`, the same `step_facet`. When these do not align, recovery fields are set to `NA` and the output records the reason. Even when these contract checks pass, the recovery summaries still assume compatible orientation and anchoring conventions across the generator and fitted model.

Value

An object of class `mfrm_design_evaluation` with components:

- `design_grid`: evaluated design conditions. When `sim_spec` carries custom public facet names, matching design-variable alias columns are included alongside the standard result columns.
- `results`: facet-level replicate results, with the same design-variable alias columns when applicable.
- `rep_overview`: run-level status and timing, with the same design-variable alias columns when applicable. `Observations` counts generated rating rows; `MaxRatingsPerRater` is the largest

row count for a single rater-like facet level. One row is one person-rater-criterion rating, regardless of `Weight`. These workload counts are retained even when fitting or diagnostics fail. `RaterComponents` and `CriterionComponents` count connected components in each generated Person-facet assignment graph, using the same audit as `describe_mfrm_data()`. They refer to the rater-like and criterion-like roles even when custom facet names are used, and also survive failures. Failed fits retain the condition class, failure component, and category-support state/reason codes when available; a completely failed design still returns the documented zero-row `results` schema rather than a zero-column table.

- `design_descriptor`: role-based design-variable metadata used by planning summaries and plots
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of which design variables remain mutable under the current simulation specification
- `planning_schema`: structured planning metadata bundling the role descriptor, scope boundary, and current mutability map
- `gpcm_boundary`: `GPCM` caveat row when a `GPCM` design route is used
- `notes`: short interpretation notes
- `settings`: simulation settings
- `ademp`: simulation-study metadata (aims, DGM, estimands, methods, performance measures)
- `reproducibility`: file identity, original inputs and per-trial seeds when `file` is supplied.

Reported metrics

Facet-level simulation results include:

- `Separation` ($G = \mathrm{SD}_{\mathrm{adj}}/\mathrm{RMSE}$): how many statistically distinct strata the facet resolves.
- `Reliability` ($G^2/(1+G^2)$): model-based separation reliability for the facet levels in that fit, not Cronbach's alpha, a G-theory coefficient, or the accuracy of an individual pass/fail decision.
- `Strata` ($(4G+1)/3$): number of distinguishable groups.
- Mean `Infit` and `Outfit`: average fit mean-squares across elements.
- `MisfitRate`: share of elements with $|\mathrm{ZSTD}| > 2$.
- Sparse-design diagnostics when `assignment = "sparse_linked"`: `MeanDesignDensity`, `MeanPlannedMissingRate`, `MeanLinkPersons`, `MeanMinCommonPersonsPerRaterPair`, `MaxZeroCommonRaterPairs`, and `MaxRaterPairsBelowTarget`.
- `SeverityRMSE`: root-mean-square error of recovered parameters vs the known truth **after facet-wise mean alignment**, so that the usual Rasch/MFRM location indeterminacy does not inflate recovery error. This quantity is reported only when the generator and fitted model target the same facet-parameter contract.
- `SeverityBias`: mean signed recovery error after the same alignment; values near zero are expected. This is likewise omitted when the generator/fitted-model contract does not align.

Interpreting output

Start with `summary(x)$design_summary`, then plot one focal metric at a time (for example rater `Separation` or criterion `SeverityRMSE`).

Person separation/reliability describes how well the design distinguishes persons under the model. High rater separation/reliability instead means that severity differences are distinguishable; it does not mean agreement or better calibration. Low rater separation can reflect similar severity or inadequate precision. Interpret each facet against its purpose rather than maximizing every separation coefficient. Recovery RMSE and elapsed time concern accuracy and cost; a low misfit-screen rate alone does not establish useful detection, since sensitivity and false flags differ.

When choosing among designs, look for the point where increasing `n_person` or `raters_per_person` yields diminishing returns in separation and RMSE, then compare the actual rating workload. Inspect failed-run reasons when convergence is low: iteration limits, weak identification, sparse categories and model mismatch require different remedies. Increasing sample size alone does not resolve every failure. Read Monte Carlo errors and available-result counts alongside the means; passing thresholds on simulation means is not a guarantee for a future sample.

This is a Monte Carlo design-evaluation helper. It can visualize how separation, reliability, strata, RMSE, and fit-screen rates change when you vary person, rater, criterion, or assignment counts. For analytic generalizability-theory planning, pair observed variance-component review from `mfrm_generalizability()` with D-study projections from `mfrm_d_study()`. The complementary multivariate observed-score route is `mfrm_g_study()` -> `mfrm_d_study()`. See mfrmr_workflow_methods for choosing between simulation and G-theory: neither is automatically required by the other, and neither replaces the other's estimand. Read `IdentificationStatus`, `GStatus`, and `PhiStatus` before using the projected coefficients: boundary or singular mixed-model fits are design-identification warnings rather than high-stakes-ready reliability evidence.

References

The simulation logic follows the general Monte Carlo / operating-characteristic framework described by Morris, White, and Crowther (2019) and the ADEMP-oriented planning/reporting guidance summarized for psychology by Siepe et al. (2024). In `mfrmr`, `evaluate_mfrm_design()` is a practical many-facet design-planning wrapper rather than a direct reproduction of one published simulation study.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

Saving and resuming simulations

Pass the same `file` and analysis arguments on a subsequent call to reuse the completed result without generation or fitting, or to resume an interrupted run. The identity includes generator inputs/specification, fitting controls, requested replications, seed, RNG kind, package R code and native library fingerprints, relevant numerical/diagnostic R options, R/platform, numerical libraries and dependency versions. Changing `reps` starts a different study; extension of an existing study is not supported. `progress` and `parallel` do not change the identity. Cached results retain the settings of their original run.

Recovery runs and serial design evaluations save after every finished trial. Future design evaluations save after a whole design row returns; an interrupted row may need to run again. A partially completed optimizer is not resumed. Fit failures are completed attempts and are saved too; resuming does not redraw failures until they succeed. An unexpected error outside the recorded fit/diagnostic failure routes stops the call and leaves earlier checkpoints available. These files do not change the interpretation or eligibility of the underlying estimates and intervals.

Per-trial seeds are allocated before computation and retained even when `seed = NULL`; an explicit seed is preferable for reproducing a new run in another file. Each trial controls generation, fitting and diagnostics with its own seed. A matching environment is required for automatic reuse; recorded seeds alone do not guarantee identical floating-point results on other systems. Keep the analysis script and software environment as well.

After completion, `readRDS(file)` returns the usual result object with an additional `reproducibility` component containing identity, inputs and trial seeds. Before completion the file is an internal checkpoint; resume through the same public function. Writes use a temporary file in the same directory and checked replacement. A `.lock` directory prevents concurrent writers. Normal returns/errors release it; after process termination, remove a remaining lock only after checking that no writer is active. Unreadable files are never silently reused or replaced by `"on_change"`.

See Also

`simulate_mfrm_data()`, summary.mfrm_design_evaluation, plot.mfrm_design_evaluation

Examples

```
# Long-running workflow; run explicitly when needed.
# example("evaluate_mfrm_design", run.dontrun = TRUE)
## Not run:
sim_eval <- suppressWarnings(evaluate_mfrm_design(
  design = list(person = c(8, 12), rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  seed = 123
))
s_eval <- summary(sim_eval)
s_eval$design_summary[, c("Facet", "n_person", "MeanSeparation", "MeanSeverityRMSE")]
p_eval <- plot(sim_eval, facet = "Rater", metric = "separation", x_var = "n_person", draw = FALSE)
names(p_eval)

## End(Not run)
```

`evaluate_mfrm_diagnostic_screening`

Evaluate legacy and strict marginal diagnostic screening under controlled misfit scenarios

---

Description

Evaluate legacy and strict marginal diagnostic screening under controlled misfit scenarios

Usage

```
evaluate_mfrm_diagnostic_screening(
  n_person = c(30, 50, 100),
  n_rater = c(4),
  n_criterion = c(4),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  scenarios = c("well_specified", "local_dependence"),
  local_dependence_sd = 0.8,
  local_dependence_facet = NULL,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  include_report = FALSE,
  report_include = c("fit", "diagnostics", "tables", "precision", "reporting"),
  report_style = c("qc", "apa", "validation", "reviewer", "technical"),
  seed = NULL
)
```

Arguments

n_person	Vector of person counts to evaluate.
n_rater	Vector of rater counts to evaluate.
n_criterion	Vector of criterion counts to evaluate.
raters_per_person	Vector of rater assignments per person.
design	Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases implied by sim_spec, or role keywords (person, rater, criterion, assignment). Values may be vectors.
reps	Number of replications per design condition and scenario.
scenarios	Screening scenarios to evaluate. Supported values are "well_specified", "local_dependence", and "latent_misspecification", plus "step_structure_misspecification".

`local_dependence_sd`
: Standard deviation of the shared context effect injected in the `"local_dependence"` scenario.

`local_dependence_facet`
: Facet that receives the shared `Person x facet` dependence effect. Use `"criterion"`, `"rater"`, or an active public facet name. Defaults to the criterion-like facet.

`score_levels`
: Number of ordered score categories.

`theta_sd`
: Standard deviation of simulated person measures.

`rater_sd`
: Standard deviation of simulated rater severities.

`criterion_sd`
: Standard deviation of simulated criterion difficulties.

`noise_sd`
: Optional observation-level noise added to the linear predictor.

`step_span`
: Spread of step thresholds on the logit scale.

`model`
: Measurement model passed to `fit_mfrm()`. `GPCM` is supported with caveats as slope-aware screening sensitivity evidence.

`step_facet`
: Step facet passed to `fit_mfrm()` when `model = "PCM"` or `model = "GPCM"`.

`slope_facet`
: Slope facet passed to `fit_mfrm()` when `model = "GPCM"`. Defaults to the fitted step facet.

`slopes`
: Optional `GPCM` slope specification used by direct simulation calls when `sim_spec = NULL`.

`maxit`
: Maximum iterations passed to `fit_mfrm()`.

`quad_points`
: Quadrature points for the underlying `MML` fit.

`residual_pca`
: Residual PCA mode passed to `diagnose_mfrm()`.

`sim_spec`
: Optional output from `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()` used as the base data-generating mechanism.

`include_report`
: Logical; if `TRUE`, each successful replicate also builds `mfrm_results()` and `mfrm_report()` and records the `report_index` readiness/signaling surface. This is intentionally opt-in because it repeats the comprehensive result-building workflow.

`report_include`
: `include` vector passed to `mfrm_results()` when `include_report = TRUE`.

`report_style`
: Report style passed to `mfrm_report()` when `include_report = TRUE`.

`seed`
: Optional seed for reproducible replications.

Details

This helper performs a compact Monte Carlo evaluation of the package's diagnostic architecture under user-specified simulation conditions.

For each design condition and scenario, the function:

1. generates synthetic data with `simulate_mfrm_data()`
2. fits the model with `method = "MML"`
3. computes diagnostics with `diagnostic_mode = "both"`
4. stores legacy residual-screen metrics and strict marginal-fit metrics

5. optionally stores mfrm_report() report_index readiness signals
6. aggregates the results into scenario_summary, performance_summary, report_signal_summary, and scenario_contrast

The "well_specified" scenario uses the ordinary generator with no injected extra structure. The "local_dependence" scenario adds a shared Person x facet random effect, centered within the selected facet levels, so responses in the same context become correlated without changing the facet-level mean effect contract. The "latent_misspecification" scenario keeps the same marginal spread targets but replaces the normal person distribution with a centered bimodal empirical support distribution, while leaving the non-person facets on the original scale contract. The "step_structure_misspecification" scenario uses a PCM or GPCM generator with facet-specific threshold tables that intentionally mismatch the fitted step contract: RSM fits receive criterion-specific thresholds, and PCM / GPCM fits receive threshold structures indexed by the opposite non-person facet. For GPCM, the generator and fit each keep slope_facet == step_facet; the misspecification is the generator-versus-fit step/slope facet mismatch.

This function is intentionally screening-oriented. The strict marginal branch remains exploratory, so the returned summaries should be used to compare relative sensitivity across scenarios rather than to claim calibrated inferential power. GPCM rows add explicit gpcm_boundary caveats and should be read as slope-aware operating characteristics under the evaluated role-based design.

Value

An object of class mfrm_diagnostic_screening with:

- design_grid: evaluated design conditions, including public alias columns when applicable
- results: replicate-level screening metrics for each design and scenario
- scenario_summary: aggregated scenario-by-design screening summaries
- performance_summary: scenario-by-design screening-performance summary including runtime, agreement, Type I proxy, and sensitivity proxy columns
- report_signal_summary: optional scenario-by-design summary of mfrm_report() report_index availability, readiness, and review-signal counts when include_report = TRUE
- scenario_contrast: each misspecification scenario minus the well-specified baseline when the baseline scenario was evaluated
- design_descriptor: role-based design-variable metadata
- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: structured planning metadata
- gpcm_boundary: GPCM caveat row when present
- settings: simulation and fitting settings
- ademp: simulation-study metadata
- notes: short interpretation notes

See Also

simulate_mfrm_data(), evaluate_mfrm_design(), diagnose_mfrm()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("evaluate_mfrm_diagnostic_screening", run.dontrun = TRUE)
## Not run:
diag_eval <- evaluate_mfrm_diagnostic_screening(
  design = list(person = 10, rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  seed = 123
)
diag_eval$scenario_summary
diag_eval$scenario_contrast

## End(Not run)
```

---

evaluate_mfrm_signal_detection

Evaluate DIF power and bias-screening behavior under known simulated signals

---

Description

Evaluate DIF power and bias-screening behavior under known simulated signals

Usage

```
evaluate_mfrm_signal_detection(
  n_person = c(30, 50, 100),
  n_rater = c(4),
  n_criterion = c(4),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  group_levels = c("A", "B"),
  reference_group = NULL,
  focal_group = NULL,
  dif_level = NULL,
  dif_effect = 0.6,
  bias_rater = NULL,
  bias_criterion = NULL,
  bias_effect = -0.8,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
```

```
  fit_method = c("JML", "MML"),
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  dif_method = c("residual", "refit"),
  dif_min_obs = 10,
  dif_p_adjust = "holm",
  dif_p_cut = 0.05,
  dif_abs_cut = 0.43,
  bias_max_iter = 2,
  bias_p_cut = 0.05,
  bias_abs_t = 2,
  seed = NULL
)

## S3 method for class 'mfrm_signal_detection'
plot(
  x,
  signal = c("dif", "bias"),
  metric = c("power", "false_positive", "estimate", "screen_rate",
    "screen_false_positive"),
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

n_person	Vector of person counts to evaluate.
n_rater	Vector of rater counts to evaluate.
n_criterion	Vector of criterion counts to evaluate.
raters_per_person	Vector of rater assignments per person.
design	Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases implied by sim_spec (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). Values may be vectors. The nested named-facet form design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both design and the scalar design-grid arguments.
reps	Number of replications per design condition.

`group_levels` Group labels used for DIF simulation. The first two levels define the default reference and focal groups. If omitted, inherits the groups stored in `sim_spec`, when present.

`reference_group`
Optional reference group label used when extracting the target DIF contrast.

`focal_group` Optional focal group label used when extracting the target DIF contrast.

`dif_level` Target criterion level for the true DIF effect. Can be an integer index or a criterion label such as `"C04"`. Defaults to the last criterion level in each design.

`dif_effect` True DIF effect size added to the focal group on the target criterion's linear predictor. For GPCM, discrimination multiplies this shift in the adjacent-category log odds.

`bias_rater` Target rater level for the true interaction-bias effect. Can be an integer index or a label such as `"R04"`. Defaults to the last rater level in each design.

`bias_criterion` Target criterion level for the true interaction-bias effect. Can be an integer index or a criterion label. Defaults to the last criterion level in each design.

`bias_effect` True interaction-bias effect added to the target `Rater x Criterion` cell's linear predictor, with the same GPCM scaling.

`score_levels` Number of ordered score categories.

`theta_sd` Standard deviation of simulated person measures.

`rater_sd` Standard deviation of simulated rater severities.

`criterion_sd` Standard deviation of simulated criterion difficulties.

`noise_sd` Optional observation-level noise added to the linear predictor.

`step_span` Spread of step thresholds on the logit scale.

`fit_method` Estimation method passed to `fit_mfrm()`.

`model` Measurement model passed to `fit_mfrm()`. `GPCM` is supported with caveats as slope-aware signal-detection sensitivity evidence.

`step_facet` Step facet passed to `fit_mfrm()` when `model = "PCM"` or `model = "GPCM"`. When left `NULL`, the function inherits the generator step facet from `sim_spec` when available and otherwise defaults to `"Criterion"`.

`slope_facet` Slope facet passed to `fit_mfrm()` when `model = "GPCM"`. Defaults to the fitted step facet.

`slopes` Optional `GPCM` slope specification used by direct simulation calls when `sim_spec = NULL`.

`maxit` Maximum iterations passed to `fit_mfrm()`.

`quad_points` Quadrature points for `fit_method = "MML"`.

`residual_pca` Residual PCA mode passed to `diagnose_mfrm()`.

`sim_spec` Optional output from `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()` used as the base data-generating mechanism. When supplied, the design grid still varies `n_person`, `n_rater`, `n_criterion`, and `raters_per_person`, but latent spread, thresholds, and other generator settings come from `sim_spec`. The target DIF and interaction-bias signals specified in this function override any signal tables stored in `sim_spec`. If `sim_spec` stores an active latent-regression population generator, this helper currently requires `fit_method = "MML"` so each replication can refit the population model.

dif_method	Differential-functioning method passed to analyze_dff().
dif_min_obs	Minimum observations per group cell for analyze_dff().
dif_p_adjust	P-value adjustment method passed to analyze_dff(). Target and non-target screens use the returned adjusted tail areas; an unavailable adjusted value never falls back to an unadjusted one.
dif_p_cut	P-value cutoff for counting a target DIF detection.
dif_abs_cut	Optional absolute contrast cutoff used when counting a target DIF detection. When omitted, the effective default is 0.43 for dif_method = "refit" and 0 (no additional magnitude cutoff) for dif_method = "residual".
bias_max_iter	Maximum iterations passed to estimate_bias().
bias_p_cut	P-value cutoff for counting a target bias screen-positive result.
bias_abs_t	Absolute t cutoff for counting a target bias screen-positive result.
seed	Optional seed for reproducible replications. Trial seeds are allocated before execution and cover generation, fitting and diagnostics. A supplied seed preserves the caller's RNG state.
x	Output from evaluate_mfrm_signal_detection().
signal	Whether to plot DIF or bias screening results.
metric	Metric to plot. For signal = "bias", prefer metric = "screen_rate" for the screening hit rate. The older metric = "power" spelling is retained as a backwards-compatible alias that maps to BiasScreenRate. "estimate" plots the signed reference-minus-focal DIF contrast or the additive bias shift. A refit contrast requires common-scale linking.
x_var	Design variable used on the x-axis. When x was generated from a sim_spec with custom public facet names, the corresponding aliases (for example n_judge, n_task, judge_per_person) are also accepted. Role keywords (person, rater, criterion, assignment) are accepted as an abstraction over the current two-facet schema.
group_var	Optional design variable used for separate lines. The same alias rules as x_var apply.
draw	If TRUE, draw with base graphics; otherwise return plotting data.
...	Reserved for generic compatibility.

Details

This function performs Monte Carlo design screening for two related tasks: DIF detection via analyze_dff() and interaction-bias screening via estimate_bias().

For each design condition (combination of n_person, n_rater, n_criterion, raters_per_person), the function:

1. Generates synthetic data with simulate_mfrm_data()
2. Injects one known Group × Criterion DIF effect (dif_effect added to the focal group's linear predictor on the target criterion)
3. Injects one known Rater × Criterion interaction-bias effect (bias_effect added to the linear predictor)

4. Fits and diagnoses the MFRM
5. Runs analyze_dff() and estimate_bias()
6. Records whether the injected signals were detected or screen-positive

GPCM runs preserve the current package constraint slope_facet == step_facet within the generator and fitted model. The resulting DIF and bias rates are slope-aware screening summaries, not formal inferential power, alpha calibration, operational scoring, or arbitrary-facet planning evidence.

Detection criteria: A DIF signal is counted as "detected" when the target contrast has $p \leq$ dif_p_cut **and**, when an absolute contrast cutoff is in force, $|\text{Contrast}| \geq$ dif_abs_cut. For dif_method = "refit", dif_abs_cut is interpreted on the logit scale. For dif_method = "residual", DIF detection, classification and false-positive rates are unavailable (NA). Residual contrasts remain available as descriptive summaries; a missing test is not counted as a non-detection.

Bias results are different: estimate_bias() reports t and Prob. as screening metrics rather than formal inferential quantities. Here, a bias cell is counted as **screen-positive** only when those screening metrics are available and satisfy

$p \leq$ bias_p_cut **and** $|t| \geq$ bias_abs_t. If either required metric is unavailable, the result is NA, not FALSE. A descriptive-only DIF classification likewise supplies no classification detection outcome, even when the contrast itself is available.

Power is the proportion of replications in which the target signal was flagged when a comparison statistic is available. Refit DIF rates are conditional screening rates, not calibrated inferential power. For bias, the primary summary is BiasScreenRate, a screening hit rate rather than formal inferential power. With a zero generating effect, the target rate describes null flags rather than sensitivity. The historical column name DIFPower is retained for compatibility.

False-positive rate first counts flags among available non-target cells within each replication, then averages those proportions across replications. DIFNonTargetPlanned/DIFNonTargetAvailable and the corresponding BiasNonTarget* columns retain cell denominators. These rates are neither calibrated inferential alpha levels nor probabilities of any false flag in a family. For per-target and any-target rates with a complete planned roster, use mfrm_screening_performance().

Target summaries retain planned, available, unavailable and positive counts, exact 95% binomial Monte Carlo bounds, and bounds obtained by assigning every unresolved outcome negative or positive. Rates and Monte Carlo intervals condition on available outcomes; unresolved-outcome bounds are not confidence intervals. These uncertainty calculations describe independent simulation replications, not uncertainty about the quality of a real rater.

Direction and scale: DIFContrast is always reference minus focal, regardless of the order returned by analyze_dff(). For "residual", this is a difference in mean observed-minus-expected scores. For "refit", it is a difference in facet locations; means and absolute means require documented common-scale comparability. Raw differences with unavailable linking remain in results with DIFContrastComparable = FALSE and DIFScaleLinkStatus. Neither contrast is automatically an estimate of the injected dif_effect: the residual contrast uses score units, and refit locations use difficulty/ severity orientation and estimated linking anchors. BiasSize is a fitted additive shift relative to the estimated baseline, not recovery against a known unchanged baseline. See analyze_dff() and estimate_bias() for the conditioning assumptions.

Failures and replay: failures of generation, fitting or diagnostics keep their planned trial rows, with Seed, ErrorStage and Error in rep_overview. Missing outcomes do not count as negative screens. Generation failure gives unavailable observation/category counts, not zero observations.

ElapsedSec includes generation and all analysis stages. Each later trial keeps its own seed even if an earlier trial fails. Save the returned object with saveRDS() and preserve the call and software environment; this helper does not provide the file/checkpoint arguments of evaluate_mfrm_design(). Seeds alone do not guarantee identical results across software/platforms.

Rebuild older saved summaries and plots from the original evaluation to correct unavailable target outcomes and retain full-precision summaries. Older runs lack non-target availability counts, so their non-target rates are withheld: recovering those cell rates requires rerunning the simulation. Runs lacking contrast-direction metadata cannot supply signed means; old refit runs lacking comparability metadata cannot supply absolute means either. Original stored results and otherwise available screening outcomes remain. This function evaluates Group-by-Criterion DIF and Rater-by-Criterion bias; it does not evaluate general Rater-by-Group differential rater functioning.

Default effect sizes: dif_effect = 0.6 and bias_effect = -0.8 are example shifts of the latent linear predictor, not validated small/moderate/ large effect thresholds. Choose values of practical concern for the intended assessment. GPCM slopes multiply these shifts on the adjacent-logit scale; response-probability changes also depend on ability and category thresholds.

This is again a **parametric simulation study**. The function does not estimate a new design directly from one observed dataset. Instead, it evaluates detection or screening behavior under user-specified design conditions and known injected signals.

If you want to approximate a real study, choose the design grid and simulation settings so that they reflect the empirical context of interest. For example, you may set n_person, n_rater, n_criterion, raters_per_person, and the latent-spread arguments to values motivated by an existing assessment program, then study how operating characteristics change as those design settings vary.

When sim_spec is supplied, the function uses it as the explicit data-generating mechanism for the latent spreads, thresholds, and assignment archetype, while still injecting the requested target DIF and bias effects for each design condition.

If that specification also stores a latent-regression population generator, each replication carries simulated one-row-per-person background data into the MML fit. This remains a screening-oriented Monte Carlo study; it is not a person-level posterior prediction for one observed sample.

plot() method: Plot DIF/bias screening simulation results

Value

An object of class mfrm_signal_detection with:

- design_grid: evaluated design conditions. When sim_spec carries custom public facet names, matching design-variable alias columns are included alongside the standard result columns.
- results: replicate-level detection results, with the same design-variable alias columns when applicable. Seed identifies the trial; DIFReferenceGroup, DIFFocalGroup, DIFContrastDirection and DIFContrastScale identify its contrast. DIFScreenReason and BiasScreenReason retain target-level unavailability explanations.
- rep_overview: run-level status and timing, with the same design-variable alias columns when applicable, including failed attempts and their stage.
- design_descriptor: role-based design-variable metadata used by planning summaries and plots

- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of which design variables remain mutable under the current simulation specification
- planning_schema: structured planning metadata bundling the role descriptor, scope boundary, and current mutability map
- gpcm_boundary: GPCM caveat row when a GPCM screening route is used
- settings: signal-analysis settings
- ademp: simulation-study metadata (aims, DGM, estimands, methods, performance measures)
- notes: short interpretation notes

For plot(): If draw = TRUE, invisibly returns plotting data. If draw = FALSE, returns that plotting-data list directly. The returned list includes resolved canonical variables (x_var, group_var) together with public labels (x_label, group_label), design_variable_aliases, design_descriptor, planning_scope, planning_constraints, planning_schema, display_metric, and interpretation_note so callers can label bias-side plots as screening summaries rather than formal power/error-rate displays. summary_table retains per-design means and availability denominators; notes retains interpretation and legacy-result restrictions. If additional design variables are collapsed, each plotted point is an equally weighted average of available design-level means, not a pooled trial/cell rate.

References

The simulation logic follows the general Monte Carlo / operating-characteristic framework described by Morris, White, and Crowther (2019) and the ADEMP-oriented planning/reporting guidance summarized for psychology by Siepe et al. (2024). In mfrmr, evaluate_mfrm_signal_detection() is a many-facet screening helper specialized to DIF and interaction-bias use cases; it is not a direct implementation of one published many-facet Rasch simulation design.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

See Also

simulate_mfrm_data(), evaluate_mfrm_design(), analyze_dff(), analyze_dif(), estimate_bias()

evaluate_mfrm_signal_detection(), summary.mfrm_signal_detection

Examples

```
# Long-running workflow; run explicitly when needed.
# example("evaluate_mfrm_signal_detection", run.dontrun = TRUE)
## Not run:
sig_eval <- suppressWarnings(evaluate_mfrm_signal_detection(
  design = list(person = 8, rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  bias_max_iter = 1,
```

```
  seed = 123
))
s_sig <- summary(sig_eval)
s_sig$overview

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_signal_detection", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-signal-detection.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
sig_eval <- saved$result
plot(sig_eval, signal = "dif", metric = "power", x_var = "n_person", draw = FALSE)

## End(Not run)
```

`export_mfrm_bundle` *Export a fit-level analysis archive*

Description

Export a fit-level analysis archive

Usage

```
export_mfrm_bundle(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  summary_tables = NULL,
  output_dir = ".",
  prefix = "mfrmr_bundle",
 include = c("core_tables", "checklist", "dashboard", "apa", "anchors", "manifest",
    "visual_summaries", "predictions", "summary_tables", "script", "html"),
  facet = NULL,
  include_person_anchors = FALSE,
  overwrite = FALSE,
  acknowledge_sensitive = FALSE,
  zip_bundle = FALSE,
  zip_name = NULL,
  data = NULL
)
```

Arguments

fit	Output from fit_mfrm() or run_mfrm_facets().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are reused from run_mfrm_facets() when available, otherwise computed with residual_pca = "none" (or "both" when visual summaries are requested). Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
bias_results	Optional output from estimate_bias() or a named list of bias bundles.
population_prediction	Optional output from predict_mfrm_population().
unit_prediction	Optional output from predict_mfrm_units().
plausible_values	Optional output from sample_mfrm_plausible_values().
summary_tables	Optional manuscript-summary bundle input. Can be build_summary_table_bundle() output, any object supported by build_summary_table_bundle(), or a named list of such objects. When NULL and "summary_tables" is requested in include, a default set is built from fit, diagnostics, reporting_checklist(), and build_apa_outputs(). Compatible precomputed recovery-evidence summaries can be supplied here to co-locate their appendix tables with a fit-based export bundle.
output_dir	Directory where files will be written.
prefix	File-name prefix.
include	Components to export. Supported values are "core_tables", "checklist", "dashboard", "apa", "anchors", "manifest", "visual_summaries", "predictions", "summary_tables", "script", and "html". By default, export all listed components except predictions when no prediction or plausible-value object is supplied. Explicitly requesting "predictions" requires at least one such object.
facet	Optional facet for facet_quality_dashboard().
include_person_anchors	If TRUE, include person measures in the exported anchor table.
overwrite	If FALSE, refuse to overwrite existing files.
acknowledge_sensitive	Logical; set to TRUE only after acknowledging that the archive can contain direct person identifiers, person-level estimates, original labels, replay data, and local paths. This suppresses the privacy warning; it does not deidentify any file.
zip_bundle	If TRUE, attempt to zip the written files into a single archive using utils::zip(). This is best-effort and may depend on the local R installation.
zip_name	Optional zip-file name. Defaults to "{prefix}_bundle.zip".
data	Optional original analysis data frame. When supplied, export_mfrm_bundle() co-locates a CSV copy of the data alongside the replay script and updates the script's read.csv() path to point at it. The manifest's input_summary row for data describes the user's untouched input structure; it is not a byte-level equality claim about the CSV replay representation. Default NULL falls back to the legacy your_data.csv placeholder path.

Details

This function is the one-call fit-level archive and HTML route. It reuses mfrmr functions for estimation and diagnostics. When diagnostics = NULL, the exporter computes the diagnostics it needs, then writes the requested CSV/text/replay artifacts and a lightweight HTML page from the fitted object. Use mfrm_results() and mfrm_report() first when you want to inspect a results object before writing files; use export_mfrm_bundle() when the goal is a project-folder bundle from fit. For only the core CSV tables without automatic diagnostics, use export_mfrm_tables(). For a complete saved R result that reloads without refitting, use export_mfrm_results(). The optional "script" component here is a refitting recipe, not a copy of the fitted R object.

The returned written_files and optional ZIP include only files written by this call. With overwrite = TRUE, the ZIP is built afresh and checked before replacing an existing archive; obsolete members are not retained. If ZIP creation fails, any older ZIP remains in place but is not counted as a new output: summary$ZipWritten is zero and notes explains the failure. Unrequested loose files in the folder are left in place.

Every bundle is an analysis archive, not a deidentified or automatically shareable package. Core tables, anchors, predictions, replay sidecars, scripts, and HTML can contain identifying or study-sensitive information. Review and transform every file under the applicable data-handling policy before sharing it.

Value

A named list with class mfrm_export_bundle.

Choosing exports

The include argument lets you assemble a bundle for different audiences:

- "core_tables" for analysts who mainly want CSV output.
- "manifest" for a compact analysis record.
- "script" for reproducibility and reruns. For latent-regression fits, this also writes the fit-level replay person-data sidecar when available.
- "html" for a lightweight summary page. This is not a deidentification guarantee. When replay sidecars are present, the HTML shows an artifact index for them rather than embedding the raw person-level replay table.
- "summary_tables" for manuscript-facing CSV exports of documented summary() surfaces and their compact indexes.
- "visual_summaries" when you want warning maps or residual PCA summaries to travel with the bundle.

Recommended presets

Common starting points are:

- minimal tables: include = c("core_tables", "manifest")
- reporting bundle: include = c("core_tables", "checklist", "dashboard", "summary_tables", "html")
- archival bundle: include = c("core_tables", "manifest", "script", "visual_summaries", "html")

Written outputs

Depending on `include`, the exporter can write:

- core CSV tables via `export_mfrm_tables()`
- checklist CSVs via `reporting_checklist()`
- facet-dashboard CSVs and interpretation notes via `facet_quality_dashboard()`, including screening settings in CSV and HTML; HTML tables show every level's estimates and unavailable diagnostics, while CSVs retain full detail
- APA text files via `build_apa_outputs()`
- manuscript-summary CSVs via `build_summary_table_bundle()`
- anchor CSV via `make_anchor_table()`
- manifest CSV/TXT via `build_mfrm_manifest()`
- exact source-fit readiness CSVs with fit, component, and parameter scope
- visual warning/summary artifacts via `build_visual_summaries()`
- prediction/forecast CSVs via `predict_mfrm_population()`, `predict_mfrm_units()`, and `sample_mfrm_plausible_values()`
- a package-native replay script via `build_mfrm_replay_script()`
- replay-source readiness CSVs retained as provenance; the replayed fit recomputes its own readiness
- for latent-regression fits, a replay-side person-data CSV paired with the replay script
- a lightweight HTML report that bundles the exported tables/text and, for replay sidecars, an artifact summary instead of raw person-level rows

For latent-regression fits, prediction-side artifacts can carry the fitted population-model scoring basis when you explicitly supply the corresponding prediction objects. `predict_mfrm_population()` remains the scenario-level forecast helper, whereas `predict_mfrm_units()` and `sample_mfrm_plausible_values()` are the scoring layer. To keep exports and replay scripts practical, large structural-design schemas from scenario-level population predictions are not flattened into `*_population_prediction_settings.csv` or ADeMP CSVs; the compact simulation specification files carry the replay-relevant settings instead.

For `GPCM`, this exporter is available as a caveated partial bundle over supported diagnostics, report text, visual summaries, manifests, and replay scripts. The returned object and manifest include `gpcm_boundary`. Package-native `GPCM` scorefile export is available with caveats, while full FACETS-style score-side contract review and design forecasting remain outside this bundle contract.

Interpreting output

The returned object reports both high-level bundle status and the exact files written. In practice, `bundle$summary` is the direct status check, while `bundle$written_files` is the file inventory to inspect or hand off to other tools.

Typical workflow

1. Fit a model and compute diagnostics once.
2. Decide whether the audience needs tables only, or also a manifest, replay script, and HTML summary.
3. Call `export_mfrm_bundle()` with a dedicated output directory.
4. Inspect `bundle$written_files` or open the generated HTML file.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model. Supplied unit scores, plausible values and fit-derived forecasts must also match this calibration. A forecast created with `sim_spec` remains a separate scenario; its source is recorded without asserting a match to the fit.

See Also

`fit_mfrm()`, `mfrm_results()`, `mfrm_report()`, `export_mfrm_results()`, `build_mfrm_manifest()`, `build_mfrm_replay_script()`, `export_mfrm_tables()`, `reporting_checklist()`, `export_summary_appendix()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("export_mfrm_bundle", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bundle <- export_mfrm_bundle(
  fit,
  diagnostics = diag,
  output_dir = tempdir(),
  prefix = "mfrmr_bundle_example",
  include = c("core_tables", "manifest", "script", "html"),
  overwrite = TRUE,
  acknowledge_sensitive = TRUE
)
bundle$summary[, c("FilesWritten", "HtmlWritten", "ScriptWritten")]
head(data.frame(
  Component = bundle$written_files$Component,
  File = basename(bundle$written_files$Path)
))
# Full paths and data-handling notes remain in bundle$written_files.

## End(Not run)
```

`export_mfrm_model` *Export a measurement model to Stan or brms*

Description

Write a self-contained Bayesian model bundle without running an estimator. Supports ordinary RSM/PCM and fixed-population one-family GPCM MML fits in both formats. Estimated-population one-family GPCM and two-family GPCM use Stan. Fixed facet/group and step anchors are retained. Sources must be unit-weight, with a fixed standard-normal Person population or (Stan only, one-family GPCM) an estimated intercept-only normal population, and no facet interactions or post-fit shrinkage. Unsupported specifications fail before files are written.

Usage

```
export_mfrm_model(
  fit,
  output_dir,
  format = c("stan", "brms"),
  location_prior_sd = 1,
  step_prior_sd = 1,
  overwrite = FALSE,
  acknowledge_sensitive = FALSE,
  log_slope_prior_sd = 1,
  population_prior = list(mean = c(0, 1), log_sd = c(0, 0.5))
)
```

Arguments

`fit`	A native RSM/PCM or GPCM MML `mfrm_fit` returned by `fit_mfrm()`.
`output_dir`	Destination directory. The bundle includes observed ratings and original level labels; review it before sharing.
`format`	Either `"stan"` (CmdStanR) or `"brms"` (custom family).
`location_prior_sd`	Positive marginal prior SD for each varying facet location; fixed or uniquely determined locations have zero prior SD.
`step_prior_sd`	Positive marginal prior SD for each varying step offset.
`overwrite`	Replace bundle files that already exist? Defaults to `FALSE`.
`acknowledge_sensitive`	Suppress the reminder that ratings and original identifiers are included. Defaults to `FALSE`.
`log_slope_prior_sd`	Positive marginal prior SD of each component log slope for GPCM export. Defaults to 1; inspect prior sensitivity. Not used by RSM/PCM models.

`population_prior`
: Normal prior parameters for an estimated Person population: a named list with `mean = c(location, scale)` for its mean and `log_sd = c(location, scale)` for the log of its standard deviation. Each scale must be positive. Defaults give `mu ~ normal(0, 1)` and `log(sigma) ~ normal(0, 0.5)`. Unused for fixed-population exports.

Details

The default one-family GPCM estimates its population mean and variance. Its intercept-only population can be exported to Stan, retaining the native metric, relative-slope constraint and supported fixed anchors. The generated model uses `theta = mu + sigma * z`, with independent standard-normal `z`. The population mean and SD are sampled, not fixed at their MML estimates. `population_prior` adds explicit Bayesian priors: by default sigma has median 1 and a central 95% prior interval of approximately 0.38 to 2.66. These are starting assumptions, not a population-size recommendation; inspect prior predictions and sensitivity for your rubric and anchors. The displayed Person prior SD includes uncertainty in both mu and sigma, not only sigma. Population regression, estimated-mean models with a fixed SD, and estimated-population brms or two-family exports are not supported. Choosing `gpcm_mml_identification = "fixed_standard_normal"` changes the model; it is not a file-format conversion or a workaround for these exclusions.

Both formats retain the category ladder, facet signs, facet centering and RSM/PCM step ownership of the source specification. RSM/PCM adjacent-category log odds are `theta + signed facet locations - step`. Steps follow their declared within-owner sum targets (zero by default) and are **not ordered**. A binary ladder has its single offset fixed at its target. Fixed-population ability parameters have independent standard-normal priors; realized abilities are not centered across the observed Persons in either population mode.

All supported exports preserve the native logistic metric (L-metric), with scaling constant $D = 1$. No extra 1.7 or 1.702 factor is used. Standard-normal ability priors do not imply a normal-ogive response link. The "Logistic metric (D = 1)" section of `fit_mfrm()` explains the scale and the adjustments needed before comparing other parameterizations.

Centered locations and steps use an orthonormal contrast basis. Independent normal coordinates are scaled so each constrained component has the requested marginal SD. A centered singleton is fixed at zero. Uncentered locations have independent normal priors. These proper structural priors are additions to MML: the Bayesian posterior is not a reproduction of MML point estimates or frequentist confidence intervals. Review prior sensitivity for the application.

Two-family GPCM is available with `format = "stan"` only. Its effective slope is the product of the two component slopes and multiplies the entire adjacent-category predictor, including locations and steps. The first family's log slopes sum to zero; the second family's log slopes are free on the fixed N(0,1) ability scale. Normal priors on orthonormal/free log-slope coordinates give each component the specified marginal log-slope SD. `slope-levels.csv` maps the generated `component_slopes` and `log_component_slopes` to their owners and levels. This preserves the source likelihood, not an MML interval or the exact priors/program of Uto and Ueno (2020). In particular, the extra 1.7 multiplier in their Appendix is absent. One-family GPCM supports both formats. Its single owner's log slopes sum to zero (geometric mean one); its slope also multiplies the entire predictor. A singleton slope owner is fixed at one.

Fixed non-Person facet anchors, group means and step anchors are exact affine constraints. The export preserves whole-facet versus remainder centering and each step owner's target. Priors on the

remaining free contrasts are centered so each free level shares its block's remaining total equally. These priors are defined on the constrained subspace, not obtained by conditioning an earlier unanchored posterior. `levels.csv` and `steps.csv` in these bundles record the affine prior means, marginal SDs and fixed-coordinate indicators; `model-data.rds$constraints` retains the source constraints. One-family bundles use the same affine representation even without anchors. Anchors are fixed inputs: zero posterior spread does not establish certainty about their earlier calibration. Uncertain external calibration is not propagated. Retained step constraints are exported; this does not add a step-anchor fitting argument. Native two-family fitting retains its anchor exclusion. Person anchors require JML in native fitting and remain outside this MML exporter.

`model-data.rds` records the design matrices, priors and mappings used by both formats. `levels.csv` maps original labels to generated parameters; `categories.csv` maps Stan/brms categories to the source score scale. `run.R` fits the exported model and saves draws, diagnostics and replicated ratings. Run it from the bundle directory, after installing CmdStanR/CmdStan and, for the brms format, brms. Neither is required to export a bundle. Both formats support `Rscript run.R --summarize` to regenerate summaries from `fit.rds` without compiling or sampling. Keep `model-data.rds`, `summarize.R` and the brms `model.R` with the saved fit. Read `summary-decision.csv` first, then `sampling-review.csv` for the checks, criteria and next actions. `parameter-summary.csv` joins original labels to estimates, references and score-direction notes. `value_role` distinguishes estimated values, fixed anchors and constraint-determined values. Fixed values have no estimated SD or interval (`NA`); they do not establish zero external calibration uncertainty. Varying parameters retain equal-tailed 95% credible bounds, endpoint MCSE, R-hat and bulk/tail ESS, even during review, with `interval_status` beside the values. `interval_computed` means finite bounds for a varying parameter, not reliable sampling or coverage. `sampler-diagnostics.csv` contains per-chain diagnostics; `posterior-summary.rds` retains all four tables and interpretation notes.

The screen checks at least four chains, R-hat below 1.01, bulk/tail ESS at least 100 per chain, zero divergences/tree-depth hits and E-BFMI at least 0.3. Missing diagnostics require review; fixed coordinates are excluded using model constraints, not sample variance. Tree-depth hits flag efficiency separately from divergences. These checks follow the Stan diagnostic guidance and are not a certificate of validity. Review all varying coordinates, model fit and priors. The larger endpoint MCSE divided by interval width measures simulation precision; no universal acceptable fraction is imposed. Read the matching summary and model-data files with `read_mfrm_posterior()` to use `summary()`, `plot()`, stored credible intervals and `mfrm_report()` without either sampling backend. Keep `fit.rds` for joint-draw calculations. Generated brms callbacks support `log_lik()`, `posterior_predict()` and `posterior_epred()`; the last returns category probabilities.

Log likelihood and replicated ratings condition on the sampled abilities of existing Persons. They do not provide leave-one-Person-out validation or predictions for new Persons. Exporting does not qualify the original fit's convergence, inference, or an external sampler's diagnostics. Repeated ratings retain the source conditional-independence assumption.

Value

Invisibly, a list with `format`, `model`, `output_dir` and named `files`. No model is fitted and no software is installed.

References

Uto, M., & Ueno, M. (2020). A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo. *Behaviormetrika, 47*, 469-496. doi:10.1007/s41237020001157.

Equation 9, Sections 6-7 and Appendix.

See Also

export_mfrm_results(), fit_mfrm()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("export_mfrm_model", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
# Illustrative fixed reference; applications need justified calibration values.
anchor <- data.frame(Facet = "Rater",
                     Level = as.character(ratings$Rater[1]), Anchor = 0)
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                model = "GPCM", method = "MML", step_facet = "Criterion",
                slope_facet = "Criterion", anchors = anchor,
                anchor_policy = "silent")
# Keep the default estimated population. The example data are synthetic.
# Export needs neither Stan nor brms installed and runs no sampler.
stan <- export_mfrm_model(fit, tempfile("mfrmr-stan-"), "stan",
                          acknowledge_sensitive = TRUE)
names(stan$files)
bundle <- readRDS(stan$files[["model-data.rds"]])
bundle$categories # Generated categories mapped to original scores
bundle$population # Estimated mean/SD and their priors
bundle$priors
# Run this bundle's run.R separately if external sampling is wanted.
unlink(stan$output_dir, recursive = TRUE)

## End(Not run)
```

---

export_mfrm_results *Export an mfrm_results analysis archive*

Description

export_mfrm_results() writes the contents of an existing mfrm_results() object to a compact analysis folder. It is a results-download helper for the comprehensive first-screen workflow, not a new estimation, diagnostics, or validation step. The folder is not deidentified or automatically shareable.

Usage

```
export_mfrm_results(
  x,
  output_dir = ".",
  prefix = "mfrmr_results",
```

```
  include = "default",
  preset = NULL,
  overwrite = FALSE,
  acknowledge_sensitive = FALSE,
  zip_bundle = FALSE,
  zip_name = NULL,
  plot_width = 1200,
  plot_height = 900,
  plot_res = 144
)
```

Arguments

x	An mfrm_results() object.
output_dir	Directory where files should be written.
prefix	File-name prefix. Non-alphanumeric characters are converted to underscores.
include	Export components. "default" expands to "summary", "tables", "html", "rds", "replay", and "manifest". "replay" always includes "rds" to reload the complete saved result. Add "report" to write mfrm_report() tables plus Markdown and HTML; add "plots" to write available plot routes as PNG files with separate title/notes CSV files, or use "all".
preset	Optional reader-facing analysis-archive preset. "starter" adds the report and plot routes to the default files and writes index.html with the required Wright map for ordinary models, or the available model-specific figures for testlet and random-rater results.
overwrite	Logical; if FALSE, existing files stop the export.
acknowledge_sensitive	Logical; set to TRUE only after acknowledging that every preset can contain direct person identifiers, person-level results, original labels, local paths, and a complete result object. This suppresses the privacy warning; it does not deidentify any file.
zip_bundle	Logical; if TRUE, create a best-effort zip archive of the written files.
zip_name	Optional zip file name. When omitted, {prefix}_mfrm_results.zip is used.
plot_width, plot_height, plot_res	PNG device settings used when include contains "plots".

Details

The helper writes:

- summary CSVs from summary(x) such as overview, status, triage, plot routes, next actions, mapping, and replay-code lines;
- collected x$tables as CSV files (tables with no columns are omitted; zero-row tables with defined columns retain their headers);
- optional report artifacts from mfrm_report(x), including report-index, evidence-summary, and reporting-template CSVs plus Markdown and HTML;

- a lightweight HTML report equivalent to `mfrm_results(x, output = "html")` for the already-created object;
- an `.rds` copy of the `mfrm_results` object;
- a replay `.R` script that reloads the complete exported result;
- optional PNG figures and matching `_plot_{type}_notes.csv` files. Each CSV retains `plot_data()` notes and their source-specific columns, with suggested titles identified by `Type = "title"`. Titles and explanatory notes remain outside the image; the starter index links each PNG to its CSV. A multi-plot `fit` bundle produces one PNG/CSV pair per member, named `_plot_fit_wright_map`, `_plot_fit_pathway_map` and `_plot_fit_category_characteristic_curves`, rather than one combined PNG;
- a written-files manifest and compact export summary.

For every supported result type, replay reloads the exported RDS without reconstructing the analysis, refitting, rescoring or resampling. Requesting `"replay"` also includes `"rds"`; source-session variables such as `fit` and `diagnostics` are not required. Run the script with `source("path/to/prefix_replay.R", chdir = TRUE)`, replacing the path with the exported script's location, or run it from the exported folder. Use `export_mfrm_bundle()` for a separate refitting script. Result CSV files use UTF-8; when importing with base R, use `read.csv(path, encoding = "UTF-8", check.names = FALSE)` to preserve user-supplied names across locales.

Use `export_mfrm_tables()` when only the fit's core CSV tables are needed. Stored prediction settings, unavailable rows, numerical checks and interval meanings travel with the result. A bootstrap interval may have infinite endpoints; exports retain them instead of substituting finite ordinary intervals.

All presets, including `"starter"`, are analysis archives. In particular, the default `.rds` file retains the complete result object, and CSV, HTML, plot, and replay artifacts can retain direct identifiers or other sensitive study information. The manifest labels each file for review, but the helper does not pseudonymize, redact, or certify an export for sharing. Apply the study's data-governance process before moving the files outside the approved analysis environment.

Plot export is intentionally optional because some plot routes can be comparatively slow or require richer graphics devices. Plot failures are recorded in the returned `plot_errors` table rather than stopping the export. The `"starter"` preset is the recommended reader-oriented analysis archive because, for ordinary models, it requests the Wright map in addition to the result summary, report, replay script, and manifest. Its Infit pathway includes a bounded selection of person rows so person fit can be reviewed without replacing the required Wright-map first screen.

With `overwrite = TRUE`, a requested ZIP is rebuilt from files written by this call; members from an older archive are not retained. ZIP command or content-check failure leaves any older ZIP in place, sets `ZipWritten` to `FALSE`, and explains the failure in `zip_note`. That older archive is not listed as a new output. Unrequested loose files in `output_dir` are neither deleted nor included. The on-disk file manifest and export summary describe the non-ZIP artifacts, including any plot-error CSV; the returned `written_files` and `summary` additionally record a successful ZIP.

Value

An `mfrm_results_export` object with `summary`, `written_files`, `plot_errors`, and zip status fields.

See Also

mfrm_results(), launch_mfrmr_viewer(), export_mfrm_bundle(), export_summary_appendix()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("export_mfrm_results", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Fit these ratings; replace the data-loading line for your own analysis
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

res <- mfrm_results(fit)

# Create a new temporary folder for this example's files
# For your own analysis, use a permanent folder you can write to
output_dir <- tempfile("mfrmr-example-")
exported <- export_mfrm_results(
  res,
  output_dir = output_dir,
  preset = "starter",
  acknowledge_sensitive = TRUE # These data are synthetic; exports retain IDs
)

# Preview filenames and check whether any plots could not be exported
head(data.frame(
  Component = exported$written_files$Component,
  File = basename(exported$written_files$Path)
))
exported$plot_errors
# Full paths remain in exported$written_files$Path.
# Open index.html in output_dir to read the report.

## End(Not run)
```

---

export_mfrm_tables *Export fitted MFRM tables to CSV files*

Description

Writes tidy CSV files suitable for import into spreadsheet software or further analysis in other tools. This is the CSV-only route from a fitted model. It does not run diagnostics, save a complete R object, or create a replay script. Use `export_mfrm_results()` to archive an existing results object and reload it without refitting, or `export_mfrm_bundle()` for a fit-level collection with an optional refitting script. In 0.2.4 this replaces `export_mfrm()`; arguments, CSV filenames and the returned file table are unchanged. Enriched facet CSVs now also retain the diagnostic availability and uncertainty fields.

Usage

```
export_mfrm_tables(
  fit,
  diagnostics = NULL,
  output_dir = ".",
  prefix = "mfrm",
  tables = c("person", "facets", "summary", "steps", "measures"),
  overwrite = FALSE,
  acknowledge_sensitive = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm`.
`diagnostics`	Optional output from `diagnose_mfrm`. When provided, enriches facet estimates with SE, fit statistics, their calculation/availability fields and reasons, and writes the full measures table. Must belong to the same `fit`; mismatched or outdated diagnostics are rejected before creating files.
`output_dir`	Directory for CSV files. Created if it does not exist.
`prefix`	Filename prefix (default `"mfrm"`).
`tables`	Character vector of tables to export. Any subset of `"person"`, `"facets"`, `"summary"`, `"steps"`, `"measures"`. Default exports all available tables.
`overwrite`	If `FALSE` (default), refuse to overwrite existing files.
`acknowledge_sensitive`	Logical; set to `TRUE` only after acknowledging that these tables can contain direct person identifiers, person-level estimates, and original facet labels. This suppresses the privacy warning; it does not deidentify any file.

Value

Invisibly, a data.frame listing written files, their paths, and explicit privacy/data-handling metadata. `Deidentified` and `ShareableWithoutReview` are always `FALSE`.

Exported files

`{prefix}_person_estimates.csv` Person ID, Estimate, SD.

`{prefix}_facet_estimates.csv` Facet, Level, Estimate, and optionally SE, Infit, Outfit, PTMEA and their qualification fields when diagnostics are supplied.

`{prefix}_fit_summary.csv` One-row model summary.

`{prefix}_step_parameters.csv` Step/threshold parameters.

`{prefix}_measures.csv` Full measures table (requires diagnostics).

Interpreting output

The returned data.frame tells you exactly which files were written and where. This is convenient for scripted pipelines where the output directory is created on the fly. The files are analysis tables, not a deidentified sharing package; review each file under the applicable data-handling policy before sharing it. In enriched facet tables, read `SE_Method`, `SEUse`, `CIUse`, `InferenceReady` and `SupportsFormalInference` with the SE. Read `FitStatus` and `FitReason` with Infit/Outfit. A finite value does not imply valid formal inference. These columns preserve the matching diagnostics' qualifications; the export does not compute a new SE or confidence interval. Missing fields in an older diagnostic result are not filled with assumed approval.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Optionally compute diagnostics with `diagnose_mfrm()` when you want enriched facet or measures exports.
3. Call `export_mfrm_tables(...)` and inspect the returned `Path` column.

See Also

`fit_mfrm`, `diagnose_mfrm`, `as.data.frame.mfrm_fit`, `export_mfrm_results()`, `export_mfrm_bundle()`, mfrmr_compatibility_layer

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Use the diagnostics saved with this fit
diagnostics <- example_fit$diagnostics

# Use a new temporary folder for this example; choose a permanent one for your work
output_dir <- tempfile("mfrmr-tables-")
files <- export_mfrm_tables(
  fit,
  diagnostics = diagnostics,
  output_dir = output_dir,
  acknowledge_sensitive = TRUE # Synthetic data; exported tables retain person IDs
)
```

```
# Preview filenames; full paths remain in files$Path
data.frame(Table = files$Table, File = basename(files$Path))
# Open a path from files$Path in a spreadsheet app or with read.csv()
```

`export_summary_appendix` *Export manuscript appendix tables from documented summary surfaces*

Description

Export manuscript appendix tables from documented summary surfaces

Usage

```
export_summary_appendix(
  x,
  output_dir = ".",
  prefix = "mfrmr_appendix",
  include_html = TRUE,
  preset = c("all", "recommended", "compact", "methods", "results", "diagnostics",
    "reporting"),
  overwrite = FALSE,
  zip_bundle = FALSE,
  zip_name = NULL,
  digits = 3,
  top_n = 10,
  preview_chars = 160
)
```

Arguments

`x`	A supported `summary()` source, a prebuilt `build_summary_table_bundle()` result, or a named list of such objects.
`output_dir`	Directory where files will be written.
`prefix`	File-name prefix for written artifacts.
`include_html`	If `TRUE`, also write a lightweight HTML appendix page.
`preset`	Appendix table-selection preset: `"all"` keeps every returned summary table, `"recommended"` keeps manuscript-facing summary tables while dropping bridge-only or preview-only surfaces, and `"compact"` keeps a smaller reviewer-facing subset. Section-aware presets `"methods"`, `"results"`, `"diagnostics"`, and `"reporting"` keep only the returned tables classified to those appendix sections in the summary-table catalog.
`overwrite`	If `FALSE`, refuse to overwrite existing files.
`zip_bundle`	If `TRUE`, attempt to zip the written appendix artifacts.

zip_name	Optional zip-file name. Defaults to "{prefix}_appendix.zip".
digits	Digits forwarded when raw objects must be normalized through build_summary_table_bundle().
top_n	Row cap forwarded when raw objects must be normalized through build_summary_table_bundle().
preview_chars	Character cap forwarded when APA-output summaries must be normalized through build_summary_table_bundle().

Details

This helper is the narrow public bridge from documented summary() surfaces to manuscript appendix artifacts. It accepts the same reporting objects that build_summary_table_bundle() supports, exports their table bundles as CSV, and optionally assembles a lightweight HTML appendix page.

Inspect written_files for the actual file paths. An analysis table whose name conflicts with bundle metadata (for example, overview) is retained with a table_ prefix, with a numeric suffix if needed for uniqueness. Metadata keeps its existing filename; the analysis table is also retained in HTML. Summary rounding follows digits; retain the source R object for full-precision reuse. ZIP archives contain only files written by this call. With overwrite = TRUE an existing ZIP is replaced after the new archive is built and checked; obsolete members are not carried forward. If ZIP creation fails, any older ZIP is retained, summary$ZipWritten is zero and notes explains the failure. Other files in the destination folder are left in place.

Fit-level caveats are exported through the analysis_caveats role, and pre-fit score-support caveats are exported through the score_category_caveats role. Both roles are classified as diagnostics, so they remain available under "recommended" and "diagnostics" presets when the source summary contains caveat rows.

Precision-review summaries keep fit_separation_basis in the exported precision-review role so fit, ZSTD, separation/reliability/strata, and package review thresholds can be reported without turning them into universal recovery criteria. Fit-measure and FACETS fit-review summaries keep df/ZSTD sensitivity and optional external FACETS matching tables in the same precision-review section.

Parameter-recovery studies can be exported by passing simulate_mfrm_recovery() or assess_mfrm_recovery() output directly. The exported bundle keeps the ADEMP-style simulation basis, recovery metrics, replication status, adequacy checklist, thresholds, and next actions in separate appendix-ready tables.

A compatible precomputed recovery-evidence summary can also be exported, including its case assessments, condition notes, diagnostic notes, and domain assessments.

Unlike export_mfrm_bundle(), this helper does not require a fitted model. It is intended for the stage where compact reporting summaries already exist and the task is to hand off appendix-ready tables, catalogs, and reporting maps.

Value

A named list of class mfrm_summary_appendix_export with:

- summary
- written_files
- selection_summary

- selection_table_summary
- selection_section_table_summary
- selection_handoff_table_summary
- selection_handoff_preset_summary
- selection_handoff_summary
- selection_handoff_bundle_summary
- selection_handoff_role_summary
- selection_handoff_role_section_summary
- selection_role_summary
- selection_section_summary
- selection_catalog
- settings
- notes

Typical workflow

1. Build `summary(...)` objects from fit, diagnostics, data description, reporting checklist, or APA outputs.
2. Call `export_summary_appendix(...)` on one object or a named list.
3. Hand off the written CSV/HTML appendix artifacts to manuscript or QA workflows.

See Also

`build_summary_table_bundle()`, `export_mfrm_bundle()`, `apa_table()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("export_summary_appendix", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
appendix <- export_summary_appendix(
  list(fit = fit, diagnostics = diag),
  output_dir = tempdir(),
  prefix = "mfrmr_appendix_example",
  include_html = TRUE,
  overwrite = TRUE
)
appendix$summary

## End(Not run)
```

`extract_mfrm_sim_spec` *Derive a simulation specification from a fitted MFRM object*

Description

Derive a simulation specification from a fitted MFRM object

Usage

```
extract_mfrm_sim_spec(
  fit,
  assignment = c("auto", "crossed", "rotating", "resampled", "skeleton"),
  latent_distribution = c("normal", "empirical"),
  source_data = NULL,
  person = NULL,
  group = NULL,
  population_assignment = c("matched", "independent")
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`assignment`	Assignment design to record in the returned specification. Use `"resampled"` to reuse empirical person-level rater-assignment profiles from the fitted data, or `"skeleton"` to reuse the observed person-by-facet design skeleton from the fitted data.
`latent_distribution`	Latent-value generator to record in the returned specification. `"normal"` stores spread summaries for parametric draws; `"empirical"` additionally activates centered empirical resampling from the fitted person/rater/criterion estimates.
`source_data`	Optional original source data used to recover additional non-calibration columns, currently person-level `group` labels, when building a fit-derived observed response skeleton.
`person`	Optional person column name in `source_data`. Defaults to the person column recorded in `fit`.
`group`	Optional group column name in `source_data` to merge into the returned `design_skeleton` as person-level metadata.
`population_assignment`	How an active population model is associated with `"resampled"` or `"skeleton"` assignments. The default `"matched"` draws one template Person jointly for covariates and assignments. Both tables must cover the same `TemplatePerson` IDs, with matching `Group` labels if present in both. `"independent"` samples covariate rows separately and intentionally breaks these associations. In that case both tables cannot define `Group`; rename a covariate if it represents a different grouping. For other assignment designs this setting has no effect. Population-template `Group`, when present, supplies the response group; `group_levels` must include all template labels and does not rebalance them.

Details

`extract_mfrm_sim_spec()` uses a fitted model as a practical starting point for later simulation studies. It extracts:

- design counts from the fitted data
- empirical spread of person and facet estimates
- optional empirical support values for semi-parametric draws
- fitted threshold values
- either a simplified assignment summary (`"crossed"` / `"rotating"`), empirical resampled assignment profiles (`"resampled"`), or an observed response skeleton (`"skeleton"`, optionally carrying `Group`/`Weight`)
- when the fit used the latent-regression branch, the fitted `population_formula`, coefficient vector, residual variance, and the stored person-level covariate table, including model-matrix xlevel and contrast provenance for categorical covariates

This is intended as a **fit-derived parametric starting point**, not as a claim that the fitted object perfectly recovers the true data-generating mechanism. Users should review and, if necessary, edit the returned specification before using it for design planning. Observed skeleton rows retain repeated rating occasions. Remove accidental duplicate source rows before fitting; the generator cannot distinguish them from intentional repeated ratings. Covariates and assignment templates are linked by `TemplatePerson` under `population_assignment = "matched"`.

Without an active population regression, `theta_sd` is the empirical SD of the reported Person estimates, including any shrinkage in those estimates. It is not an estimate of the latent population SD corrected for measurement error. The empirical option resamples centered estimates independently of their original IDs; it is not a draw from the source posterior. With an active population model, its coefficients, residual variance and fitted transformation basis replace this Person-spread approximation. Neither route propagates uncertainty in the source fit's generating parameters.

`GPCM` fits with the same slope and step owner are supported here for direct data generation and parameter-recovery checks, provided that the returned simulation specification stores both a threshold table and a parallel slope table. The same fit-derived specification can feed caveated role-based design evaluation, population forecasting, and fit-based report/export bundles. Diagnostic and signal-detection design screening is available with explicit caveats. Full FACETS score-side contract review, posterior predictive checks, and MCMC estimation are not available for `GPCM`.

If you want to carry person-level group labels into a fit-derived observed response skeleton, provide the original `source_data` together with `person` and `group`. Group labels are treated as person-level metadata and are checked for one-label-per-person consistency before being merged.

Value

An object of class `mfrm_sim_spec`.

Interpreting output

The returned object is a simulation specification, not a prediction about one future sample. It captures one convenient approximation to the observed design and estimated spread in the fitted run.

See Also

build_mfrm_sim_spec(), simulate_mfrm_data()

Examples

```
toy <- simulate_mfrm_data(
  n_person = 8,
  n_rater = 3,
  n_criterion = 2,
  seed = 123
)
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
spec <- extract_mfrm_sim_spec(fit, latent_distribution = "empirical")
spec$assignment
spec$model
head(spec$threshold_table)
```

---

facets_chisq_table *Build facet variability diagnostics with fixed/random reference tests*

Description

Build facet variability diagnostics with fixed/random reference tests

Usage

```
facets_chisq_table(
  fit,
  diagnostics = NULL,
  fixed_p_max = 0.05,
  random_p_max = 0.05,
  top_n = NULL
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
fixed_p_max	Warning cutoff for fixed-effect chi-square p-values.
random_p_max	Warning cutoff for random-effect chi-square p-values.
top_n	Optional maximum number of facet rows to keep.

Details

This helper summarizes facet-level variability with fixed and random chi-square indices for spread and heterogeneity checks.

Value

A named list with:

- `table`: facet-level chi-square diagnostics
- `summary`: one-row summary
- `thresholds`: applied p-value thresholds

Interpreting output

- `table`: facet-level fixed/random chi-square and p-value flags.
- `summary`: full-family available/unavailable counts and screen-positive counts, before `top_n` limits displayed rows.
- `thresholds`: p-value criteria used for flagging.

Use this table together with inter-rater and displacement diagnostics to distinguish global facet effects from local anomalies.

Typical workflow

1. Run `facets_chisq_table(fit, ...)`.
2. Inspect `summary(chi)` then facet rows in `chi$table`.
3. Visualize with plot(x, type = "facets_chisq").

Output columns

The `table` data.frame contains:

Facet Facet name.

Levels Number of estimated levels in this facet.

MeanMeasure, SD Mean and standard deviation of level measures.

FixedChiSq, FixedDF, FixedProb Fixed-effect chi-square test (null hypothesis: all levels equal). Significant result means the facet elements differ more than measurement error alone.

RandomChiSq, RandomDF, RandomProb, RandomVar FACETS-form random spread index and reference tail area. It does not establish normality or random sampling: with equal SEs and positive adjusted variance the statistic is identically eligible-level count minus one, regardless of distributional shape. Full covariance is not propagated.

FixedFlag, RandomFlag Logical flags for significance.

See Also

`diagnose_mfrm()`, `interrater_agreement_table()`, plot(x, type = "facets_chisq")

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("facets_chisq_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
chi <- facets_chisq_table(fit)
summary(chi)
p_chi <- plot(chi, draw = FALSE)
p_chi$data$plot

## End(Not run)
```

---

facets_feature_coverage

FACETS Feature Coverage Matrix

Description

`facets_feature_coverage()` summarizes how `mfrmr` maps the main FACETS output-table, output-file, and graph-menu surface to package functions. It is a surface-coverage guide, not a statement that estimands, conditioning, extreme-score handling, degrees of freedom, or numerical results are equivalent.

Use this helper before migration work when you need a public, user-facing answer to three questions:

- which FACETS outputs have a close `mfrmr` route,
- which outputs are only partially covered by structured R objects,
- which FACETS-specific outputs are not implemented or intentionally outside the current package scope.

Usage

```
facets_feature_coverage(
 status = c("all", "implemented", "supported_with_caveat", "partial", "not_implemented",
    "not_targeted")
)
```

Arguments

`status`	Which rows to return. `"all"` returns the full matrix. Other values filter by the `Status` column.

Details

The matrix is based on the FACETS 64-bit output index, which lists output Tables 1–14, DIF/bias plots, R/Web plots, output files, and graph-menu curves. `mfrmr` intentionally prioritizes structured R tables and reusable plot data over exact FACETS line-printer output.

The executed comparison uses FACETS 64-bit 4.5.0. The online update history lists 4.5.3 (October 2026); current web help is not a version-pinned executable specification. In particular, Table 13 web df and executed df differ. The 4.5.1 history replaces the gtheory interface while its help still names gtheory; mfrmr's independent G-study does not depend on that menu. Numerical comparisons require their recorded executable version, design and estimand.

Status meanings:

- `implemented`: a package-native route covers the substantive output surface; this status alone does not claim an externally matched statistical contract.
- `supported_with_caveat`: a package-native route exists, but the output must be read with explicit identification, validation, or scope caveats.
- `partial`: the concept is covered, but not the full FACETS formatting, option surface, file type, or external integration.
- `not_implemented`: a FACETS feature has no direct package-native route in the documented package scope.
- `not_targeted`: the feature is tied to FACETS UI, Web/Excel handoff, or another external program format and is outside the package scope.

The four contract axes are deliberately independent:

- `SurfaceCoverage` records whether a corresponding package surface exists; familiar visual grammar belongs only to this axis.
- `StatisticalContract` records the package-native statistical scope and never implies a FACETS-matched estimand.
- `ValidationEvidence` records whether this table establishes matched external numerical evidence. It currently does not; that conclusion requires a separate comparison using a matched model and estimand.
- `OperationalStatus` records route availability. A package route being available does not make `mfrmr` operationally interchangeable with FACETS.

For observed-score assessment planning, `mfrm_g_study()` and `mfrm_d_study()` support their documented univariate or multivariate component designs. The older `mfrm_generalizability()` / `mfrm_d_study()` pair has a different main-effects decomposition. Neither route reproduces the FACETS interface to the external `gtheory` package. For balanced ANOVA, `confint.mfrm_multivariate_gstudy()` adds component SEs/intervals for a score or fixed composite under normal random effects; two-score covariance intervals have explicit generalized and conservative choices with signed bounds. Explicit `reml_profile` variance and covariance intervals also accept MINQUE(0), with a separate REML center and no sampling SE; two-score covariance profiles fit a joint normal model and retain signed bounds. `vcov.mfrm_multivariate_gstudy()` separately supplies component sampling covariance/SEs for balanced ANOVA and identifiable MINQUE sources using normal product moments; these describe the original estimator, not REML. Individual G/Phi intervals from balanced two-crossed-facet sources default to approximate MLS null inversion. Undefined MLS widths are conservatively retained and flagged; numerical availability does not guarantee coverage.

G/Phi and SEM have an explicit experimental `reml_profile` option for balanced or identifiable incomplete sources; the separate REML fit retains original ANOVA/MINQUE points and does not guarantee finite-sample or boundary coverage. With `reference`, it maps one shared profile interval for one-facet plans or nested plans with a common future child count, including incomplete sources. Balanced crossed G also accepts explicit experimental `score` intervals, retaining empty-set and numerical diagnostics; Phi is not supplied by this route. `mfrm_multivariate_d_compare()` supplies a separate, limited normal-theory paired-plan calculation. See `vignette("mfrmr-gtheory", package = "mfrmr")` for a runnable workflow.

Value

A data.frame with columns:

- `FACETSArea`
- `FACETSFeature`
- `FACETSReference`
- `mfrmrRoute`
- `Status`
- `SurfaceCoverage`
- `StatisticalContract`
- `ValidationEvidence`
- `OperationalStatus`
- `Capability`
- `Limitation`
- `Alternative`

References

Linacre, J. M. (2026). *A user's guide to FACETS, version 4.5.0*. Winsteps.com. See the guide's output-table index for the documented FACETS files, plots, and graphs.

See Also

`facets_positioning_guide()`, `mfrmr_output_guide()`, `facets_fit_df_guide()`, `read_facets_fit_table()`, `facets_fit_review()`, `gpcm_capability_matrix()`

Examples

```
facets_feature_coverage()
facets_feature_coverage("partial")
facets_feature_coverage("supported_with_caveat")
facets_feature_coverage("not_implemented")
```

facets_fit_df_guide *Guide FACETS-style fit df and ZSTD standardization*

Description

`facets_fit_df_guide()` gives a compact user-facing guide to the degrees of freedom and ZSTD standardization choices used when comparing mfrmr fit output with FACETS-style fit tables.

Usage

```
facets_fit_df_guide(include_references = TRUE)
```

Arguments

`include_references`
: If `TRUE`, include source-reference rows for the FACETS/Winsteps documentation and Rasch measurement texts that motivate the guide.

Details

The guide separates mean-square size from ZSTD standardization. Infit and outfit MnSq values answer how large the residual noise or predictability signal is. ZSTD values standardize those MnSq values using a degrees-of- freedom convention and a Wilson-Hilferty-style transformation, so ZSTD can differ even when the underlying MnSq values are nearly identical.

Two boundaries sit upstream of any df comparison. First, the residual basis: `method = "MML"` fits evaluate residuals at shrunken EAP person measures, whereas FACETS evaluates them at JMLE estimates, so MnSq values themselves can differ before any standardization is applied; refit with `method = "JML"` when the comparison requires a JMLE-style residual basis. Second, small df: the explicit engine convention withholds ZSTD below df=1, whereas the default FACETS convention admits positive df and caps ZSTD at +/-9. Both FACETS WHEXACT choices use a cube root; NO omits q/3. Small-df availability differences require checking the selected convention.

Value

A bundle of class `mfrm_facets_fit_df_guide` with:

- `summary`: one-row scope summary
- `formula_guide`: formulas and package columns
- `column_guide`: where engine and FACETS-style columns appear
- `decision_guide`: recommended comparison steps
- `interpretation_guide`: how to read common difference patterns
- `references`: optional source-reference rows
- `settings`: guide metadata

See Also

`diagnose_mfrm()`, `fit_measures_table()`, `facets_fit_review()`

Examples

```
facets_fit_df_guide()
facets_fit_df_guide()$decision_guide
```

`facets_fit_review` *Review fit standardization against FACETS-style ZSTD conventions*

Description

Review fit standardization against FACETS-style ZSTD conventions

Usage

```
facets_fit_review(
  fit,
  diagnostics = NULL,
  facets_fit = NULL,
  facet_col = NULL,
  level_col = NULL,
  mnsq_tolerance = 0.01,
  external_zstd_tolerance = 0.05,
  df_tolerance = 0.5,
  df_zstd_tolerance = 0.05,
  df_zstd_large_shift = 0.5,
  df_ratio_tolerance = 0.05
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. If it does not contain FACETS-style fit columns, diagnostics are recomputed with `fit_df_method = "both"` and `residual_pca = "none"`. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
`facets_fit`	Optional external FACETS fit table, or a list of such tables. The helper matches rows by `Facet` and `Level`; a person-only table with a `Person` column is also accepted.
`facet_col, level_col`	Optional explicit column names for the external FACETS table when automatic detection is not sufficient.
`mnsq_tolerance, external_zstd_tolerance, df_tolerance`	Numeric tolerances used to classify external FACETS-vs-mfrmr differences.
`df_zstd_tolerance`	Smallest absolute engine-vs-FACETS-style ZSTD difference treated as interpretively visible rather than rounding noise in `df_sensitivity`. Default `0.05`.

`df_zstd_large_shift`
: Absolute engine-vs-FACETS-style ZSTD difference labeled `large_zstd_shift` when the |ZSTD| flag status is unchanged. Default `0.5`.

`df_ratio_tolerance`
: Relative df-difference tolerance used to classify the within-mfrmr engine-vs-FACETS-style df difference; for example, `0.05` means a 5 percent df difference.

Details

This helper separates two questions that are often conflated when comparing mfrmr output with FACETS:

- how much the package-native `engine` ZSTD changes when the same MnSq values are standardized with the FACETS/Wright-Masters fourth-moment df convention;
- when an external FACETS table is supplied, whether the FACETS-reported rows match mfrmr's FACETS-style companion columns closely enough for practical reporting.

The review is row-matched by `Facet` and `Level`. It treats MnSq, ZSTD, and df differences separately because FACETS documentation makes the df convention and Wilson-Hilferty/WHEXACT handling central to ZSTD interpretation.

Two prior limitations also apply. For `method = "MML"` fits, residuals are evaluated at shrunken EAP person measures while FACETS uses JMLE estimates, so MnSq itself can differ before standardization; refit with `method = "JML"` for a JMLE-style residual basis. And mfrmr withholds ZSTD as `NA` when the applicable df falls below 1 (Wilson-Hilferty instability), while FACETS under `WHEXACT` can report a value on the same sparse cell; such NA-vs-finite pairs are availability differences, not fit differences. Both notes are repeated in the returned `guidance` table.

Value

An `mfrm_facets_fit_review` bundle with:

- `summary`: one-row overview of within-mfrmr and external comparison counts
- `standardization`: the fit-standardization guide from diagnostics
- `df_sensitivity`: engine-vs-FACETS-style df/ZSTD comparison using the same row-level status taxonomy as `fit_measures_table()$df_sensitivity`
- `df_sensitive`: subset of `df_sensitivity` whose df convention changes the |ZSTD| flag or materially changes ZSTD interpretation
- `df_sensitivity_summary`: counts by df-sensitivity status
- `external_table_quality`: completeness and duplicate-key review for the supplied FACETS fit table, including reported-token and displayed-ZSTD boundary counts
- `external_comparison`: optional external FACETS-vs-mfrmr comparison
- `df_conversion_guide`: formulas, column map, and comparison decisions for FACETS-style df/ZSTD review
- `guidance`: interpretation notes
- `settings`: tolerances and review metadata

See Also

diagnose_mfrm(), facets_output_contract_review(), mfrmr_compatibility_layer

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("facets_fit_review", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
review <- facets_fit_review(fit)
summary(review)

## End(Not run)
```

---

facets_output_contract_review

Build a FACETS output-contract review

Description

Build a FACETS output-contract review

Usage

```
facets_output_contract_review(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  branch = c("facets", "original"),
  contract_file = NULL,
  include_metrics = TRUE,
  top_n_missing = 15L
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). If omitted, diagnostics are computed internally with residual_pca = "none". Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
bias_results	Optional output from estimate_bias(). If omitted and at least two facets exist, a 2-way bias run is computed internally.
branch	Contract branch. "facets" checks FACETS-style contract columns. "original" adapts branch-sensitive contracts to the package's compact naming.

`contract_file` Optional path to a custom contract CSV.

`include_metrics`
If `TRUE`, run additional numerical consistency checks.

`top_n_missing` Number of lowest-coverage contract rows to keep in `missing_preview`.

Details

This function checks produced report components against a FACETS-style output-contract specification (`inst/references/facets_column_contract.csv`) and returns:

- column-level coverage per contract row
- table-level coverage summaries
- optional metric-level consistency checks

It is intended to show users which FACETS-style output fields have a package-native counterpart and which remain unavailable. It does not establish external validity or software equivalence beyond the specific schema/metric contract encoded in the contract file.

Value

An object of class `mfrm_facets_contract_review` with:

- `overall`: one-row output-contract review summary
- `column_summary`: coverage summary by table ID
- `column_review`: row-level output-contract review
- `missing_preview`: lowest-coverage rows
- `metric_summary`: one-row metric-check summary
- `metric_by_table`: metric-check summary by table ID
- `metric_checks`: row-level metric checks
- `settings`: branch/contract metadata

GPCM boundary

This helper is unavailable for `GPCM` fits because the FACETS output contract includes score-side rows whose measure-to-score and uncertainty semantics are supported for the Rasch-family route, not for free-discrimination `GPCM`. Use `gpcm_capability_matrix()` before routing a `GPCM` fit into score-side compatibility-output helpers.

Coverage interpretation in `overall`:

- `MeanColumnCoverage` and `MinColumnCoverage` are computed across all contract rows (unavailable rows count as 0 coverage).
- `MeanColumnCoverageAvailable` and `MinColumnCoverageAvailable` summarize only rows whose source component is available.

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_facets_contract_review` (`type = "column_coverage"`, `"table_coverage"`, `"metric_status"`, `"metric_by_table"`).

Interpreting output

- `overall`: high-level output-contract coverage and metric-check pass rates.
- `column_summary` / `column_review`: where output-schema mismatches occur.
- `metric_summary` / `metric_checks`: numerical consistency checks tied to the current contract.
- `missing_preview`: direct path to unresolved output-contract gaps.

Typical workflow

1. Run `facets_output_contract_review(fit, branch = "facets")`.
2. Inspect `summary(contract_review)` and `missing_preview`.
3. For unresolved rows, use the documented package-native alternative or retain the scope limitation in the report.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `build_fixed_reports()`, mfrmr_compatibility_layer

Examples

```
# Long-running workflow; run explicitly when needed.
# example("facets_output_contract_review", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
contract_review <- facets_output_contract_review(fit, diagnostics = diag, branch = "facets")
summary(contract_review)
p <- plot(contract_review, draw = FALSE)

## End(Not run)
```

---

facets_output_file_bundle

Build native graph and observation-score output files

Description

Build native graph and observation-score output files

Usage

```
facets_output_file_bundle(
  fit,
  diagnostics = NULL,
  include = c("graph", "score"),
  theta_range = c(-6, 6),
  theta_points = 241,
  digits = 4,
  score_se_method = c("both", "native", "score_side", "none"),
  include_fixed = FALSE,
  fixed_max_rows = 400,
  write_files = FALSE,
  output_dir = NULL,
  file_prefix = "mfrmr_output",
  overwrite = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used for score file). Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics. Exported score rows are observations, whereas FACETS Scorefile rows are elements. These formats are not interchangeable.
include	Output components to include: "graph" and/or "score".
theta_range	Theta/logit range for graph coordinates.
theta_points	Number of points on the theta grid for graph coordinates.
digits	Rounding digits for numeric fields. ConditionalVariance in the score table and delimited export retains full precision so small positive variances remain distinguishable from zero.
score_se_method	For GPCM scorefile exports, which observation-level score uncertainty columns to compute. "both" (default) includes native structural expected-score SEs and score-side delta-method SEs; "native" includes only the structural expected-score route; "score_side" includes only the score-side delta route; "none" records explicit not_requested status columns.
include_fixed	If TRUE, include fixed-width text mirrors of output tables.
fixed_max_rows	Maximum rows shown in fixed-width text blocks.
write_files	If TRUE, write selected outputs to files in output_dir.
output_dir	Output directory used when write_files = TRUE.
file_prefix	Prefix used for output file names.
overwrite	If FALSE, existing output files are not overwritten.

Details

Legacy-compatible output files often include:

- graph coordinates for Table 8 curves (GRAPH= / Graphfile=), and
- observation-level modeled score lines (SCORE=-style inspection).

This helper returns both as data frames and can optionally write CSV/fixed-width text files to disk.

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_output_bundle (type = "graph_expected", "score_residuals", "obs_probability", "score_se").

Value

A named list including:

- graphfile / graphfile_syntactic when "graph" is requested
- scorefile when "score" is requested
- graphfile_fixed / scorefile_fixed when include_fixed = TRUE
- written_files when write_files = TRUE
- settings: applied options

Interpreting output

- graphfile: legacy-compatible wide curve coordinates (human-readable labels).
- graphfile_syntactic: same curves with syntactic column names for programmatic use.
- scorefile: observation-level observed/expected/residual diagnostics.
- written_files: traceability record of files produced when write_files = TRUE.

For reproducible pipelines, prefer graphfile_syntactic and keep written_files in run logs.

Preferred route for new analyses

For new scripts, prefer category_curves_report() or category_structure_report() for scale outputs, then use export_mfrm_bundle() for file handoff. Use facets_output_file_bundle() for its native graph/observation-row contract. Its score file is not the element-level FACETS Scorefile format; use read_facets_fit_table() to import actual FACETS element output.

GPCM boundary

For GPCM, graph output and package-native scorefile output are available with caveats. include = "score" returns observation-level fitted expected score, residual, standardized residual, observed-category probability, GPCM slope fields, and native structural delta-method expected-score uncertainty and/or score-side delta-method SEs when the required MML diagnostics are available. Use score_se_method to choose "both" (default), "native", "score_side", or "none". The score-side route transforms a logit-side standard error with the GPCM expected-score derivative $dE[X]/d\eta = \alpha Var(X)$, where ScoreSlope is α. ScoreSideLogitSE remains on the logit side; ScoreSideSE and its interval columns are on the expected-score scale. The scorefile also carries explicit score-side caveat columns. It is not a FACETS score-side equivalence file, does not export FACETS-equivalent score-side standard errors, and does not establish an operational score-scale decision. Use gpcm_capability_matrix() with view = "scores" and gpcm_capability_matrix() for the current scope.

Typical workflow

1. Fit and diagnose model.
2. Generate bundle with `include = c("graph", "score")`.
3. Validate with `summary(out)` / `plot(out)`.
4. Export with `write_files = TRUE` for reporting handoff.

See Also

`category_curves_report()`, `diagnose_mfrm()`, `unexpected_response_table()`, `export_mfrm_bundle()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
# Long-running workflow; run explicitly when needed.
# example("facets_output_file_bundle", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
out <- facets_output_file_bundle(fit, diagnostics = diagnose_mfrm(fit, residual_pca = "none"))
summary(out)
p_out <- plot(out, draw = FALSE)
p_out$data$plot

## End(Not run)
```

---

facets_positioning_guide

FACETS Positioning Guide

Description

`facets_positioning_guide()` gives user-facing wording for the relationship between `mfrmr` and FACETS. Use it when a report, migration note, or methods appendix must make clear that `mfrmr` is not a FACETS numerical clone.

Usage

```
facets_positioning_guide()
```

Details

The guide separates five ideas that are easy to conflate:

- estimation authority: fitted values come from `mfrmr` unless external FACETS output is explicitly supplied;

- compatibility purpose: FACETS-style names and files are transition, handoff, and report-organization surfaces;
- external comparison: FACETS comparisons require a supplied external table and should separate MnSq differences from df/ZSTD convention differences;
- current model boundary: one ordered-categorical response-model family and one observed score scale are used per fit; nominal and count-response families, mixed families, multiple independent scales, general threshold anchoring, and importing FACETS or third-party frozen calibrations are not part of the current public estimator; native portable calibration is available within the scope of `mfrm_calibration_capabilities()`, and posterior scoring from an existing fitted object remains separate;
- extension surface: native R tables, plot data, GPCM diagnostics, network views, and G/D-study helpers are package extensions, not promises of FACETS menu-level reproduction.

Value

A data.frame with columns:

- `Topic`
- `Position`
- `RecommendedWording`
- `PrimaryRoute`

See Also

`facets_feature_coverage()`, `mfrmr_output_guide()`, `read_facets_fit_table()`, `facets_fit_review()`

Examples

```
facets_positioning_guide()
```

---

facets_term_crosswalk *FACETS-to-mfrmr term crosswalk*

Description

`facets_term_crosswalk()` maps common FACETS report and specification terms to their closest `mfrmr` objects or routes. The relationship column makes explicit whether a row is a substantive counterpart, a presentation convention, or only a migration aid.

Usage

```
facets_term_crosswalk()
```

Value

A data.frame with `FACETSTerm`, `mfrmrTerm`, `mfrmrRoute`, `Relationship`, and `Boundary` columns.

See Also

facets_positioning_guide(), facets_feature_coverage(), facets_visual_contract()

Examples

```
facets_term_crosswalk()
```

---

facets_visual_contract *FACETS-facing visual contract*

Description

facets_visual_contract() identifies the closest package route for common FACETS visual surfaces and states what can and cannot be claimed from that visual correspondence. It distinguishes the FACETS-style asterisk ruler from the native uncertainty-aware Wright map.

Usage

```
facets_visual_contract()
```

Value

A data.frame with visual surface, status, first route, editable data route, and claim-boundary columns.

See Also

plot(x, type = "wright", view = "unified"), plot_data(), facets_term_crosswalk(), facets_feature_coverage()

Examples

```
facets_visual_contract()
```

---

facet_quality_dashboard *Facet-quality dashboard for facet-level screening*

Description

Build a compact dashboard for one facet at a time, combining facet severity, misfit, central-tendency screening, and optional bias counts.

Usage

```
facet_quality_dashboard(
  fit,
  diagnostics = NULL,
  facet = NULL,
  bias_results = NULL,
  severity_warn = 1,
  misfit_warn = NULL,
  central_tendency_max = NULL,
  bias_count_warn = 1L,
  bias_abs_t_warn = 2,
  bias_abs_size_warn = 0.5,
  bias_p_max = 0.05
)

## S3 method for class 'mfrm_facet_dashboard'
summary(object, digits = 3, top_n = 10, ...)

## S3 method for class 'mfrm_facet_dashboard'
plot(x, y = NULL, type = c("severity", "flags"), ...)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() for the same fit. Mismatched or outdated diagnostics are rejected. Recompute diagnostics after refitting; the model itself does not need to be fitted again.
facet	Optional facet name. When NULL, the function tries to infer a rater-like facet and otherwise falls back to the first modeled facet.
bias_results	Optional output from estimate_bias() or a named list of such outputs. Non-matching bundles are skipped quietly.
severity_warn	Absolute estimate cutoff used to flag severity outliers.
misfit_warn	NULL (default) uses both bounds returned by mfrm_misfit_thresholds(), including any session options. A numeric value instead uses that upper cutoff and its reciprocal as the lower cutoff; it does not preserve the current lower bound. For example, 1.5 selects about 0.67–1.5, whereas the unmodified package defaults are 0.5–1.5.
central_tendency_max	Legacy opt-in absolute estimate cutoff for marking facet estimates near the fitted origin. The default NULL disables this flag because origin proximity is not evidence that a rater avoids extreme score categories. Use data_quality_report() for observed category-use screening and rater_centrality_table() for model-based indirect centrality indices.
bias_count_warn	Minimum flagged-bias row count required to flag a level.
bias_abs_t_warn	Absolute t cutoff used when deriving bias-row flags from a raw bias bundle.

bias_abs_size_warn
: Absolute bias-size cutoff used when deriving bias-row flags from a raw bias bundle.

bias_p_max
: Probability cutoff used when deriving bias-row flags from a raw bias bundle.

object
: Output from facet_quality_dashboard().

digits
: Number of digits for printed numeric values.

top_n
: Number of flagged levels to preview.

...
: For plot(): Display controls from plot_facet_quality_dashboard().
For summary(): Reserved for generic compatibility.

x
: A result from facet_quality_dashboard().

y
: Ignored; present for generic compatibility.

type
: "severity" or "flags".

Details

The dashboard screens individual facet elements across four complementary criteria:

- **Severity**: elements with $|\text{Estimate}| >$ severity_warn logits are flagged as unusually harsh or lenient.
- **Misfit**: elements with Infit or Outfit MnSq outside the configured heuristic review band are flagged. The band defaults to the package pair returned by mfrm_misfit_thresholds() (Linacre 0.5-1.5); pass misfit_warn = 1.5 to keep the older symmetric $[1/\text{misfit_warn}, \text{misfit_warn}]$ form (0.67-1.5).
- **Reference proximity (legacy CentralTendencyFlag label)**: when central_tendency_max is supplied, elements with $|\text{Estimate}| <$ central_tendency_max logits are marked. This only describes proximity to the fitted origin and must not be interpreted as central-category use or restriction of range. Those are response-pattern questions; see data_quality_report() and rater_centrality_table() for their distinct descriptive indices.
- **Bias**: elements involved in $\geq$ bias_count_warn screen-positive interaction cells (from estimate_bias()) are flagged.

A **flag density** score counts how many enabled criteria each element triggers. Elements flagged on multiple criteria warrant priority review and may motivate training or a documented data-quality review; the dashboard does not justify automatic row, person, or rater exclusion.

Default thresholds are screening heuristics. Prespecify and justify any application-specific alternatives rather than treating them as universal validity or acceptance criteria. MissingMetrics lists unavailable estimate, SE, Infit or Outfit values. A severity or misfit flag is NA when it cannot be evaluated (an observed misfit exceedance still flags even if the other fit index is unavailable). FlagCount counts observed flags only; zero is not a complete pass. IncompleteLevels counts levels with any missing diagnostic. Bias counts describe supplied results only, not tests of absence of bias. Fit-readiness restrictions are retained in fit_readiness, interpretation_status and notes. Review overlap with subset_connectivity_report() and category use with data_quality_report() before interpreting between-rater differences.

summary() method: Summarize a facet-quality dashboard

plot() method: Plot a saved facet-quality dashboard

Value

An object of class `mfrm_facet_dashboard` (also inheriting from `mfrm_bundle` and `list`). The object summarizes one target facet: `overview` reports the facet-level screening totals, `summary` provides aggregate estimates and flag counts, `detail` contains one row per facet level with the computed screening indicators, `ranked` orders levels by review priority, `flagged` keeps only levels requiring follow-up, `bias_sources` records which bias-result bundles contributed to the counts, `settings` stores the resolved thresholds, and `notes` gives short interpretation messages about how to read the dashboard.

For `summary()`: An object of class `summary.mfrm_facet_dashboard`.

For `plot()`: The existing plot payload, including screening cautions.

Output

The returned object is a bundle-like list with class `mfrm_facet_dashboard` and components:

- `facet`: character scalar naming the dashboard's target facet
- `facet_source`: character scalar describing whether the target facet was inferred from the fit configuration or supplied explicitly
- `overview`: one-row structural overview
- `summary`: one-row screening summary
- `detail`: level-level detail table
- `ranked`: detail ordered by flag density / severity
- `flagged`: flagged levels only
- `bias_sources`: per-bundle bias aggregation metadata
- `settings`: resolved threshold settings
- `notes`: short interpretation notes
- `diagnostics`: the `mfrm_diagnostics` bundle the dashboard was built from (echoed for downstream helpers that need to traverse the same diagnostics object)
- `bias_results`: the `mfrm_bias` bundle (or list of bundles) when `bias_results` was supplied; `NULL` otherwise

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model.

See Also

`diagnose_mfrm()`, `estimate_bias()`, plot(x, type = "qc")

`facet_quality_dashboard()`, plot(x, type = "facet_dashboard")

mfrmr_plot_migration

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("facet_quality_dashboard", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
toy <- toy[toy$Person %in% unique(toy$Person)[1:8], ]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
dash <- facet_quality_dashboard(fit, diagnostics = diag)
summary(dash)

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_facet_dashboard", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
summary(facet_quality_dashboard(fit, diagnostics = diag))

## End(Not run)
```

`facet_small_sample_review` *Review rating counts for each facet level*

Description

Reports per-level observation counts, SE, and fit statistics for every level of every facet in a fitted MFRM model, and classifies each level as `"sparse"`, `"marginal"`, `"standard"`, or `"strong"` using configurable count bands. These labels identify levels to inspect; they do not establish measurement precision or sample adequacy.

Usage

```
facet_small_sample_review(
  fit,
  diagnostics = NULL,
  thresholds = c(sparse = 10, marginal = 30, standard = 50)
)

## S3 method for class 'mfrm_facet_sample_review'
plot(
  x,
  top_n = NULL,
  preset = c("standard", "publication", "compact", "monochrome"),
```

```
  ...
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output from the same fit. When supplied, per-level Infit and Outfit are added. Estimate and SE come from the fitted facet tables; unavailable SEs remain missing.
thresholds	Named numeric vector of count bands. Defaults are c(sparse = 10, marginal = 30, standard = 50). Supply exactly these three names with finite, positive values satisfying sparse < marginal < standard; their order in the vector is immaterial. The defaults are package screening choices, not precision thresholds.
x	An mfrm_facet_sample_review object.
top_n	Optional integer; trim the y-axis to the top_n smallest level counts per facet. NULL (default) keeps all.
preset	One of "standard", "publication", "compact", "monochrome".
...	Reserved.

Details

N counts retained rating rows for a level in fit$prep$data, without summing observation weights. For a rater it counts ratings, not distinct examinees; for a person it counts that person's ratings. Repeated or dependent ratings need not supply independent information. Review the counts alongside targeting, connectedness, category use, the estimation method, and available uncertainty estimates before drawing conclusions.

plot() method: Plot a facet sample-size review

Per-level observation counts rendered as a horizontal bar chart coloured by the count band assigned in facet_small_sample_review(). Vertical dashed lines mark the sparse / marginal / standard thresholds so reviewers see where every facet level sits relative to the selected screening thresholds. Counts and colours do not establish measurement precision.

Value

A list of class mfrm_facet_sample_review with:

- table: one row per (Facet, Level) with N, Estimate, SE, Infit, Outfit, and SampleCategory.
- summary: counts of levels in each sample-size category, by facet.
- facet_summary: count range and lowest count band per facet.
- thresholds: the applied count bands.
- notes: count definition and interpretation limits, also shown by print() and summary().

For plot(): Invisibly, the data.frame used for the plot, with the review's interpretation limits in its notes attribute.

Interpreting output

With the default thresholds, the four labels mean only `N < 10`, `10 <= N < 30`, `30 <= N < 50`, and `N >= 50`, respectively. `WorstCategory` is the lowest count band represented within a facet. Neither `"standard"` nor `"strong"` guarantees narrow intervals, correct coverage, good targeting, or suitability for publication. Do not pool or discard levels solely because of a count label.

Linacre (1994) relates Rasch item/person precision to sample size under targeting and model assumptions. Those guidelines do not establish a precision guarantee for these raw many-facet rating counts. Jones and Wind (2018) study repeated ratings in incomplete designs and highlight examinee precision. Integrating over a population distribution in MML does not remove uncertainty in individual person scores. Interpret `SE` according to the fitted estimator; this count review adds no interval qualification or hierarchical model.

Typical workflow

1. Fit with `fit_mfrm()`; optionally also produce `diagnostics` with `diagnose_mfrm()` if you want per-level Infit/Outfit.
2. Call `facet_small_sample_review(fit, diagnostics)`.
3. Read the `facet_summary` first: it highlights the worst level per facet. The `summary` table gives counts in each band.
4. If any facet is flagged as sparse or marginal, discuss it in the Methods section; `build_apa_outputs()` already adds a sentence about the band when `fit$summary$FacetSampleSizeFlag` is set.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

References

Linacre, J. M. (1994). Sample size and item calibration stability. *Rasch Measurement Transactions, 7*(4), 328. https://rasch.org/rmt/rmt74m.htm

Jones, E., & Wind, S. A. (2018). Using repeated ratings to improve measurement precision in incomplete rating designs. *Journal of Applied Measurement, 19*(2), 148-161. PMID: 29894984.

See Also

detect_facet_nesting(), analyze_hierarchical_structure(), compute_facet_icc(), compute_facet_design_eff
reporting_checklist().

facet_small_sample_review().

Examples

```
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
review <- facet_small_sample_review(fit)
summary(review)

# Custom thresholds (e.g. a stricter protocol).
strict <- facet_small_sample_review(
  fit,
  thresholds = c(sparse = 15, marginal = 40, standard = 100)
)
strict$facet_summary
```

---

facet_statistics_report

Build a facet statistics report (preferred alias)

Description

Build a facet statistics report (preferred alias)

Usage

```
facet_statistics_report(
  fit,
  diagnostics = NULL,
  metrics = c("Estimate", "Infit", "Outfit", "SE"),
  ruler_width = 41,
  distribution_basis = c("both", "sample", "population"),
  se_mode = c("both", "model", "fit_adjusted")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() for the same fit. Mismatched or outdated diagnostics are rejected. Recompute diagnostics after refitting; the model itself does not need to be fitted again.
metrics	Numeric columns in diagnostics$measures to summarize.
ruler_width	Width of the fixed-width ruler used for M/S/Q/X marks.
distribution_basis	Which distribution basis to keep in the appended precision summary: "both" (default), "sample", or "population".
se_mode	Which standard-error mode to keep in the appended precision summary: "both" (default), "model", or "fit_adjusted".

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_facet_statistics (type = "means", "sds", "ranges").

Value

A named list with facet-statistics components. Class: mfrm_facet_statistics.

Interpreting output

- facet-level means/SD/ranges of selected metrics (Estimate, fit indices, SE).
- fixed-width ruler rows (M/S/Q/X) for compact profile scanning.

Typical workflow

1. Run facet_statistics_report(fit).
2. Inspect summary/ranges for anomalous facets.
3. Cross-check flagged facets with fit and chi-square diagnostics. The returned bundle now includes:

- precision_summary: facet precision/separation indices by DistributionBasis and SEMode
- variability_tests: fixed/random variability tests by facet
- se_modes: compact list of available SE modes by facet

See Also

diagnose_mfrm(), summary.mfrm_fit(), plot(x, type = "facets_chisq"), mfrmr_reports_and_tables

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("facet_statistics_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
out <- facet_statistics_report(fit)
summary(out)
p_fs <- plot(out, draw = FALSE)
p_fs$data$plot

## End(Not run)
```

---

fair_average_table *Build an adjusted-score reference table bundle*

Description

Build an adjusted-score reference table bundle

Usage

```
fair_average_table(
  fit,
  diagnostics = NULL,
  facets = NULL,
  totalscore = TRUE,
  umean = 0,
  uscale = 1,
  udecimals = 2,
  reference = c("both", "mean", "zero"),
  label_style = c("both", "native", "legacy"),
  omit_unobserved = FALSE,
  xtreme = 0,
  fair_se = FALSE,
  ci_level = 0.95,
  xtreme_method = c("display", "facets"),
  reference_plan = NULL
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() for this fit. Matching saved diagnostics are reusable; recompute them after refitting.

facets	Optional subset of facets.
totalscore	Include all observations for score totals (TRUE) or apply extreme-row exclusion (FALSE). In ordinary additive RSM/PCM JML, finite direct anchors remain eligible even with all-minimum/all-maximum responses, under both xtreme_method conventions.
umean	Additive origin shift for Measure (not fair-score values).
uscale	Multiplicative scale for Measure and measure SEs (not fair scores).
udecimals	Rounding digits used in formatted output.
reference	Which adjusted-score reference to keep in formatted outputs: "both" (default), "mean", or "zero".
label_style	Column-label style for formatted outputs: "both" (default), "native", or "legacy".
omit_unobserved	If TRUE, remove unobserved levels.
xtreme	Nonnegative correction smaller than the score range for all-minimum/all-maximum rows. Default 0 leaves measures unchanged. Its units depend on xtreme_method; no responses or primary estimates are changed, and this does not correct JML bias.
fair_se	Logical. When TRUE and fit is an MML GPCM fit, add structural delta-method standard errors and confidence limits for Fair(M) / AdjustedAverage and Fair(Z) / StandardizedAdjustedAverage. Person rows remain NA because MML person EAP estimates are not part of the structural Hessian. For RSM, PCM, and JML fits this option leaves fair-average SE columns unavailable.
ci_level	Confidence level used when fair_se = TRUE; default 0.95.
xtreme_method	"display" (default) retains the historical average-score display adjustment. "facets" uses conditional total-score extrapolation for ordinary additive RSM/PCM JML, without group-mean anchors. Supply xtreme = .3 explicitly to use the usual FACETS main correction.
reference_plan	Optional data frame of non-Person facet columns, using the fitted facet names, and an optional positive Weight column. For ordinary additive RSM/PCM JML, adds reference_scores: each Person's expected average on this same plan, using its fitted facet locations and scale profiles. Weights are normalized to sum to one. This distinct target does not replace FairM/FairZ or their administered PCM scale mixture. Primary estimates are used, irrespective of display/extrapolation settings. Scores use internal score units, unaffected by umean or uscale. The result retains the fit for confint.mfrm_fair_average(). For an explicit mfrm_jeffreys_fit, this argument is required and the result contains only common-plan reference_scores, the plan, fit and estimator identity. Other arguments beyond fit and reference_plan are refused: ordinary FairM/FairZ diagnostics do not apply. Inspect summary(result); use plot(confint(result, ...)) for supported interval plots. Direct FairM/FairZ plotting is unavailable for this result.

Details

This function wraps the package's adjusted-score calculations and returns both facet-wise and stacked tables. Historical display columns such as Fair(M) Average and Fair(Z) Average are re-

tained for compatibility, and package-native aliases such as `AdjustedAverage`, `StandardizedAdjustedAverage`, `ModelBasedSE`, and `FitAdjustedSE` are appended to the formatted outputs.

`RSM` fair scores use the shared scale. `PCM` retains each observation's step profile, converts the reference predictor to an expected score, then averages with rating weights. Averaging thresholds before conversion is not equivalent. FairM replaces other element locations by their means; FairZ replaces them by zero. The PCM scale exposure is retained, so fair scores can still depend on which scales were administered. The Person reference excludes nonfinite primary estimates and records eligible/excluded counts. With no finite Person reference, non-Person FairM is unavailable. FairZ does not need a Person mean. Neither target integrates over Person ability; these are expected scores, not z-scores.

With `xtreme > 0`, `PrimaryMeasure` retains the fitted measure, including infinite JML estimates and fixed anchors, on the requested reporting scale. Fixed anchors retain their measure in both columns. With `xtreme_method = "display"`, an expected average that is `xtreme` from the endpoint is inverted. Measure SEs are unavailable on replaced rows and fair-score calculations do not use that replacement. For ordinary additive RSM/PCM JML, a directly anchored element is not classified as an unanchored extreme for row exclusion. Its observations remain eligible when `totalscore = FALSE`, unless another unanchored extreme element requires their exclusion. This rule also preserves the PCM scale mixture used in fair scores and does not depend on the display adjustment. Recreate older affected fair-score tables and plots from the saved fit and matching diagnostics; refitting the calibration is unnecessary.

With `xtreme_method = "facets"`, the weighted total is moved inward by `xtreme` times the smallest positive weight among that element's eligible observations. Responses involving another unanchored extreme element are excluded. The resulting weighted score equation is solved using the actual administered calibration and PCM scales. Its reciprocal square-root conditional information supplies the reporting SE. The replacement enters that element's FairM/FairZ, while reference locations retain their original basis. `ExtremeAdjustment` records a weighted-total correction in this mode, versus an average-score correction in `"display"` mode. `MeasureBasis` and `MeasureSEBasis` identify the reported quantities. These are finite reporting values and conditional SEs at an adjusted total, not finite MLEs or SEs for the original extreme response pattern. Fair-score intervals remain unavailable when `xtreme > 0`.

Binary unit/replication-weight examples and a fixed-scale PCM kernel were checked against FACETS 4.5.0. This is not whole-program equivalence: partial anchor examples can leave native score residuals and shifted final measures, which this conditional calculation does not reproduce. Group-mean anchors, fitted interactions, corrected JML, MML and GPCM are outside this route. Recompute older diagnostics and recreate saved tables from the existing fit before summarizing or plotting them; no model refit is needed. With an extreme fixed Person plus an informative interior Person anchor, matched RSM/PCM examples also agree for conditional fair scores at the reported native calibration. Main estimates can still differ and native iteration limits or nonzero free-element score residuals must not be interpreted as calibration agreement.

`GPCM` fits are supported under a slope-aware element-conditional construction. For each slope-facet element $j^\star$ the per-row fair-average is the GPCM expected score

$$\mathrm{FA}_{p,j^\star} = \sum_k k \cdot P_{GPCM}(X = k \mid \theta_p, a_{j^\star}, \boldsymbol{\delta}_{j^\star})$$

computed at that element's own discrimination $a_{j^\star}$ and threshold structure. Rows for non-slope facets (Person, Rater, ...) use the geometric-mean-one slope by the GPCM identification convention and average the threshold profiles before score conversion. At unit slope the conversion

reduces to the PCM kernel, but this aggregation need not equal PCM's average of administered-scale expectations for non-step-facet rows. This is an identification-based reporting convention for the package's GPCM route, not a unique free-discrimination score-side analogue to FACETS fair averages. Do not report it as FACETS score-side equivalence or as an operational scoring rule unless that convention is substantively justified.

Standard errors on the fair-average value itself are opt-in for MML GPCM fits via fair_se = TRUE. The Model S.E., ModelBasedSE, Real S.E., and FitAdjustedSE columns retain the same meaning as for PCM (scaled facet-measure SEs); fair-average uncertainty is reported under distinct columns such as Fair(M) S.E., Fair(M) CI Lower, and AdjustedAverageSE.

Value

A named list with:

- by_facet: named list of formatted data.frames
- stacked: one stacked data.frame across facets
- raw_by_facet: unformatted component tables
- settings: resolved options

Interpreting output

- stacked: cross-facet table for global comparison.
- by_facet: per-facet formatted tables for reporting.
- raw_by_facet: unformatted values for custom analyses/plots; identifiers use the column Level.
- settings: scoring-transformation and filtering options used.

Observed-vs-fair gaps also reflect person mix and assignment. They are descriptive follow-up prompts, not standalone evidence of rater bias. Ordinary additive RSM/PCM JML tables also retain PTMEAExpected and its availability metadata when present in the matching diagnostics. This is a conditional reference for observed PtMea Corr, not an inferential cutoff; see diagnose_mfrm() for its finite-count correction and weighting basis.

Typical workflow

1. Run fair_average_table(fit, ...).
2. Inspect summary(t12) and t12$stacked.
3. Visualize with plot(x, type = "fair_average").

Output columns

The stacked data.frame contains the following columns, selected by reference, label_style, and fair_se:

Facet Facet name for this row.

Element Element label within the facet.

Obsvd Average Observed raw-score average.

Fair(M) Average Model-adjusted reference average on the reported score scale.

Fair(Z) Average Expected score at a zero reference environment, not a z-score.

ObservedAverage, AdjustedAverage, StandardizedAdjustedAverage Package-native aliases for the three average columns above.

AdjustedAverageSE, AdjustedAverageCI_Lower, AdjustedAverageCI_Upper Optional structural delta-method uncertainty for `AdjustedAverage` when `fair_se = TRUE` and available.

StandardizedAdjustedAverageSE, StandardizedAdjustedAverageCI_Lower, StandardizedAdjustedAverageCI_Uppe Optional structural delta-method uncertainty for `StandardizedAdjustedAverage` when `fair_se = TRUE` and available.

Measure Displayed facet measure, transformed by `umean` and `uscale`; may be replaced when `xtreme > 0`.

PrimaryMeasure Original fitted measure on the same reporting scale, including infinite JML estimates.

MeasureBasis, ExtremeAdjustment Whether the displayed measure was replaced and the replacement amount in score units.

MeasureSEBasis In `raw_by_facet`, distinguishes conditional information at a corrected total from fitted-measure uncertainty.

FairMReference The mean reference used for FairM, or why that reference is unavailable.

ModelBasedSE, FitAdjustedSE Package-native aliases for `Model S.E.` and `Real S.E.`.

Infit MnSq, Outfit MnSq Fit statistics for this level.

Standard-error caveat (read before quoting CIs)

The `Model S.E.`, `ModelBasedSE`, `Real S.E.`, and `FitAdjustedSE` columns in this table are the **measure-level** standard errors of the underlying facet element, rescaled by `abs(uscale)` to the reported Measure units. Fair scores remain on the fitted internal score scale. They are **not** delta-method standard errors of the fair-average values themselves. When `fair_se = TRUE`, the distinct `Fair(M) S.E.` / `Fair(Z) S.E.` columns are computed by propagating the joint covariance of the relevant facet element, the threshold parameters, and the slope parameters through the gradient of $\mathrm{E}[X \mid \theta_p, j^\star]$. This is a structural covariance calculation: MML person EAP estimates are conditioned on rather than included in the Hessian, so person rows receive unavailable fair-average SEs. **Do not use the measure-level** `ModelBasedSE` **/** `Model S.E.` **columns as** $\pm 1.96 \cdot \mathrm{SE}$ **confidence-interval bounds on the fair-average value.** `FairCIEligible` is `FALSE` for these diagnostic intervals; numerical availability (`ok` or `regularized`) does not establish inferential validity. `FairCIReportingUse` distinguishes diagnostic-only and unavailable rows. Full-refit coverage remains unverified. For RSM/PCM, this table does not supply fair-score SEs; plot(x, type = "fair_average") can compute a conditional interval from a fitted model by propagating only the focal measure SE. Summaries identify `FairMetric`: FairM for mean/both reference tables, FairZ for zero-reference tables, with matching score/SE column names. Export `stacked` or the summary's `summary` / `preview` data.frames with `utils::write.csv()`. `export_summary_appendix()` does not accept this bundle.

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press.

- Linacre, J. M. (1994). *Many-facet Rasch Measurement* (2nd ed.). MESA Press.
- Linacre, J. M. (2026). *A user's guide to FACETS, version 4.5.0.* Winsteps.com. (FACETS Table 7 corresponds to the fair-average construction implemented here for `RSM` / `PCM` fits; the slope-aware element-conditional construction for `GPCM` is documented in this help page.)
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573. doi:10.1007/BF02293814
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174. doi:10.1007/BF02296272
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176. (Cited for the `GPCM` slope-aware extension.)

See Also

`diagnose_mfrm()`, `unexpected_response_table()`, `displacement_table()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("fair_average_table", run.dontrun = TRUE)
## Not run:
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Use the diagnostics saved with this fit
diagnostics <- example_fit$diagnostics

# Compare observed person means with model-based scores on a common reference
fair <- fair_average_table(fit, diagnostics = diagnostics, facets = "Person",
                           reference = "mean", label_style = "native")
head(fair$raw_by_facet$Person[, c("Level", "ObservedAverage", "FairM")])
# FairM uses the mean reference for the other facets; it is in score units
# It is a conditional model prediction, not a guarantee of fairness

# Fit the same synthetic ratings by JML to illustrate total-score extrapolation
toy <- load_mfrmr_data(example_fit$dataset)
jml <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML")
extreme_report <- fair_average_table(jml, xtreme = .3, xtreme_method = "facets")
head(extreme_report$raw_by_facet$Person[, c("Level", "PrimaryMeasure",
     "Measure", "MeasureBasis", "ModelSE", "MeasureSEBasis")])
# ModelSE describes the adjusted total when MeasureBasis says extrapolation

## End(Not run)
```

`fit_measures_table` *Build a FACETS-style fit-measures review table*

Description

Build a FACETS-style fit-measures review table

Usage

```
fit_measures_table(
  x,
  diagnostics = NULL,
  facet = NULL,
  include_person = FALSE,
  lower = NULL,
  upper = NULL,
  zstd_cut = 2,
  ci_level = 0.95,
  threshold_profiles = c("literature", "active", "all", "none"),
  fit_df_method = "facets",
  df_zstd_tolerance = 0.05,
  df_zstd_large_shift = 0.5,
  df_ratio_tolerance = 0.05,
  sort_by = c("status", "abs_zstd", "facet", "level"),
  top_n = Inf,
  flag_basis = c("mnsq", "mnsq_or_zstd")
)
```

Arguments

`x`	Output from `fit_mfrm()` or `diagnose_mfrm()`.
`diagnostics`	Optional diagnostics object. If supplied, `x` may be the matching supported fitted object; mismatched or outdated diagnostics are rejected. Alternatively, pass diagnostics alone as `x`. Supplying diagnostics does not enable ordinary fit measures for corrected JML, two-family GPCM, testlet or shared-rater models. Use `mfrm_response_diagnostics()` for their separately supported descriptive response residuals.
`facet`	Optional facet-name filter, for example `"Rater"`.
`include_person`	Logical; if `FALSE` (default), excludes the `Person` facet so operational facet elements are shown first.
`lower, upper`	Optional mean-square review band. Defaults to `mfrm_misfit_thresholds()`.
`zstd_cut`	Absolute ZSTD cutoff used for directional underfit/overfit flags when `flag_basis = "mnsq_or_zstd"`; also used for the separate ZSTD and df-sensitivity columns. Default `2`.

`ci_level` Confidence level used to add approximate Wald intervals for facet measures. Default `0.95`.

`threshold_profiles` Which mean-square threshold profiles to summarize in addition to the active table band. `"literature"` (default) returns commonly cited bands from Linacre, Bond & Fox, and Wright & Linacre; `"active"` returns only the active band; `"all"` returns both; `"none"` suppresses profile summaries.

`fit_df_method` Degrees-of-freedom convention used when `diagnostics` is computed inside the helper. `"facets"` is the default, matching `diagnose_mfrm()`. `"engine"` keeps the former package-native fit df, `"facets"` makes primary ZSTD columns use the FACETS/Wright-Masters fourth-moment df convention, and `"both"` keeps engine columns primary while adding FACETS-style companion df/ZSTD columns for comparison.

`df_zstd_tolerance` Smallest absolute engine-vs-FACETS-style ZSTD difference treated as interpretively visible rather than rounding noise in `df_sensitivity`. Default `0.05`.

`df_zstd_large_shift` Absolute engine-vs-FACETS-style ZSTD difference labeled `large_zstd_shift` when the `zstd_cut` flag status is unchanged. Default `0.5`.

`df_ratio_tolerance` Relative df-difference threshold used to label `df_convention_difference`; for example, `0.05` means a 5 percent engine-vs-FACETS-style df difference. Default `0.05`.

`sort_by` Sorting rule: `"status"` prioritizes underfit/overfit rows, `"abs_zstd"` sorts by largest absolute ZSTD, and `"facet"` / `"level"` sort alphabetically.

`top_n` Optional maximum number of rows in the returned main table.

`flag_basis` `"mnsq"` (default) bases directional status on Infit/Outfit mean squares. `"mnsq_or_zstd"` also flags either large positive or negative ZSTD, reproducing the previous combined rule when all indices are available.

Details

This helper gives users a direct table route for the common FACETS-style question: which raters, criteria, or other facet elements show underfit or overfit? It uses the fit statistics already computed by `diagnose_mfrm()`. `Fixed` identifies values supplied by anchors or identification constraints. Their sampling SEs and intervals are not applicable; response-fit statistics remain available. Recompute older diagnostics to obtain these labels. The table also retains `SE_Method`, `PrecisionTier`, `SEUse`, `CIBasis`, `CIUse`, `CIEligible`, `CILabel`, `CI_Method` and `SupportsFormalInference` from the diagnostics. For JML, normal bands remain exploratory screening summaries, not qualified confidence intervals for the true parameter. Changing `ci_level` changes their width, not their inferential status. Older inputs without this metadata are labelled as having an unrecorded basis; finite SEs alone do not establish formal inference.

After `fm <- fit_measures_table(x)`, use `p <- plot(fm, type = "measure_ci", draw = FALSE)` to inspect the bands. `plot_data(p, "caption")` retains the explanation and `plot_data(p, "table")` retains the interpretation fields. Titles and notes are not drawn by default; `show_notes = TRUE` explicitly adds the note. Fixed values use open diamonds without intervals; other finite estimates without intervals use crosses.

Directional labels use the selected `flag_basis`. By default, high mean squares are labeled `underfit` and low mean squares `overfit`; conflicting directions are `mixed`. ZSTD values remain visible, and `ZSTDOnly` identifies combined-rule flags when the mean-square screen is known negative. These standardized values are sample-size and df-convention dependent. A ZSTD-only departure is not automatically a substantively important misfit.

Missing indices cannot establish a negative screen: one positive is retained, but no positive plus an unavailable index is `not_available`. `ScreenComplete` records whether all indices required by the selected rule were available. Low mean squares indicate less residual variability than expected. They do not diagnose misconduct, poor rater quality or machine-learning overfitting, and are not an automatic reason for exclusion. Investigate high mean squares and substantive consequences first. Published bands are contextual heuristics, not calibrated error probabilities or universal thresholds for all facets.

FACETS-style ZSTD comparison is controlled by `fit_df_method`. MnSq values should be compared first; df and ZSTD columns explain how the same MnSq values are standardized. Use `fit_df_method = "both"` when preparing a table for FACETS users or when explaining why |ZSTD| flags change across df conventions. The `df_zstd_tolerance`, `df_zstd_large_shift`, and `df_ratio_tolerance` arguments make the df-sensitivity screen explicit so the same table can be reproduced under stricter or more permissive review rules.

Saved calculation definitions remain beside the fit values: `FitBasis` and `FitInterpretation` distinguish residuals at saved EAPs, JML optimizer coordinates, and imported source statistics. EAP plug-in MnSq is descriptive; it is not posterior-averaged or posterior-predictive fit, and same-response EAP scoring does not guarantee expectation one. Missing definitions are marked as not recorded, without discarding finite values. `diagnostic_basis` preserves the original guide to other diagnostic paths; marginal category counts are not posterior-averaged MnSq.

Value

A bundle of class `mfrm_fit_measures` with:

- `fit_basis`: calculation target and interpretation by facet
- `diagnostic_basis`: saved diagnostic-path guide, when present
- `table`: R-friendly fit-measure table with status columns
- `facets_table`: FACETS-style column labels for reporting/review
- `status_summary`: counts by facet and fit status
- `profile_summary_by_facet`: underfit/overfit rates for each threshold profile and facet
- `profile_summary_overall`: threshold-profile rates pooled over facets
- `df_sensitivity`: row-level engine-vs-FACETS-style df/ZSTD comparison
- `df_sensitive`: subset of rows where df convention changes the ZSTD flag or materially changes ZSTD interpretation
- `df_sensitivity_summary`: counts of df-sensitive rows
- `underfit`, `overfit`, `mixed`: filtered row subsets
- `df_conversion_guide`: FACETS-style df/ZSTD comparison guide
- `settings`: thresholds and filters used

References

Linacre, J. M. (2003). Size vs. significance: standardized chi-square fit statistic. *Rasch Measurement Transactions*, 17(1), 918. https://www.rasch.org/rmt/rmt171n.htm. Wright, B. D. and Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions*, 8(3), 370. https://www.rasch.org/rmt/rmt83b.htm.

See Also

diagnose_mfrm(), facets_fit_review(), plot(x, type = "bubble"), mfrm_misfit_thresholds(), mfrm_screening_sensitivity()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("fit_measures_table", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 7, maxit = 30
)
fm <- fit_measures_table(fit, facet = "Rater")
fm$facets_table
fm$underfit

# Include FACETS-style df/ZSTD companion columns for comparison.
fm_facets <- fit_measures_table(fit, facet = "Rater", fit_df_method = "both")
fm_facets$df_conversion_guide$decision_guide

## End(Not run)
```

`fit_mfrm` *Fit many-facet ordered-response models with a flexible number of facets*

Description

Estimate effects on ordered ratings, such as rater severity and criterion difficulty, and Person abilities where the estimator supplies them. A facet is one such source of variation; its levels are the individual raters or criteria. Each data row is one rating. Supply its column names with `person`, `facets`, and `score`, as in the complete example below. The default is `method = "MML"` (marginal maximum likelihood). The `RSM` / `PCM` branches are the package's many-facet Rasch-family reference route. `GPCM` adds positive, level-specific discriminations to one facet, or provisionally to two facets under MML. With one family, MML permits a different facet to supply category steps; JML requires the same facet for both roles. Start with "Choose a model and its population" below; the model-specific sections explain uncertainty and comparisons. In the example, `ratings` stores the data and `fit` stores the fitted model. The loading line reads an existing packaged synthetic dataset; you do not need to generate ratings or create an empty object

first. Quoted column names such as "Person" must match the data, including case. For your own CSV, see the "Use your own CSV" section of vignette("mfrmr-workflow", package = "mfrmr"). If the vignette is not installed, mfrmr_workflow_methods and describe_mfrm_data() explain the input checks. Pass the reviewed rating data frame to data, not the object returned by describe_mfrm_data(). Before adapting the example, read "Check defaults before adapting an example" in mfrmr_workflow_methods. The score ladder, missing-row handling and population distribution are analysis choices even when arguments are omitted.

Usage

```
fit_mfrm(
  data,
  person,
  facets,
  score,
  rating_min = NULL,
  rating_max = NULL,
  weight = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  model = c("RSM", "PCM", "GPCM"),
  method = c("MML", "JML", "JMLE", "CML"),
  step_facet = NULL,
  slope_facet = NULL,
  facet_interactions = NULL,
  min_obs_per_interaction = 10,
  interaction_policy = c("warn", "error", "silent"),
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  positive_facets = NULL,
  anchor_policy = c("warn", "error", "silent"),
  min_common_anchors = 5L,
  min_obs_per_element = 30,
  min_obs_per_category = 10,
  quad_points = 31,
  maxit = 400,
  reltol = 1e-09,
  optimizer = c("auto", "BFGS", "L-BFGS-B"),
  mml_engine = c("direct", "em", "hybrid"),
  population_formula = NULL,
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
  facet_shrinkage = c("none", "empirical_bayes", "laplace"),
  facet_prior_sd = NULL,
  shrink_person = FALSE,
  attach_diagnostics = FALSE,
```

```
  checkpoint = NULL,
  gpcm_mml_identification = c("free_population", "fixed_standard_normal"),
  mml_integration = c("fixed", "adaptive", "auto"),
  category_policy = NULL,
  em_score_tol = NULL,
  jml_correction_order = NULL,
  jml_correction_sampling = c("fixed_rosters", "random_rosters"),
  gpcm_mml_start = NULL,
  repeated_ratings = c("review", "independent"),
  centering_scope = c("facet", "remainder", "anchor_origin"),
  fit_scope = c("full", "comparisons"),
  comparison_review = NULL,
  mml_accuracy_control = NULL,
  population_sd = NULL,
  jml_penalty = c("none", "jeffreys"),
  progress = FALSE,
  mml_starts = NULL
)

## S3 method for class 'mfrm_cml_fit'
summary(object, ...)

## S3 method for class 'mfrm_cml_fit'
print(x, ...)

## S3 method for class 'mfrm_comparison_fit'
print(x, ...)

## S3 method for class 'mfrm_comparison_fit'
summary(object, ...)
```

Arguments

data	A data.frame in long format with one row per observed rating event. Grouped or rowwise tibbles are analyzed as one table; dplyr grouping metadata is ignored. The named model columns define the design.
person	Column name for the person (character scalar).
facets	Character vector of facet column names. Facets cannot be named `Person`, `Score`, `Weight`, `score_k`, `theta`, `steps`, `interactions`, `log_slopes`, `beta` or `log_sigma2`, which are reserved data/parameter names. Rename such facet columns before analysis. Similar names such as `beta_group` are supported and matched exactly. Data-column names remain available for their intended roles, such as `person = "Person"`, `score = "Score"` and `weight = "Weight"`.
score	Column name for the observed ordered category score. Values must be coercible to real integer category codes within R's integer range. Shared preparation rounds values within `1e-6` of an integer to the nearest category, recording normalization in `fit$prep$preparation_notes`; other fractional values

are rejected. Corrected JML and two-family GPCM require exact integer inputs; normalize roundoff explicitly before those specialized fits. Binary `0/1` or `1/2` responses are supported as the ordered two-category special case. When `keep_original = FALSE`, unused intermediate categories are collapsed to a contiguous internal scale and the mapping is recorded in `fit$prep$score_map`. If `rating_min` / `rating_max` are supplied and the observed scores are a contiguous subset of that range (for example a 1-5 scale with only 2-5 observed), the supplied full range is retained so zero-count boundary categories remain visible in the data-support review. A zero-count boundary category is retained as review evidence. With `keep_original = TRUE`, however, an unobserved internal category in a polytomous fitted ladder creates an unsupported adjacent-step contrast and fitting stops before optimization with a structured category-support error. The explicit Jeffreys route below retains empty categories under a different estimating equation; it does not certify ordinary finite MLEs.

`rating_min` Optional minimum category value. Supply this with `rating_max` when the intended score scale includes unobserved boundary categories.

`rating_max` Optional maximum category value. Supply this with `rating_min` when the intended score scale includes unobserved boundary categories.

`weight` Optional weight column name. Numeric weights retain their supplied precision; character/factor weights are parsed from their labels. Retained weights must have a finite sum and are not automatically rescaled.

`keep_original` Logical. `FALSE` (the current default) collapses non-consecutive observed categories to a contiguous internal scale and records the mapping in `fit$prep$score_map` (the downstream Count = 0 rows are consequently absent). `TRUE` preserves the declared scale so unused intermediate categories remain visible in `rating_scale_table()` and APA outputs, which is recommended for preserving the rubric. This changes which category steps are fitted, not just their displayed labels. Fitting stops if a retained internal category has no observations; reviewing or revising that ladder is a substantive decision, not a formatting option. Retained for compatibility; `category_policy = "preserve"` or `"collapse"` makes the choice explicit in new code.

`missing_codes` Optional pre-processing step that converts sentinel missing-code values to `NA` before any downstream logic. One of:

- `NULL` (default): no recoding; strictly backward-compatible.
- `TRUE` or `"default"`: FACETS / SPSS / SAS convention set (`"99"`, `"999"`, `"-1"`, `"N"`, `"NA"`, `"n/a"`, `"."`, `""`) on the score column only. Person and facet identifiers are preserved because short codes such as `"N"` can be legitimate labels.
- Character vector: an explicit code set, e.g. `c("99", "999", ".a")`, applied across the person, facet, and score columns.

Replacement counts are recorded in `fit$prep$missing_recoding` and surfaced by `build_mfrm_manifest()`. Equivalent to calling `recode_missing_codes()` manually before the fit.

`model` `"RSM"` (default), `"PCM"`, or `"GPCM"`.

`method` `"MML"` (default), `"JML"`, or `"CML"`. `"JMLE"` is accepted as a backward-compatible alias for the same joint-maximum-likelihood path. Native CML supports RSM/PCM

with preserved categories and unit weights; see "Conditional maximum likelihood" for its distinct result class.

`step_facet` Facet whose levels receive separate step parameters in `PCM` and `GPCM`. Supply exactly one facet name for a final analysis. If it is omitted for `PCM`, mfrmr uses a unique item-like facet name (for example, `Item`, `Task`, or `Criterion`) when available; otherwise it retains the first-facet fallback with a warning. `GPCM` always requires an explicit value. CML also requires an explicit PCM step owner. This argument is not used by `RSM`, which has one shared set of rating-scale thresholds.

`slope_facet` Slope facet for the `GPCM` branch. mfrmr estimates one positive slope for every level of this designated facet. Thus `slope_facet = "Criterion"` gives criterion-specific slopes, whereas `slope_facet = "Rater"` gives rater-specific slopes. For one slope family, MML allows a distinct `step_facet`; JML requires `slope_facet == step_facet`. Criterion and rater slope blocks can also be estimated together with an ordered pair of column names; see "Two slope families" below for the explicit MML contract. With one family, slopes are identified on the log scale with their geometric mean fixed to 1, so the table reports relative discrimination across the selected facet's levels rather than unrelated absolute weights.

`facet_interactions`
Optional confirmatory two-way interaction terms between non-person facets, supplied as explicit character terms such as `"Rater:Criterion"` or as a list of length-two character vectors. These interactions are estimated simultaneously as fixed effects in `RSM` and `PCM` fits. Person-involving interactions, higher-order interactions, and random-effect interaction terms are outside the current scope.

`min_obs_per_interaction`
Minimum weighted observations recommended for each interaction cell. Cells below this value are flagged in `interaction_effect_table()` and handled according to `interaction_policy`.

`interaction_policy`
How to handle sparse interaction cells: `"warn"` (default), `"error"`, or `"silent"`.

`anchors` Optional table with `Facet`, `Level`, and `Anchor` columns. Each retained row fixes a location in internal logit coordinates; it is not merely a declaration that an element is common. Person anchors require JML; MML rejects them. See **Anchors and scale constraints** below.

`group_anchors` Optional group-mean constraint table with facet, level, group, and target-value columns. Its use requires a defensible external assumption about the group target; it does not create observed overlap. Use columns `Facet`, `Level`, `Group`, and `GroupValue`. Person group constraints require JML.

`noncenter_facet`
One facet to leave non-centered (default `"Person"`). Other facets use the convention selected by `centering_scope`. To estimate a cohort mean while leaving an anchored RSM/PCM facet uncentered, also set `population_formula = ~ 1` and `population_sd`; see **Anchor origin and cohort means**.

`dummy_facets` Facets to fix at zero.

`positive_facets`
Facets with positive orientation.

`anchor_policy` How to handle anchor-review issues: `"warn"` (default), `"error"`, or `"silent"`.

`min_common_anchors`
Minimum directly anchored levels per non-Person facet used in the package's local count recommendation. This does not verify cross-run element identity, invariance, or empirical connectedness.

`min_obs_per_element`
Minimum weighted observations per facet level used in anchor-review recommendations.

`min_obs_per_category`
Minimum weighted observations per score category used in anchor-review recommendations.

`quad_points` Integer number of Gauss-Hermite quadrature points used for MML integration over the person distribution. The default is `31`. Useful accuracy/runtime settings are:

7	lightweight screening run; information-criterion deltas, weights, preferences, and LRT are disabled. Helpers such as pre
15	intermediate review run when runtime matters; automatic model ranking remains disabled.
31	package default and a starting grid, not evidence by itself that numerical integration is adequate.
61+	user-selected denser grids for same-data sensitivity review; no single order is adequate for every response pattern.

Quadrature adequacy depends on the fitted distribution and score support. Orders whose weights cannot all be represented as positive finite doubles produce an error; increasing the order indefinitely is not supported. Use `mml_quadrature_sensitivity()` to compare the same model and data on user-selected grids before portable calibration and whenever numerical movement could affect a consequential result. The helper reports continuous differences without choosing a cutoff. Raw AIC/BIC/SABIC remain visible below 31 points for diagnosis, but `compare_mfrm()` fails closed rather than turning a screening/review grid into automatic selection.

`maxit` Computational ceiling on optimizer iterations. The default is `400`; for two-family EM this counts outer EM iterations. Two-family adaptive initialization also allows this many EM seed iterations; it then applies separately to every direct optimizer stage at each start. For corrected JML it applies separately to each recorded root-solving stage. This is not a convergence criterion or a model-selection control: a fit that reaches the ceiling remains non-ready until the common convergence and terminal-gradient checks pass. Smaller values used in brief examples are for demonstration only and should not be copied into a substantive analysis without an explicit computational protocol.

`reltol` Portable tolerance setting for the initial optimizer stage. For two-family fixed-grid EM, omit this argument and use `em_score_tol`. Adaptive two-family direct MML uses `reltol` and the direct optimizer's gradient check. New two-family adaptive fits cap the ordinary total-gradient review tolerance at `N * 1e-6`, where N is the number of fitted Persons; a stricter ordinary tolerance is retained. This aligns optimizer finishing with the subsequent per-Person score check. The rule is recorded in the fitted controls; older saved fits retain their recorded review convention. Curvature, derivative and integration checks remain separate. The default is `1e-9`. For BFGS this is passed as `reltol`; for L-BFGS-B it is mapped to `factr` and `pgtol`, whose actual values are recorded

in the fit. When this setting is at least as strict as the public default (`reltol <= 1e-9`), optimizer code zero followed by a failed common terminal-gradient review triggers a bounded warm-started polish ladder. The best non-worsening stage under the recorded selection rule is retained. Requested and selected-stage settings remain in `fit$summary`, and the complete stage history remains in `fit$opt$optimizer_polish`. Ordinary additive RSM/PCM JML without shrinkage can finish a remaining stall with at most three sparse Newton steps from the exact likelihood information. Each step must reduce the gradient without likelihood loss beyond roundoff, pass its linear-solve check and move no coordinate by more than one logit. Extreme Persons, nonpositive information and failed proposals retain their diagnostic state. The final stage is labelled `jml_information_polish`, with method `Newton`, `InformationSteps` and `InformationSolveResidual`. This does not correct JML bias or qualify interval coverage; the original terminal-gradient criterion still applies. If ordinary polishing still stalls, fixed-grid RSM/PCM/GPCM MML fits (including estimated population parameters) or adaptive GPCM MML fits with a fixed population can use one local curvature step to restart the selected optimizer, provided there are at most 64 free parameters. The step requires positive-definite, well-conditioned curvature, a smaller gradient and an objective that does not worsen beyond floating-point roundoff. The original convergence and terminal-gradient criteria still apply; failed proposals retain their reasons in the stage history. Fixed-grid GPCM MML fits with at most 64 free parameters also check numerical curvature after optimizer code zero. Negative curvature can trigger up to three BFGS restarts in rescaled search coordinates, even when the raw gradient is small. Each restart uses the requested `maxit` ceiling. A replacement must pass the original gradient/convergence checks, have no detected negative curvature and not worsen the objective beyond roundoff. Failed recovery retains the estimate with a numerical warning. For fixed-population, fixed-grid RSM/PCM interaction fits, a remaining gradient stall also permits one L-BFGS-B polish after subtracting a constant from the objective (`centered_objective_polish`). This removes the large likelihood constant from relative stopping, without changing derivatives or precision requirements. Stored likelihoods retain their original scale; a resolved likelihood loss is never accepted. This changes only the search coordinates, not the model or information matrix used for inference. It does not establish a global optimum, adequate quadrature, or valid confidence intervals. Inspect the `SmallestCurvature`, `CurvatureScale` and `CurvatureReviewError` fields in the stage history alongside the objective and terminal gradient.

`optimizer` Direct-optimization method. `"auto"` (default) uses the BFGS M step for the two-family EM route. For other routes it uses the limited-memory `"L-BFGS-B"` method for MML and for larger JML parameter vectors (at least 200 free parameters), while retaining BFGS for smaller JML fits. Use `"BFGS"` or `"L-BFGS-B"` to request one method explicitly. The method actually used is recorded in `fit$summary$OptimizerMethod`. For L-BFGS-B, inspect `OptimizerFactr` and `OptimizerPgtol` rather than interpreting `EffectiveReltol` as a native `stats::optim()` control.

`mml_engine` MML optimization engine for `method = "MML"`: `"direct"` (default) uses the selected direct optimizer on the marginal log-likelihood, `"em"` uses an EM loop for

RSM / PCM with population = NULL, including model-estimated facet_interactions, or the explicitly scoped two-family GPCM route described below. "hybrid" uses EM as a warm start before the direct optimizer. Ordinary EM/hybrid requires fixed quadrature. Numerical M steps implement generalized EM, checking both the auxiliary and marginal log likelihood for non-decrease beyond roundoff before accepting an update; failed checks stop without replacing a valid checkpoint. This is not a block ECM algorithm or a guarantee of global convergence or faster fitting. Use mml_starts to review specified starts and keep integration sensitivity separate. For a raw EM fit with a large terminal gradient, use "hybrid" or "direct" and review the resulting readiness. Unsupported one-family combinations fall back to "direct" and record that fallback in fit$summary. Direct, hybrid, and EM engines all require the common terminal-gradient check for the Numerical component of fit readiness; EM relative log-likelihood convergence alone does not establish numerical readiness. InferenceReady is TRUE only when every stored fit-readiness component passes.

population_formula

Optional one-sided formula for a person-level latent-regression population model, for example ~ grade + ses. Latent regression is implemented only for method = "MML" with a unidimensional conditional-normal population model. With NULL, RSM/PCM MML uses a fixed $N(0, 1)$ basis; default GPCM MML instead estimates an intercept-only normal population. Version 0.2.4 does not accept arbitrary fixed normal means or standard deviations: portable RSM/PCM calibration and its anchors are defined on the standard-normal basis, and a silent change of basis would change the meaning of those stored values. A formula-compatible terms object returned in attr(simulated_data, "mfrm_population_data")$population_formula is also accepted, preserving the simulation template's transformation basis, factor levels and contrasts when refitting generated data.

person_data

Optional one-row-per-person data.frame holding background variables for population_formula. Numeric, logical, factor, ordered factor, and character predictors are expanded through stats::model.matrix(); categorical xlevels and contrasts are stored for replay and scoring. Prediction-aware transformations such as scale(), poly() and splines::ns() retain their training basis when scoring new persons. Required when population_formula is supplied.

person_id

Optional person-ID column in person_data. Defaults to person when that column exists in person_data.

population_policy

How missing background data are handled for a latent-regression fit. "error" (default) requires complete person-level covariates; "omit" fits the model on the complete-case subset and records omitted persons / omitted response rows in the returned population metadata while retaining the observed-person-aligned pre-omit table for replay/export provenance.

facet_shrinkage

Character. "none" (default) keeps the unshrunk fixed-effects estimates. "empirical_bayes" applies a post-hoc zero-centered normal-model adjustment to each non-person facet (Efron & Morris, 1973); fit$facets$others gains ShrunkEstimate, ShrunkSE, and ShrinkageFactor columns, and fit$shrinkage_report records

the per-facet prior variance and sum of retained weights (not model degrees of freedom). This descriptive approximation has no general James-Stein risk guarantee; see apply_empirical_bayes_shrinkage(). "laplace" is retained as a compatibility alias for "empirical_bayes"; it does not fit a penalized likelihood.

facet_prior_sd Optional numeric scalar. When supplied, the shrinkage prior variance is fixed at facet_prior_sd^2 instead of being estimated by method of moments. Useful for eliciting a prior from domain knowledge or a previous fit.

shrink_person Logical. When TRUE and facet_shrinkage is active, the same descriptive adjustment is applied to ordinary-JML Person estimates with available sampling SEs. Default FALSE. MML EAP/posterior SDs cannot be used as observations/sampling SEs for a second normal-model update; use FALSE with other estimators.

attach_diagnostics
Logical. When TRUE, diagnose_mfrm() is run once after the fit with residual_pca = "none", and the per-level SE, Infit, Outfit, InfitZSTD, OutfitZSTD, and PtMeaCorr columns from diagnostics$measures are merged onto fit$facets$others (non-person facets) and fit$facets$person (Person rows). This is convenient when downstream code expects a FACETS Table 7 style facet table with fit statistics in one place, and lets summary(fit) show per-person fit columns alongside the measure. For person rows, an existing posterior SE (typical for method = "MML") is preserved and the diagnostic SE is only attached when the existing column is empty. Adds diagnostic runtime (typically +1-2 s on moderate designs) and sets fit$config$attached_diagnostics = TRUE. Default FALSE preserves the minimal Facet / Level / Estimate layout.

checkpoint Optional list(file = ..., every_iter = ...). When supplied, the MML EM engine writes its state to file every every_iter outer EM iterations using checked same-directory replacement. If the file already exists when the fit starts, the engine resumes only when its versioned identity exactly matches the current data, model, parameter layout, constraints, quadrature, package version, and engine stage. Legacy, corrupt, or incompatible files fail closed. A non-converged pure-EM run may resume with a larger maxit; a completed checkpoint cannot re-enter the same iteration boundary. Only the EM engine (mml_engine = "em" or the EM warm-start step of mml_engine = "hybrid") honours the checkpoint; the direct optim() engine ignores it. Use this to make long MML EM fits crash-resilient on shared compute environments.

gpcm_mml_identification
Scale-identification convention for model = "GPCM" with method = "MML". "free_population" (the default) estimates an intercept-only person distribution $N(\beta_0, \sigma^2)$ when population_formula is omitted, while retaining geometric-mean-one relative slopes. This restores the common discrimination degree of freedom used by a conventional fixed-latent- variance GPCM; in the documented item-only overlap it is a one-to-one reparameterization of ConQuest scoresfree GPCM. An explicitly supplied population_formula is retained under this convention. With one slope family, "fixed_standard_normal" is the legacy restricted branch: it requires population_formula = NULL, fixes the person distribution to $N(0, 1)$, and also fixes the slope geometric mean to one. The latter is then a substantive relative-discrimination restriction rather than an identification requirement.

With two families, fixed N(0,1) instead accompanies a geometric-mean-one first family and free second-family slopes. This argument does not change JML, whose geometric-mean-one slope constraint is required to identify its freely estimated person coordinates.

`mml_integration`

Numerical integration for MML: `"fixed"` (default) uses a common Gauss-Hermite grid relative to each person's normal prior; `"adaptive"` centers and scales the grid at each person's posterior mode on every objective evaluation. Adaptive fitting uses the gradient of the moving-node objective, including changes in its center and width. It requires `method = "MML"`, `mml_engine = "direct"`, and no checkpoint. The choice also controls the fitted likelihood, Hessian, posterior person summaries, fitted-object scoring, and supported portable calibration. For these two modes `quad_points` remains the order; neither guarantees adequate accuracy at a particular order. Compare orders before reporting. During adaptive optimization, trial values that cause numerical overflow are rejected so the search can shorten its step. Starting values must still be evaluable, and final estimates must pass the usual convergence checks; rejecting a trial does not establish a parameter boundary or impose a discrimination upper limit. Adaptive fits with nonlinear coordinates have no completed probability- map identification audit; GPCM also lacks an adaptive slope-boundary audit. Their overall `InferenceReady` flag remains false. Supported native-location and RSM/PCM population intervals have separate source and numerical checks; see the MML integration workflow section below and `mfrm_facet_intervals()`. Available intervals retain their asymptotic assumptions and coverage cautions. ConQuest exports currently require fixed integration. `"auto"` first fits the fixed grid at `quad_points`, then checks every Person's log marginal likelihood, EAP and posterior SD against two adaptive orders. It also checks the change in the full marginal gradient and its absolute maximum at the retained and finer rules. If any criterion fails, it warm-starts adaptive refits at the orders in `mml_accuracy_control`. This fixed-to-adaptive sequence applies to ordinary full direct MML (RSM, PCM or one-slope-family GPCM), with no checkpoint. Population regression, anchors and interactions retain their existing model restrictions. On success the resolved fixed/adaptive rule and order control all subsequent fit calculations; inspect `fit$config$mml_accuracy` for controls, stage history and final Person checks. These checks concern the fitted response patterns; scoring new patterns reuses the resolved rule and needs its own integration review. The initial fixed rule is retained if its checks pass. Requested settings are preserved for replay; explicit sensitivity refits use the resolved rule at each requested order. If optimization fails or all refinement orders are exhausted, an error of class `mfrmr_mml_accuracy_error` carries `history`, `control` and the last parameter vector instead of returning a fit. In current errors (`provenance_version = 2`), `last_candidate` holds the last returned optimizer vector, its parameter-block names and sizes, its matching `config`, and its stage diagnostics. These are free optimization coordinates, not a table of facet estimates or a precision-qualified fit. `last_attempt` records the final attempted stage and failure phase; it may describe a later failed optimization. `last_integration` and `last_quad_points` now describe `last_parameters`, not that later attempt. If no optimizer returned, `last_candidate` is NULL, `parameter_source` is `"initial_values"`, and the

integration aliases are NA. Initial values are retained separately as initial_parameters. Earlier saved errors lack this provenance version; their integration aliases can describe the failed attempt rather than the retained vector. Inspect their stage history. Capture the condition with tryCatch(fit_mfrm(...), mfrmr_mml_accuracy_error = function(e) e) and inspect both records. Saving the condition preserves the values and reasons without converting unsuccessful accuracy checks into a successful model fit.

For two-family GPCM, "auto" instead preserves the requested engine: mml_engine="em" uses fixed integration and mml_engine="direct" uses adaptive integration at every stage. Each refit retains the existing neutral EM start or the specified adaptive gpcm_mml_start policy; no cross-order warm start or engine switch is introduced. The default hard ceiling is 241 points, including the 2*q-1 reference checks. Person log marginal likelihood, EAP and posterior SD differences must pass, together with the existing two-family local-information and stationarity checks described in confint.mfrm_fit(). This conservative automatic route therefore requires more than delivery of a point estimate. A sole quadrature-sensitivity failure or inaccurate Person moments causes refinement; another failed information, convergence or model check stops with its own reason. It does not automatically diagnose a boundary. In two-family errors, last_candidate$accuracy retains available checks and Person comparisons. If fitting failed before a candidate was returned, parameter_source="unavailable" and last_parameters=NULL; there is no single initial vector to report for a multi-start policy. No two-family global optimum or interval-coverage guarantee follows from auto success. The latest successful automatic fit can reuse its complete qualification for subsequent two-family intervals in the same R session, provided the entire fit, information-memory budget and numerical backend are unchanged. Source and result retention is bounded by that budget. Warning-producing or refused qualification is not reused. This transient cache is not saved in the fit; a new session recomputes.

category_policy

Optional explicit category choice: "collapse" maps gaps in the observed categories to consecutive scores; "preserve" keeps the intended ladder, declared with rating_min and rating_max. This changes the fitted category steps, not just labels. NULL (default) uses keep_original, whose default is FALSE ("collapse"). Supplying both choices is allowed only when they agree. Preservation does not estimate unsupported steps: fitting stops if a retained internal category has no observations. Use the same policy in describe_mfrm_data() and review_mfrm_anchors().

em_score_tol

Stopping tolerance for the two-family GPCM MML-EM route only. NULL selects 1e-6 for that route. Stops when the largest absolute derivative of the negative marginal log likelihood, divided by the number of Persons, is at most this value. It does not control interval eligibility. The ascent-checked BFGS M step allows 100 iterations with reltol = 1e-12; both the marginal likelihood and EM auxiliary objective must not decrease beyond numerical round-off. If BFGS stops before the requested score accuracy, usable positive curvature can refine the auxiliary objective with the same E-step counts. The ascent checks still apply. Dense curvature refinement is limited to 1–64 free parameters; otherwise the ordinary M-step proposal is retained. The saved

fitoptem_trace records refinement attempts and failures; mstep_score is the proposal's auxiliary-objective score before any step halving. Other model routes require NULL.

jml_correction_order

NULL (default) leaves the usual estimator unchanged. A positive integer explicitly selects an experimental profile-score adjustment for shared-owner model = "GPCM", method = "JML" fits. There is no automatic choice of order. See **Corrected JML** for the input restrictions, uncertainty meaning and connected outputs.

jml_correction_sampling

Sampling assumption for corrected JML's Person covariance: "fixed_rosters" (default) centers contributions within each observed assignment pattern; "random_rosters" centers globally when assignment patterns are sampled. This changes covariance, not the point equation. The fixed-roster calculation needs at least two Persons per pattern. Both describe new independent Persons, including variation in their ability composition; neither estimates the variance from reassessing the same Persons at fixed abilities. Omit this argument when no correction is requested.

gpcm_mml_start

Initial values for two-family adaptive direct MML, including its mml_integration="auto" refinement stages. NULL (default) selects "neutral_em": optimize from both the neutral vector and a same-data fixed-grid EM vector, and select the lowest finite adaptive negative log likelihood, retaining its convergence status. "neutral" retains the earlier neutral-only initialization. Numerical optimizer repairs still apply; exact historical reproduction requires the original source. An EM vector is only a start; its fixed-grid likelihood is never compared with an adaptive likelihood. The seed uses the requested quadrature order, maxit outer iterations, 100 M-step iterations and per-Person score tolerance 1e-6. A finite seed need not have converged. Each direct start uses the requested optimizer, reltol and its existing polishing sequence. All starts, errors, warnings, stage histories, seed trace and elapsed costs are stored in fitoptmml_initialization; results and reports include its comparison table. This comparison does not prove a global optimum or interval validity. Alternatively, supply a plain named list of at least two finite native free-parameter vectors. For example, reuse p <- previous_fitoptpar from the same model coding with list(previous = p, neutral = 0 * p). The order is first-owner free locations, second-owner locations, second-owner free steps (within each level), first-owner free log slopes, and second-owner log slopes, using the fitted level order. Vector lengths must match; optional parameter names must exactly match the model's native names. Reported positive slopes or expanded constrained parameters cannot be supplied in place of native coordinates. Explicit starts replace the default start roster and do not run an EM initialization. Their values are reevaluated and optimized under the requested adaptive objective; no likelihood, convergence flag, covariance or scoring eligibility is imported from a previous calculation. The roster and vectors remain in the saved initialization record and replay inputs, including quadrature refits.

repeated_ratings

Treatment of repeated Person-by-facet cells. "review" (default) retains the duplicate-cell warning and ordinary MML/JML input-review state. Use "independent" only to declare that the rows are distinct, conditionally independent rating events

under the specified model. Every row and its weight are retained; this argument does not aggregate scores, add a facet, change the likelihood or establish independence from the data. Other input, numerical and inference checks still apply. The declaration is saved in preparation notes, configuration and replay settings. Corrected JML already requires conditionally independent repeated ratings; the argument records that assumption explicitly there. It is not supported by the two-family GPCM route.

`centering_scope`

`"facet"` (default) centers each declared facet across all its levels, including direct anchors and group totals. `"remainder"` reproduces the earlier convention that centers only levels outside direct and group constraints. For ordinary additive RSM/PCM JML, `"anchor_origin"` lets direct or group anchors replace centering on their facet; select an unanchored, active `noncenter_facet`. Other facets remain centered. This option excludes comparison-only fitting. None of these conventions changes the population model or category steps. Unsupported specialized routes reject an explicit setting. See **Anchors and scale constraints**.

`fit_scope` `"full"` (default) fits the ordinary model and retains its identification guards. `"comparisons"` explicitly fits only the targets in `comparison_review`; see **Comparison-only estimation**.

`comparison_review`

For `fit_scope = "comparisons"`, the result of `describe_mfrm_data()` with `comparisons`, or its `$comparison_review`. Pass the review's `fit_arguments` and the same original ratings.

`mml_accuracy_control`

Named list used only with `mml_integration = "auto"`. To specify only a ceiling, use `list(max_points=121)`. It bounds every quadrature rule used by automatic fitting and its acceptance checks, including reference rules; it does not bound optimizer iterations, total runtime or later separately requested inference/scoring. Orders increase automatically and stop at the first accepted stage. `max_points` must be an integer of at least 7 for ordinary MML, or 3 for two-family GPCM; requested rules must be representable. Unused finer rules are not constructed. A ceiling and an explicit `adaptive_points` vector cannot both be supplied. An explicitly supplied `quad_points` must fit within the ceiling (including its finer reference for two-family fitting); an omitted starting order is reduced if needed. For example, the two-family ceiling 121 permits fitting/checking orders 31/61 and 61/121. A ceiling of 100 permits 31/61 and 50/99, preserving the same reference-order formula. Reaching the ceiling without passing all criteria raises `mfrmr_mml_accuracy_error`, retaining the attempted rules and source-bound candidate; it does not relax tolerances or return a precision-qualified fit.

For ordinary MML without a ceiling, the retained defaults are `adaptive_points = c(31, 61, 121)` (strictly increasing integer refit orders, each checked against `2*q+1`), `loglik_tol = 1e-7` (maximum absolute log-marginal difference per Person), `posterior_tol = 1e-6` (maximum absolute EAP or posterior-SD difference, on the ability scale), and `gradient_tol = 1e-4` (maximum absolute gradient or gradient difference, in free optimization coordinates for the summed negative log likelihood). At the initial fixed fit the first adaptive order and its `2*q+1` refinement are used. All criteria and optimizer convergence must pass.

These are empirical numerical tolerances, not certified error bounds, relative tolerances, a bound on parameter movement, or a guarantee of a global optimum or interval coverage. Weak curvature can amplify small gradient errors. Hessian precision and formal inference require their separate checks; automatic integration does not qualify previously unavailable identification or inference. `maxit` applies to each fit, so refinement increases total computation. Two-family controls instead default to `max_points=241`, `loglik_tol=1e-7` and `posterior_tol=1e-6`, with `2*q-1` reference rules. They do not accept `adaptive_points` or `gradient_tol`: their established engine-specific stationarity and information criteria are retained. Passing component-interval checks at an order does not ensure that its Person likelihood and posterior moments meet these additional tolerances. A prerequisite check that stops two-family refinement records `history$Status = "check_refused"`. Inspect the failed rows of `last_candidate$accuracy$checks`; this status does not identify the cause. A failed higher-order information check also stops refinement. Earlier saved histories use `"nonintegration_refusal"` for the same stopping rule; neither label establishes a boundary or whether more points would help.

`population_sd` Optional positive residual population SD, fixed during estimation. `NULL` preserves the existing behavior: an active population model estimates its residual variance; RSM/PCM without a population formula uses N(0,1). A numeric value requires `population_formula`, full direct RSM/PCM MML, and no interactions. It does not activate a population model implicitly. Supplied SD and anchor values are treated as fixed, without source-error propagation. GPCM and other estimation engines do not support this option.

`jml_penalty` `"none"` (default) preserves the ordinary estimator. Explicit `"jeffreys"` with additive RSM/PCM `method = "JML"` selects canonical information-penalized joint estimation. It returns a separate `mfrm_jeffreys_fit` with point estimates and numerical checks, without ordinary SEs or FACETS diagnostics. See **Jeffreys-penalized JML**.

`progress` Logical; `TRUE` displays a progress indicator with elapsed time. Two-family GPCM reports EM iterations, adaptive starts, quadrature stages and precision checks; direct optimization reports objective evaluations. Other routes may provide only a coarse phase display. Iteration/stage limits are budgets, not percent completion or a remaining- time forecast: convergence and required refinement are not known in advance. Display follows `cli` console settings and is cleared on return, error or interruption. The default is silent; it does not change numerical controls or become part of saved calibration/replay settings.

`mml_starts` Optional starting-value review for ordinary full RSM/PCM or single-family GPCM MML. `NULL` keeps the usual single start. A numeric vector such as `c(0, 0.5, -0.5)` adds each nonzero offset times the alternating vector `(1, -1, 1, ...)` to the default free coordinates. These deterministic offsets explore one direction in the fitted free- coordinate order, not random or exhaustive searches. The default start is always included. To explore other directions, supply a uniquely named list of named numeric coordinate values; obtain names from `fit$opt$start_review$parameter_map$Coordinate` after `mml_starts = 0`. Unlisted coordinates retain their default initial values. Anchored and dependent coordinates are expanded from the existing constraints. All attempts share the

same fixed or adaptive integration rule and order; automatic integration, checkpoints and two-family GPCM are unsupported. Each attempt uses the requested engine, with existing fallbacks recorded. The smallest reevaluated negative log likelihood is retained even when that attempt is unfinished; a worse converged attempt is not substituted. Inspect `summary(fit)$starting_values` and `fit$opt$start_review` for all starts, optimizer outcomes, warnings, errors and coordinate differences. If all attempts fail, the error condition retains `$start_review`. Missing curvature means it was not evaluated. This review increases fitting time and does not establish a global optimum, adequate integration or valid inference. Use `mml_quadrature_sensitivity()` for integration sensitivity. Same-model replay and resampling retain this policy. Subgroup DFF refits accept numeric offsets; named starts require an explicit subgroup fit because linking changes the free-coordinate basis.

`object`, `x` A fitted CML result for its summary, print, covariance and log-likelihood methods, or a comparison-only result for its summary and print methods.

`...` Unused.

Details

Data must be in **long format** (one row per observed rating event). Exact duplicate Person-by-facet combinations are retained, warned once, and propagated as a Data review state by default. Repeated cell keys alone do not establish dependence. With `repeated_ratings = "independent"`, the analyst declares distinct rating events that are independent conditional on Person ability and the model's effects. Keep event IDs in the source data for audit. Event IDs alone do not demonstrate independence; correlated re-ratings require a suitable dependence model. Adding an estimated occasion facet changes the response model and is appropriate only when intended. Repeated ratings do not increase the number of independent Persons.

Value

With `method = "JML", jml_penalty = "jeffreys"`, a separate `mfrm_jeffreys_fit`; see `summary.mfrm_jeffreys_fit()` for its point tables, probabilities, numerical checks and unsupported inference. With `method = "CML", fit_scope = "full"`, an `mfrm_cml_fit` with structural estimates, joint `covariance` and `information` (missing where unavailable), `person_totals`, `checks`, preparation, configuration and optimizer controls; see "Conditional maximum likelihood". With `fit_scope = "comparisons"`, an `mfrm_comparison_fit` containing `comparisons`, the prospective `comparison_review`, numerical/category/boundary `checks`, preparation/configuration, controls and audit-only parameter `representation/optimization`. CML adds contrast `covariance` and `person_totals`. Its `summary()` returns the comparison table. Otherwise, an object of class `mfrm_fit` (named list) with:

- `summary`: one-row model summary including `LogLik`, `Deviance`, canonical free dimension `Npar`, response-row/weight/Person counts, the versioned information-criterion contract, Person-based MML `AIC`/`BIC`/`SABIC`, explicitly descriptive legacy fields when the common panel is ineligible, and convergence; it also includes user-facing `Method`, engine-facing `MethodUsed`, MML-engine fields, terminal-gradient readiness, requested/selected-stage tolerance settings, and the actual L-BFGS-B `OptimizerFactr` / `OptimizerPgtol` controls when applicable
- `facets$person`: person estimates (`Estimate`; plus `SD` for MML), with `SourceFitReadiness`, `SourceInferenceReady`, and `EstimateUse` separating a defined Person summary from the

source fit's permission to interpret it. In particular, a finite prior-regularized MML EAP does not become reportable when the source fit is blocked.

- `facets$others`: facet-level estimates for each facet
- `steps`: estimated threshold/step parameters as a one-row-per-step `tibble` with `Estimate`. Bare fits keep this table as point estimates. `diagnose_mfrm()` exposes MML observed-information step uncertainty in `diagnostics$parameter_uncertainty$steps`. Check `CIEligible` and `CIUse`: finite curvature-based bands alone do not establish ordinary inference. When `attach_diagnostics = TRUE`, those `SE`, confidence-limit, and status columns are attached to `fit$steps` when the Hessian is available. For step-structure quality, also use the step-collapse and disordering warnings from `diagnose_mfrm()` and `category_structure_report()`.
- `slopes`: discrimination parameters for `GPCM` fits as a one-row-per-slope-element `tibble`. `LogEstimate` and `Estimate` retain the finite optimizer values for compatibility and numerical diagnosis; `OptimizerLogEstimate` / `OptimizerEstimate` name that role explicitly. Read `ParameterStatus`, `PrimaryLogEstimate`, `PrimaryEstimate`, `SEEligible`, `CIEligible`, and `ReasonCodes` before interpretation. A certified JML slope-only path receives a typed extended-real primary boundary. `config$boundary_audit` retains the supporting fixed-objective, joint-path, and terminal-gradient records. A certified path can establish that a finite JML maximum is unattained for the evaluated case. The converse is deliberately not used: failure to find a path in the evaluated families, or retention of a finite optimizer point, does not establish existence of a finite global maximum for the non-concave GPCM likelihood. These JML diagnostic records do not supply uncertainty estimates, MML results or evidence of agreement with other software. `config$boundary_audit$gpcm_terminal_gradient_stability` reconstructs the same fixed JML objective and analytic terminal gradient, checks stored optimizer/polish summaries and deterministic central-difference probes, and reports gradient norms by free-parameter block. Positive boundary certificates take precedence over a finite-point zero or small gradient; otherwise a coherent small gradient is retained-point first-order evidence only. The numerical tolerance is not a statistical significance threshold. These checks do not establish a finite global maximum, boundary absence, uncertainty, external comparability, or readiness. The conditional JML boundary checks are not reused for MML. `confint(fit, parm = "slopes")` and `diagnose_mfrm()` separately check the current local MML solution before supplying approximate pointwise relative-slope intervals. `CIEligible` and `InferenceReview` record this output-specific decision, without overriding the global boundary audit. Ineligible local calculations remain in `Optimizer*SE` / `Optimizer*CI`. Refresh saved fits through either function; no refitting is necessary. The identification convention pins the geometric mean of finite optimizer slopes at 1.
- `readiness`: the versioned fit record, five component rows, and current parameter-level rows. The current parameter slice includes GPCM slopes; other non-Person parameter classes remain scheduled for later propagation.
- `interactions`: model-estimated facet interaction effects and metadata when `facet_interactions` is supplied
- `population`: population-model metadata. Ordinary RSM/PCM and JML fits keep an inactive record (`active = FALSE`, `posterior_basis = "legacy_mml"`). Default GPCM MML and active latent-regression fits store the fitted design matrix, regression coefficients, residual variance, omission review, the complete-case estimation table (`person_table`), and the observed-person-aligned replay/export provenance table retained before complete-case omission (`person_table_replay`), plus stored categorical `xlevels` / `contrasts` for model-matrix replay and scoring, together with `posterior_basis = "population_model"`. `estimation_converged`

records optimizer convergence and `inference_ready` records the separate formal readiness decision. The older `population$converged` field is retained as a compatibility alias of `inference_ready`; its basis is recorded in `population$converged_basis`.

- `data_review`: pre-fit Data, Design, Stability, and Reporting readiness evidence propagated into summaries and plot-interpretation checks
- `config`: resolved model configuration used for estimation, including `config$anchor_review` and the recorded estimation controls
- `prep`: preprocessed data/level metadata
- `opt`: optimizer result augmented with `optimizer_diagnostics`, the complete `optimizer_polish` stage history, method-selection metadata, and evaluation-cache counters. For direct fitting, its core fields originate from `stats::optim()`; EM additionally records engine-specific diagnostics.

Choose a model and its population

For the `Rater` and `Criterion` columns in the Examples, the model choices are below. A slope family gives one discrimination per level of its owner; "two families" does not mean two ability dimensions or just two slopes. No Q matrix is required or accepted by this unidimensional API.

Model	Step owner	Slope owner	MML population
`RSM`	Shared	Fixed at one	Fixed N(0,1)
`PCM`	`Criterion`	Fixed at one	Fixed N(0,1)
One-family `GPCM`	`Criterion`	`Criterion`	Estimated mean and SD
Two-family `GPCM`	`Rater`	`Criterion`, then `Rater`	Explicit fixed N(0,1)

Supply these owner names through `step_facet` and `slope_facet`; for two families use `slope_facet = c("Criterion", "Rater")`. The first three rows show their default MML populations. Two-family settings must be chosen explicitly; see "Two slope families" below. One-family slopes are relative.

Changing only `model` therefore need not hold the population assumptions constant. For RSM/PCM, estimating its mean and SD requires `population_formula = ~ 1` and `person_data = unique(ratings["Person"])`. This is a model choice, not a universal recommendation. Fixing both the population SD and the one-family GPCM slope geometric mean to one is a narrower model, not just an alternative reporting scale.

Start with the complete RSM Examples. For PCM, add `step_facet` and change `model`; for one-family GPCM, also explicitly choose `slope_facet` and read the population choice above. Keep the data-column and rubric arguments. The complete two-family call and its narrower output workflow are in `vignette("mfrmr-gpcm-scope", package = "mfrmr")`, under "A provisional two-family workflow". Its zero-based scores and ordered owners require the explicit preparation described there; changing `slope_facet` alone is insufficient.

Choose an estimator and the result you need

The ordinary worked example uses RSM/MML. Choose the model and estimation assumptions for your question before choosing a table or figure.

- **Abilities, a population model or GPCM inference:** use `method = "MML"` and inspect `summary(fit)$decision`. Its interpretation depends on the declared ability distribution. Ordinary RSM/PCM fixes N(0,1) by default; `population_formula = ~ 1` instead estimates its mean and SD. GPCM estimates them by default. See the population and estimator sections.
- **RSM/PCM structural effects without an ability distribution:** use `method = "CML", category_policy = "preserv` Read `summary(fit)$locations` and `$steps`, then `mfrm_facet_intervals()`. CML returns no Person abilities or population SD; ordinary diagnostics and consolidated reports do not apply. The conditional-likelihood section below and the facet-interval vignette give its assumptions.
- **Joint point estimates or FACETS comparisons:** explicitly select JML when that estimand is intended. With few ratings per Person, adding Persons does not remove incidental-parameter bias. Stored structural SEs or corrected-JML root uncertainty are not automatically truth intervals.

Facet names are your column names, not fixed roles imposed by the package. For example, `facets = c("Judge", "Task")` uses those two columns; it does not require columns called Rater or Criterion. Supply the same names to follow-up functions and use the fitted level labels for contrasts or curves. Begin with the Examples section, then read only the specialist sections for your selected model. mfrmr_methods_reference separates measurement, G-theory assessment planning and observed-score agreement questions.

Choose the arguments by their purpose

- **Identify the ratings:** `data` is the rating table; `person`, `facets` and `score` are quoted column names, not the values in those columns.
- **Declare the rubric:** set `rating_min`, `rating_max` and `category_policy` consistently with `describe_mfrm_data()`. The default can collapse gaps; preservation can reveal an unsupported category step.
- **Choose the statistical model:** `model`, `step_facet`, `slope_facet`, anchors and population arguments determine what is estimated. Ordinary RSM/PCM MML with `population_formula = NULL` fixes N(0,1); the default GPCM MML instead estimates the normal population mean and variance.
- **Control computation:** `quad_points`, `maxit`, `reltol`, `optimizer` and `mml_engine` govern numerical fitting. Increasing them does not change the model's support or automatically justify statistical inference.
- **Choose follow-up output:** `attach_diagnostics = TRUE` computes and attaches diagnostics; the default leaves that separate. `summary(fit)` explains the result's status; `diagnose_mfrm()` reviews response fit.

Corrected JML

When each Person has few ratings, ordinary JML can retain structural bias even with many Persons. The explicit `jml_correction_order` option adjusts the profile-score equation to address this source of bias. It changes the estimator, not the response model. A higher order is not necessarily more accurate: bias, variance and numerical stability can change in different directions. Choose the order as part of the analysis plan and examine sensitivity; neither AIC nor the smallest RootSE chooses a justified order.

This experimental route requires the optional `nleqslv` package, observed integer scores, declared `rating_min` and `rating_max`, additive fixed facets, and a single explicit `step_facet` that also owns the slopes. Facet names are unrestricted. Use `category_policy = "preserve"` to retain the declared score ladder. All facet locations and each step ladder sum to zero; slopes have geometric mean one. Persons are independent sampling units, with conditionally independent ratings within Person. Only observed assignments contribute; unassigned or missing scores are not imputed. Repeated cell rows are treated as independent responses, not correlated repeated measurements. Fixed anchors, nonunit weights, interactions, shrinkage, separate slope/step owners, corrected RSM/PCM, and ordinary optimizer controls are unavailable. Explicit unsupported arguments produce an error rather than being ignored.

For an order k, the equation uses $U_k = (I - K_\beta)^k U_0$, where U_0 is the negative profile-likelihood score and K_β averages over responses at their profiled Person MLE. This is the MLE plug-in construction discussed by Dhaene and Weidner (2023, section 8.1), applied here to the declared GPCM structure. Exact owner-total enumeration is limited to 5,000 joint states per assignment pattern; larger calculations stop explicitly. No approximate pruning is substituted. The full equation Jacobian and empirical Person contributions give a sandwich covariance, with the selected assignment-sampling assumption. This is not an inverse likelihood Hessian or a general proof of bias removal.

For example, `"fixed_rosters"` describes new cohorts with the same number of Persons assigned to each rater/task combination. It does not hold each Person's ability fixed across repeated cohorts. Ability distributions are unspecified and may differ between assignment patterns. With `"random_rosters"`, both Persons and their assignments are sampled from the same joint population; assignment need not be independent of ability. Neither option models shared random raters or dependent Persons. The empirical sandwich has no small-sample degrees-of-freedom correction. Its fixed-roster interpretation requires enough independent Persons within each pattern; two Persons is a computational minimum, not an assurance of accurate uncertainty. Taking `"random_rosters"` solely to obtain a RootSE changes the sampling assumption and is not a repair for sparse information.

Use `summary(fit)$tables` for locations, steps, relative slopes and the uncertainty explanation; `include_person = TRUE` adds Person profiles. `RootSE` describes local variation around the adjusted-equation solution, which may remain biased for the true structural parameter. `LogRootSE` describes log-slope variation. Neither is supplied as a standard error with established structural coverage; no confidence limits are constructed. Persons are reprofiled at the adjusted calibration, with infinite limits for extreme scores and no invented Person SE. Numerical attempts remain available in `summary(fit)$attempts` and `fit$jml_adjustment`.

A valid point estimate is retained if covariance is unavailable, with its reason. Different accepted roots yield missing primary estimates rather than selection by the ordinary likelihood. Agreement between starting values is a local check, not proof of a unique global solution. A failure remains a reportable fitted object with missing estimates and its attempts.

`plot(fit, type = "slopes")`, `"locations"` (select one `facet`) and `"steps"` show point estimates. Choose `style = "distribution"` for an empirical cumulative view; `as_ggplot(plot(fit, draw = FALSE))` is supported. Use `mfrm_results(fit)` then `mfrm_report()` or `export_mfrm_results()` to retain these meanings through reporting and saved replay. For conditional probabilities and descriptive residuals on the observed fitted rows, use `mfrm_response_diagnostics()` and attach the saved result with `mfrm_results(fit, response_diagnostics = result)`. This holds corrected calibration and Person profiles fixed; no posterior averaging, uncertainty intervals or fit cutoffs are supplied. Extreme-score probabilities remain available; zero variance prevents standardization, with separate availability for Infit and Outfit. See that helper's help before interpreting a summary. For

new Persons, predict_mfrm_units() and extract_mfrm_calibration() provide conditional EAP scoring after their corrected-equation source checks. These scores add a separate normal reference prior. Their posterior intervals condition on calibration and do not establish structural coverage or remove residual bias. The portable artifact omits training responses and Person estimates; its own summary and score plots retain the correction order. Ordinary fit tests, Wright/Pathway maps, structural confidence intervals, likelihood ranking and corrected Person ML/WLE remain unavailable. The default MML and uncorrected JML workflows are unchanged.

Two slope families

To estimate, for example, criterion and assessor discrimination together, use `slope_facet = c("Criterion", "Assessor")`. The order declares roles: the first facet has centered locations and slopes with geometric mean one; the second has free locations/slopes and owns the centered category steps. The product of their slopes multiplies the complete adjacent-category predictor. There is still one ability dimension. Column names and data column order do not select these roles.

This route requires exactly those two non-Person facets, `model = "GPCM"`, `method = "MML"`, `gpcm_mml_identification = "fixed_standard_normal"`, either fixed integration with `mml_engine = "em"` or `mml_integration = "adaptive"` with `mml_engine = "direct"`, and both `step_facet` and `noncenter_facet` set to the second slope facet. These options must be chosen explicitly; the one-family defaults are not silently replaced. Use observed, unweighted integer scores on a scale from zero to `rating_max`, without missing IDs, duplicate person-facet rows or surrounding ID whitespace. Categories are not collapsed; if any are absent, declare `rating_max` and `category_policy = "preserve"` so category-support checks can assess that scale. Unobserved assignments are not zero scores. Anchors, population covariates, shrinkage, interactions, positive/dummy facets, checkpoints and automatic diagnostics are not supported in this route; explicitly supplying their arguments, including unused policy controls, produces an error. Fixed-grid EM uses `em_score_tol`; adaptive direct MML uses `reltol` and the direct optimizer's gradient check. Passing the other engine's tolerance is an error. Adaptive integration preserves the same N(0,1) population and response equation while moving each Person's grid. It is direct maximization of the marginal likelihood, not adaptive EM. By default it compares neutral and EM-derived starts through that same adaptive objective; see `gpcm_mml_start`. A better unfinished solution is retained as unfinished, rather than replaced by a worse converged candidate. If a retained starting point is better than all terminal candidates beyond roundoff, the returned fit remains numerically unresolved. A failed alternative is disclosed; if every direct attempt fails, the error carries an `initialization` record. Neither fixed-grid likelihoods nor interval outcomes select a candidate. Extreme slopes and weak information can still prevent inference even after a successful start comparison. Use mml_quadrature_sensitivity() to compare refits at different orders; it preserves the chosen integration method, engine and initialization policy.

`summary(fit)` and `print(fit)` retain numerical status and the two slope references. mfrm_curve_intervals() evaluates provisional category or per-rating information curves at supplied abilities; all their intervals are unavailable. Use `plot(curves)` and `mfrm_results(fit, include = c("fit", "plots"), compute = "never", intervals = list(curves = curves))` to report saved curves. Their values can be inspected even after nonconvergence, but are not qualified estimates. Separately request `confint(fit, parm = "slopes")` for experimental component log-Wald intervals from the full observed marginal information. Their numerical checks do not qualify global identification or sampling coverage; failed checks retain missing bounds. Attach the result alongside curves for saved plots/reports. To profile one component with the fitted integration method, explicitly request `confint(fit, parm = "slopes", method = "profile", slope = c(Task = "t1"))`, replacing the named owner and level with your fitted identifiers. This reoptimizes other coefficients; it

is an experimental local interval without established coverage. mfrm_response_diagnostics() integrates each Person's ability posterior at the saved calibration for descriptive residuals. Both slopes are retained; unavailable integration stays explicit. There are no reference fit cutoffs. Attach this result through response_diagnostics in the results call above. Adaptive two-family log-Wald checks use the moving-node marginal objective and compare adaptive quadrature orders. Posterior residuals preserve the fitted integration method and complete conditioning record, with separate row-wise integration checks. Adaptive profile searches recalculate the nodes, scales and Jacobians at every constrained trial and keep separate checks. predict_mfrm_units() separately supplies experimental conditional new-Person EAP and posterior intervals with the retained N(0,1) prior, known levels, unit weights and separate source/batch checks; calibration uncertainty is excluded. mfrm_facet_intervals() separately supplies experimental model-based intervals for either owner's locations and within-facet contrasts, retaining slope/step nuisance uncertainty. Failed numerical checks leave missing bounds; location differences need not imply uniform rating differences. Attach these intervals to mfrm_results() for plots, reports and exports. Step/curve intervals, sandwich inference, model ranking/LRT, ordinary fit/bias diagnostics and Wright/Pathway plots remain unavailable. extract_mfrm_calibration() provides a separately checked portable two-family route. Matching native or portable scores can be attached with mfrm_results(fit, scores = scores) without rescoring. Numerical agreement of the fitting implementation is not a general identification, convergence or coverage guarantee. See vignette("mfrmr-gpcm-scope") for a complete example and interpretation.

GPCM model and inference

One selected facet supplies level-specific positive discriminations. With MML, a different facet may supply category steps. For example, slope_facet = "Criterion", step_facet = "Rater" estimates a relative discrimination for each criterion and category-step contrasts for each rater. It does not simultaneously estimate rater discriminations. JML requires the same owner for both blocks. The model has one substantive ability dimension; its structural choices and currently unavailable inferential outputs are separate considerations.

Free-slope fits retain numerical estimates for review. confint.mfrm_fit() with parm = "slopes" supplies approximate relative-slope intervals for eligible MML solutions. MML information-criterion ranking uses separate likelihood and local-solution checks in compare_mfrm(). compare_mfrm() and build_weighting_review() also accept nested = TRUE for an equal-slope PCM/GPCM test after verifying matching population and constraint settings. Fitted-object scoring and information have their own scope and do not imply a portable GPCM calibration artifact. Consult gpcm_capability_matrix() for each operation before using its output.

Model

fit_mfrm() estimates many-facet ordered-response models. The RSM and PCM branches follow the many-facet Rasch-family tradition (Linacre, 1989); the GPCM branch extends the partial-credit kernel with estimated positive slopes under the package's documented identification constraints. For the equal-slope RSM/PCM branch, a two-facet design (rater j, criterion i) is:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k - 1)} = \theta_n - \delta_j - \beta_i - \tau_k$$

where θ_n is person ability, δ_j rater severity, β_i criterion difficulty, and τ_k the k-th Rasch-Andrich threshold. Any number of facets may be specified via the facets argument; each enters as an

additive term in the linear predictor η.

With `model = "RSM"`, thresholds τ_k are shared across all levels of all facets. With `model = "PCM"`, each level of `step_facet` receives its own threshold vector $\tau_{i,k}$ on the package's shared observed score scale.

One response-model family is used per `fit_mfrm()` call. The current public interface does not combine binary, RSM, PCM, or GPCM observations in one fit, define multiple independent rating scales, or accept general threshold/scale anchors and fixed-calibration starting values.

With `model = "GPCM"`, the adjacent-category kernel is multiplied by a positive slope for the designated slope-facet level:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k-1)} = \alpha_g(\eta - \tau_{h,k}), \quad \alpha_g > 0.$$

Here g indexes the slope facet and h the step facet; MML permits these to differ. Slopes have a sum-to-zero constraint on log slopes, so their geometric mean is 1. Step contrasts are centered within each step owner, separately from facet location effects. Crossing the facets and observing enough categories is necessary to distinguish their effects; allowing separate owners does not ensure identification in a given design. A selected facet owns a vector rather than one common number: if there are G levels, the fit returns G positive slopes with $G - 1$ free log-slope contrasts. Every other facet remains additive inside η and receives no separate slope. Selecting a rater facet is therefore a different restricted model from selecting a criterion or task facet. The placement of the slope is part of the model identity: it multiplies the complete adjacent-category predictor, including the person coordinate, all additive facet locations and fitted facet interactions inside η, and the owned step. It is not a loading-only formulation in which the slope multiplies ability while rater severity and other intercept terms remain unscaled. Such a formulation, including TAM multifacet `GPCM.design` constructions with separate linear intercept and slope designs, is a different model unless an algebraic reduction establishes equivalence. In this one-family many-facet GPCM, exactly one facet supplies slopes. MML permits a different step owner; JML requires a shared owner. It is not the broader Uto–Ueno generalized MFRM, whose task and rater slopes enter multiplicatively and whose step owner must be stated separately. The provisional two-family route above fits that product equation with its own scale and output restrictions. Setting every slope to one recovers the equal-discrimination PCM kernel; neither route supplies multidimensional traits or response-style parameters. Under the default one-family `gpcm_mml_identification = "free_population"` branch, the population standard deviation carries the common discrimination scale while the geometric-mean-one slopes describe relative discrimination. Equivalently, on a standardized latent variable the absolute slopes are $\sigma\alpha_g$. Under `gpcm_mml_identification = "fixed_standard_normal"`, both the population standard deviation and slope geometric mean are fixed to one; that legacy branch is a narrower relative-discrimination model. Under JML, the geometric-mean-one constraint is required to resolve the ability/slope scale because person coordinates are estimated jointly.

The model name does not imply finite parameter bounds. The JML branch maximizes the identified joint log-likelihood without a statistical penalty or finite bounds on person, location, step, or slope coordinates. Numerical line-search rejection of non-representable slope proposals is not regularization. When a recession direction is certified, the finite optimizer iterate remains a numerical trace and the primary result uses the appropriate extended-real or typed boundary status; it is not relabelled as a finite maximizer of the original JML objective.

With only two ordered categories ($K = 1$), the `RSM/PCM` branch reduces to the usual binary Rasch logit for the single category boundary:

$$\ln \frac{P(X_{n\cdot} = 1)}{P(X_{n\cdot} = 0)} = \eta - \tau_1$$

GPCM uses the slope-scaled counterpart $\alpha_g(\eta - \tau_{g,1})$.

With method = "MML", person parameters are integrated out using Gauss-Hermite quadrature and EAP estimates are computed post-hoc. With method = "JML", all parameters are estimated jointly as fixed effects. "JMLE" remains an accepted compatibility alias, but package output now uses "JML" as the public label. See the "Estimation methods" section of mfrmr-package for details.

Logistic metric (D = 1)

Native RSM, PCM and GPCM models, including two-family GMFRM, use the logistic metric (L-metric), with scaling constant $D = 1$. No additional normal-ogive approximation factor such as 1.7 or 1.702 multiplies the adjacent-category predictor. This response-scale convention is separate from the ability distribution: fixing a population to $N(0, 1)$ does not change the link to a normal-ogive model or introduce a 1.7 multiplier.

In RSM/PCM, increasing ability by one unit, holding everything else fixed, increases each adjacent-category log odds by one (one logit). In one-family GPCM, the change is α_g logits; in two-family GMFRM it is the product of the two component slopes. Thus a one-unit change in ability is not universally a one-logit change when slopes vary.

Before comparing or importing parameter values from other software, reconcile its scaling constant, slope placement and identification constraints with this model. Changing only the multiplier while retaining all parameter values changes response probabilities. An equivalent reparameterization must also transform any affected population parameters, anchors and, for Bayesian estimation, priors. See export_mfrm_model() for the same L-metric in the supported Stan/brms exports.

Weighting policy

mfrmr treats RSM / PCM as the equal-weighting reference route for operational many-facet measurement. In that Rasch-family branch, discrimination is fixed, so the scoring model does not differentially reweight item-facet combinations through estimated slopes.

GPCM is supported as an alternative when users explicitly accept discrimination-based reweighting. This often improves model fit, but the package does not treat better fit alone as a sufficient reason to replace an equal-weighting Rasch-family model.

The weight argument is separate from that modeling choice. It supplies an observation-weight column; it does not create a free-form facet-weighting scheme and does not change the fixed-discrimination contract of RSM / PCM.

Input requirements

Minimum required columns are:

- person identifier (person)
- one or more facet identifiers (facets)
- observed score (score)

Scores are treated as ordered categories. Although the fitted category probabilities form a multinomial probability vector, category order is part of the likelihood: unordered nominal/multinomial-logit responses are not supported. Poisson, negative-binomial, and grouped binomial-trial counts are also not response families in `fit_mfrm()`. Integer counts supplied as `score` are interpreted only as ordered category codes. Non-numeric score labels are dropped with a warning after coercion, whereas fractional numeric scores outside the rounding tolerance described above are rejected with an error instead of being silently truncated.

A positive numeric `weight` may encode a replication/likelihood weight for an ordered-rating row when weighting that conditional contribution is the intended estimand. It is not a general collapsed-person frequency-table interface: under MML, powering responses inside one Person's conditional pattern is not the same as replicating a complete Person pattern after marginalization. A weight also does not turn the score into a count outcome or model dependence among repeated ratings. Non-positive finite weights are excluded during preparation, and non-unit observation-weight fits are not eligible for ordinary inference, facet equivalence, or the common MML information-criterion panel under the current package contract. Normalizing weights to mean one does not remove this restriction. Point estimates and computed curvature/posterior precision remain available for diagnostic review; they do not establish sampling SEs or confidence coverage for the weighted objective. Omitted weights and explicitly all-unit weights use the same eligibility rules. Earlier saved fits require refitting or a current readiness audit; their old inference flags are not carried forward.

The fitted many-facet ordered-response model assumes conditional independence of observations given the person and facet parameters (Linacre, 1989). Repeated ratings of the same person-criterion combination by the same rater violate this assumption. When such structures may be present, follow fitting with `diagnose_mfrm(fit, diagnostic_mode = "both")`; its `strict_pairwise_local_dependence` screen is an exploratory check for residual dependence beyond what the additive linear predictor absorbs.

Binary responses are therefore supported as ordered two-category scores (for example `0/1` or `1/2`) under the same ordered-response interface. If your observed categories do not start at 0, set `rating_min`/`rating_max` explicitly to avoid unintended recoding assumptions. For example, if the intended instrument is a 1-5 scale but the current sample only uses 2-5, set `rating_min = 1, rating_max = 5` to retain the zero-count category 1 in the data-support review. That boundary absence is a review condition for the separate element-boundary contract, not by itself an unsupported free step contrast. By contrast, retaining an unobserved internal category in a polytomous fitted ladder creates an adjacent-step recession direction, so `fit_mfrm()` stops before optimization rather than reporting finite step estimates for that ladder. If these bounds are omitted, the observed score range is used and the provenance is stored in `fit$prep` and `summary(fit)$settings_overview`. Set `options(mfrmr.show_inferred_rating_range = TRUE)` when you want an interactive reminder whenever a bound is inferred. Data-preparation events such as row drops, ID trimming, duplicate person-by-facet cells, and single-level facets are stored in `fit$prep$row_retention` and `fit$prep$preparation_notes`. Routine row-drop/trim/single-level messages are quiet by default; set `options(mfrmr.show_preparation_messages = TRUE)` to show them during interactive checks.

When `keep_original = FALSE`, observed gaps such as `1, 3, 5` are recoded internally to a contiguous scale (`1, 2, 3`) and the mapping is stored in `fit$prep$score_map`. To retain zero-count intermediate categories as part of the original scale, set `keep_original = TRUE` in addition to supplying the full `rating_min` / `rating_max` range.

Facet effects and rating counts

In MML, person ability is integrated out under a `N(0, 1)` distribution (or under the `N(X\beta, \sigma^2)` population model when `population_formula` is supplied). GPCM MML instead activates an intercept-only `N(\beta_0, \sigma^2)` population model by default so its common discrimination scale is estimable. JML instead estimates person parameters; native CML conditions on person totals. Non-person facet parameters (`Rater`, `Criterion`, `Task`, ...) are fitted as fixed effects, identified by the configured centering or anchor constraints. By default, these facets have no hierarchical prior or shrinkage. Optional `facet_shrinkage` applies a separate post-estimation adjustment; see its argument documentation for scope and uncertainty limitations.

Few ratings can limit information about an individual facet level; few distinct raters also limit what can be learned about a wider rater population. These are different concerns, and neither is resolved by a count threshold. Precision depends on targeting, connectedness, category use, dependence and the estimator. Jones and Wind (2018) examine repeated ratings in incomplete designs, finding particular sensitivity in examinee estimates. Useful reviews include:

- `facet_small_sample_review()` for per-level retained rating counts, count-screening labels and available SEs.
- `detect_facet_nesting()` and `analyze_hierarchical_structure()` when raters are nested in regions, schools, or other strata that the additive fixed-effects MFRM cannot partition out.
- `compute_facet_icc()` and `compute_facet_design_effect()` for descriptive variance- component summaries based on `lme4` (optional).

`fit$summary$FacetSampleSizeFlag` summarizes the lowest rating-count band across non-person facet levels (`"sparse"` < 10, `"marginal"` < 30, `"standard"` < 50, `"strong"` >= 50). These package screening bands are not guarantees of precision, interval coverage or sample adequacy.

Jeffreys-penalized JML

`jml_penalty = "jeffreys"` minimizes $-\ell(\beta) - \frac{1}{2}\log|I(\beta)|$ in the actual affine free canonical coordinates, with anchor offsets retained. Here I is the full joint expected response information, including free Person parameters. The optimizer starts at the free-coordinate origin, uses BFGS with `reltol = min(reltol, 1e-14)` and the requested `maxit`, and requires maximum penalized gradient at most `1e-4`, positive information with checked solves, and numerically interior category probabilities. It uses no probability clipping, parameter box, ridge or ordinary extreme-score replacement. If BFGS stops on objective change before satisfying the gradient criterion, at most three local Newton steps use the analytic penalized Jacobian. Steps must be numerically identified, at most one in every free coordinate, not increase the same objective and reduce the full penalized gradient; backtracking is bounded. `checks$polish` records an attempted finish. The original gradient criterion and failure status remain in force if finishing fails.

The supported scope is an identified additive RSM/PCM design, unit weights, explicitly declared `rating_min` and `rating_max`, and unchanged full score coding (`category_policy = "preserve"`). Direct finite anchors and ordinary centering conventions are supported; group anchors, interactions, GPCM, population models, shrinkage, corrected-JML arguments and attached ordinary diagnostics are not. `optimizer` must be `"auto"` or `"BFGS"`. Repeated cells require the explicit independent-occasion declaration when substantively justified. Missing categories and constant outcomes are retained on the declared rubric; rank deficiency is still refused.

`summary.mfrm_jeffreys_fit()` describes Person, facet and step points, category probabilities and numerical diagnostics. For common-plan Person expected scores use `fair_average_table()` with

reference_plan, followed by confint.mfrm_fair_average() for experimental refit or delta intervals. Calibration, scoring and generating fits must all select this estimator; every replicate re-estimates its calibration and anchors using the same equation. The ordinary delta derivative and inverse information SEs do not describe this estimator; its delta route uses analytic penalized score/anchor derivatives and shared categorical response covariance. It is not an inverse-curvature interval. Parameter covariance, Wald intervals, AIC, likelihood-ratio inference, ordinary FairM/FairZ diagnostics and FACETS export are not supplied.

This is an opt-in alternative to ordinary FACETS-oriented JML, separate from both the generator-only Jeffreys option and corrected GPCM. Firth's (1993) fixed-dimensional canonical bias result does not establish unbiased nonlinear expected scores, coverage, or incidental-parameter consistency when the number of Person parameters grows. The binomial finiteness theorem of Kosmidis and Firth (2021) is not asserted as a general many-facet PCM theorem. A qualified local stationary point does not establish a global optimum. Small-sample, sparse and weak-link bias/coverage remain to be qualified independently. References: Firth (1993), doi:10.1093/biomet/80.1.27; Kosmidis and Firth (2021), doi:10.1093/biomet/asaa052.

Estimator choice and the JML incidental-parameter caveat

Joint maximum likelihood (method = "JML" / "JMLE") estimates both the structural parameters (facets, thresholds, slopes) and every person measure as fixed parameters in one optimization. This is the **incidental-parameter problem** of Neyman & Scott (1948): structural-parameter bias can persist as the number of persons grows with the number of items per person held fixed. In classical Rasch settings this bias can be of order $1/L$ (where L is the number of items per person) and therefore need not vanish by adding persons alone. Wright & Stone (1979) and Wright & Masters (1982, ch. 5) document an empirical $(L-1)/L$ correction that approximately removes the bias for the dichotomous Rasch model; mfrmr does **not** apply that correction (no bias_correction argument exists). A common positive multiplier is not a correction for relative GPCM slopes: multiplying all slopes by the same factor leaves their ratios unchanged, and rescaling them to geometric mean one returns the original slopes. A correction for relative slopes therefore needs its own justification.

Uncorrected JML facet/location SEs use observation-table information:

$$\mathrm{SE}_g \approx \left[\sum_{r \in g} w_r a_r^2 \mathrm{Var}(X_r \mid \widehat{\eta}_r)\right]^{-1/2}.$$

Here g is a facet level, r an observed response row, w_r its weight, and a_r its GPCM slope (one for RSM/PCM). Only finite information contributions are used. These exploratory SEs treat the other fitted parameters as fixed; they are not slope SEs or a joint covariance adjusted for estimating Person and structural parameters together. The local joint-curvature check used for portable GPCM JML scoring does not supply such an inferential covariance. Values fixed by anchors or identification constraints are not estimated: diagnostic tables mark them Fixed = TRUE and leave their sampling SEs and intervals missing, rather than assigning an observation-information SE.

Nuisance-parameter adjustment and estimation bias are separate issues. Haberman (2004, Sections 1.4-1.5) shows for a binary Rasch setting that, with fixed test length, JML can concentrate around a biased limit as the number of persons grows. Even a variance appropriate to that limit does not establish coverage of the true parameter. This result motivates checking test length separately from sample size; it does not validate a correction or an interval for sparse many-facet GPCM. Formal JML slope intervals are not currently available in mfrmr.

When comparing software, report response-score adjustment and post-fit bias correction separately, including how the correction defines exposure when responses are missing or unequal across Persons. Matching the label "JML" alone does not establish matching estimates or uncertainty.

Practical recommendation:

- For manuscript or operational reporting, choose the estimator from the inferential target and assumptions, and report the choice. MML integrates person measures under a specified population model and provides marginal observed-information SEs; consistency of its structural estimates is conditional on an adequate response model, population distribution, and regularity conditions.
- JML remains useful for a JMLE-oriented FACETS comparison, descriptive or exploratory work, and designs with substantial information per person. Report its incidental-parameter limitation and the exploratory basis of this package's JML structural SEs rather than treating estimator choice as a universal reporting rule.
- Native RSM/PCM CML conditions out Person parameters, requiring no ability population distribution. It supplies structural-parameter intervals under conditional-likelihood regularity; see "Conditional maximum likelihood". General GPCM inference uses MML. JML remains a point-estimation/FACETS comparison route, not a fixed-exposure truth-centered Wald procedure. Third-party CML fits can still be imported with `import_erm_fit()`.

MML integration workflow

Choose integration together with the result you need. The default remains `mml_integration = "fixed"`, `quad_points = 31`; this order does not certify accuracy. `mml_quadrature_sensitivity()` compares refits at specified orders. Review changes in estimates and uncertainty before reporting a fixed-grid fit.

For ordinary full direct RSM/PCM or one-family GPCM fitting, `"auto"` checks the fitted response patterns and refines integration when needed. Read `fit$config$mml_accuracy` and the resolved rule in `fit$config$estimation_control$mml_integration`. Downstream support depends on this resolved rule, which can be adaptive even when the initial rule was fixed. Check the intended output's model and integration scope before fitting:

- Ordinary fixed-N(0,1) RSM/PCM model and sandwich facet intervals require fixed integration, as do joint cell intervals requested with `interaction` and independent anchor-source propagation. See `mfrm_facet_intervals()`.
- Native-location intervals for estimated-normal RSM/PCM and one-family GPCM accept fixed or adaptive direct MML within their stated scope. RSM/PCM population-coefficient and residual-variance intervals use the same qualification; see `mfrm_population_intervals()`. GPCM latent-regression location intervals are unavailable.
- Automatic integration does not support EM/hybrid engines, checkpoints, comparison-only fitting or two slope families. ConQuest overlap export currently requires a resolved fixed rule.

Overall fit readiness, interval eligibility and coverage are different statements. A supported target-specific interval can have `CIEligible = TRUE` while its unchanged source fit has `InferenceReady = FALSE`. Inspect the interval's `Status`, `checks`, `numerical_checks` and cautions. Eligibility means the procedure's checks passed; it does not establish global identification, nominal finite-sample coverage or availability of other inference procedures. Automatic likelihood/gradient/posterior checks also do not replace the interval's full-information and finer-order review.

MML inference conditions

There is no ten-response-row minimum. Full RSM/PCM JML, native CML and fixed-population MML can accept one Person when their model-specific checks pass. One-Person JML/MML also requires `lpSolve` for a pre-fit additive recession check; a detected or unresolved boundary returns no finite estimate. Separate rating events must satisfy the assumed conditional independence; explicitly declared repeated ratings are not evidence that independence holds. The returned preparation note, interval cautions and summary explain the single-Person limitation. Any available model-information interval remains an unqualified normal approximation; one Person supplies neither independent- Person sandwich uncertainty nor empirical between-Person variation. Estimated-population MML, GPCM, corrected JML and comparison-only fitting retain their at-least-two-Person software scope. Data and anchor review can inspect one Person. The calibration interface still requires two observed score categories; these input checks do not establish identification or precision. Scoring one Person against saved calibration is a separate task; see `score_mfrm_calibration()`. Small supported datasets can return point estimates and asymptotic uncertainty when the model-specific checks pass. A singleton category count alone need not prevent RSM/PCM model-information intervals: `mfrm_facet_intervals()` performs a separate joint-information and quadrature review and retains the small-sample caution.

MML integrates the joint response vector for each Person under the declared ability distribution. Model-based structural SEs use the inverse full marginal observed information, including estimated population and other nuisance coordinates. Holding these coordinates fixed gives a different uncertainty calculation. EAP Person scores are predictions under the fitted model, not additional free parameters in this information matrix.

Regular likelihood inference requires the response and population models to be correct for the actual assignments and covariates, independent Persons, a fixed-dimensional identifiable model, a consistent interior solution, smooth derivatives and growing information in every free direction. This permits fixed exposure per Person under suitable designs; it does not establish consistency merely by increasing Person count. For an estimated normal population, zero variance is a boundary outside this regular argument. Numerical convergence or a positive local Hessian does not prove global identification, exclude a boundary solution or verify the population model.

Finite quadrature approximates the continuous-normal likelihood. Optimizer and integration errors must be small relative to sampling uncertainty; a fixed quadrature order does not automatically maintain this property as sample size grows. Comparing quadrature orders diagnoses sensitivity but does not provide a continuous-integral error bound. See `mfrm_facet_intervals()` for the assumptions, available checks and the joint delta method for contrasts divided by the same fit's estimated population SD.

All-minimum and all-maximum Persons

`fit_mfrm()` has no public option to remove Persons before fitting because all their responses are at the minimum or maximum. Both JML and MML retain those observed responses. Extreme-response flags use the usable rows after data preparation; missing responses do not count as intermediate scores. Declare `rating_min` and `rating_max` from the rubric so that an observed maximum is not mistaken for the intended scale maximum. Inspect `fit$facets$person$Extreme` for `"low"`, `"high"`, or `"none"`.

Ordinary JML. `fit_mfrm()` does not replace extreme response scores before JML fitting. For an independently free Person with all-minimum or all-maximum responses, the primary `Estimate` is `-Inf` or `Inf`; `OptimizerEstimate` retains the finite computational trace, not a finite Person MLE.

Fixed Person anchors retain their supplied values, and coupled constraints require their own boundary review. The Person audit's `BoundaryState = "has_exclusions"` identifies nonfinite parameters; it does not mean that these Persons or their input rows were deleted. Finite structural estimates alone do not establish a finite joint maximum.

Corrected JML. In the supported shared-owner GPCM route, extreme Persons also remain in the data and assignment-pattern accounting. Their profiled abilities have infinite limits and their structural estimating- equation contributions are zero at those limits. This boundary calculation is not an input filter or a general proof of bias removal.

MML. Extreme Persons contribute to the marginal likelihood through integration over the specified or estimated population distribution. Their reported abilities are posterior EAPs, with posterior SDs rather than frequentist Person-MLE standard errors. A proper normal population model with finite parameters gives finite EAPs, subject to valid numerical integration; an extreme response pattern alone does not require deletion. This does not guarantee convergence, identification or valid structural intervals for the fitted model. Removing these Persons would change the observed sample and marginal likelihood.

New-Person scoring with `predict_mfrm_units()` is a separate operation. With an eligible calibration and an admitted scoring prior, it can return finite EAPs for extreme new Persons, including after JML calibration. Those scores do not replace the original JML Person MLEs. Calibration-source checks and scoring-batch integration checks must both pass; increasing scoring nodes does not resolve a boundary in the source calibration. A finite display from `fair_average_table(..., xtreme = ...)`, or placement at the end of a Wright map, likewise does not change the fitted model or correct JML bias.

Model-estimated facet interactions

`facet_interactions` adds confirmatory fixed-effect interaction terms to the linear predictor. For example, `facet_interactions = "Rater:Criterion"` estimates a rater-by-criterion deviation matrix in the same likelihood as the main MFRM fit. The additive reference is

$$\eta_{nij} = \theta_n - \delta_j - \beta_i$$

and the interaction extension is

$$\eta_{nij} = \theta_n - \delta_j - \beta_i + \gamma_{ji}$$

where the interaction block is identified by zero marginal sums:

$$\sum_j \gamma_{ji} = 0, \quad \sum_i \gamma_{ji} = 0.$$

With J levels of the first facet and I levels of the second facet, this contributes $(J-1)(I-1)$ free parameters. Positive interaction estimates indicate scores higher than expected under the additive main-effects model for that facet-level combination; negative estimates indicate lower-than-expected scores.

This is a model-estimated interaction term, not the residual screening reported by `estimate_bias()` or `estimate_all_bias()`. In line with the MFRM bias-interaction literature, the facet pair should

be named explicitly before fitting. Exploratory use is possible, but should be reported as screening, with sparse-cell and multiplicity caveats. The current implementation is intentionally narrow: two-way non-person facet interactions for RSM and PCM only, estimated as fixed effects. GPCM interactions, person interactions, higher-order interactions, and random-effect facet interactions are deferred. This is ordered binary support, not a separate nominal-response model. In PCM, a binary fit still uses one threshold per step_facet level on the shared observed-score scale.

Supported model/estimation combinations:

- model = "RSM" with method = "MML", "JML"/"JMLE", or "CML"
- model = "PCM" with a designated step_facet; CML requires it explicitly
- Native CML returns a separate structural-parameter result with conditional-information intervals; ordinary fit/scoring/diagnostic methods below apply to MML/JML results
- facet_interactions with model = "RSM" or "PCM" for explicit two-way non-person facet interactions
- model = "GPCM" uses one positive slope family. MML permits separate slope/step owners; JML requires slope_facet == step_facet. Core summaries, fitted-object posterior scoring, compute_information(), Wright/pathway/CCC fit plots, diagnose_mfrm(), residual-PCA follow-up, interrater_agreement_table(), unexpected_response_table(), displacement_table(), measurable_summary_table(), rating_scale_table(), facet_quality_dashboard(), reporting_checklist() category_structure_report(), category_curves_report(), and graph/scorefile facets_output_file_bundle() routes are available with score-side caveats. Separate-owner MML supports slope and curve intervals, same-design bootstrap_mfrm_gpcm(), and matched PCM comparison. build_weighting_review() and the following simulation/design workflows still require the same slope/step owner. Direct simulation specifications and data generation are also supported through build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data() when the slope-aware generator contract is stored explicitly; direct recovery checks are available through simulate_mfrm_recovery() and assess_mfrm_recovery(). Slope-aware fair_average_table() and estimate_bias() are available with their documented caveats. Role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers are available as caveated sensitivity evidence. Full FACETS-style score-side contract review, posterior predictive checks, and MCMC estimation are not available for GPCM. Use gpcm_capability_matrix() as the formal boundary statement for the current GPCM scope.

Latent-regression status:

- population_formula = NULL keeps the standard unconditional behavior for RSM/PCM and JML. For GPCM MML, the default gpcm_mml_identification = "free_population" constructs an intercept-only population model internally; use "fixed_standard_normal" only to reproduce the legacy restricted likelihood.
- Supplying population_formula activates latent regression for method = "MML" only.
- This implementation assumes a one-dimensional conditional-normal population model with person-specific quadrature nodes $\theta_{nq} = x_n^\top \beta + \sigma z_q$.
- Background variables must be supplied in person_data; numeric/logical columns and categorical factor/character columns are expanded through stats::model.matrix().
- Documented overlap with the ConQuest latent-regression model is limited to direct estimation from response data under a unidimensional MML population model with package-built model-matrix covariates. It should not be described as numerical equivalence for arbitrary imported design matrices, multidimensional models, or the full ConQuest plausible-values workflow.

- `predict_mfrm_units()` and `sample_mfrm_plausible_values()` can score latent-regression fits under the fitted population model, but they require one-row-per-person background data for scored units when the fitted population model includes covariates. Intercept-only latent-regression fits (`population_formula = ~ 1`) can reconstruct that minimal person table internally during scoring.

Latent-regression workflow

For an initial latent-regression run, keep the setup explicit:

1. Put response data in `data`, with one row per rating event.
2. Put background variables in `person_data`, with exactly one row per person. The ID column must match `person`, or be supplied through `person_id`.
3. Use `method = "MML"` and a one-sided formula such as `population_formula = ~ Grade + Group`.
4. Numeric/logical and factor/character predictors are expanded with `stats::model.matrix()`. After fitting, inspect `summary(fit)$population_coding` to see the fitted levels, contrasts, and encoded design columns that will be reused for scoring/replay.
5. Start with `population_policy = "error"` while preparing data. Use `"omit"` only when complete-case removal is intended, and then inspect `summary(fit)$population_overview` and `summary(fit)$caveats` before reporting results.
6. Report `summary(fit)$population_coefficients` as coefficients of the conditional-normal latent population model, not as a post hoc regression on EAP or MLE scores.

With covariates, the estimated `sigma2` is the residual variance of ability conditional on those covariates, not the marginal population variance. Marginal variance also depends on the distribution of the covariates. Standardization and polynomial/spline bases learned during fitting are reused for new persons; changing the scoring cohort does not redefine them. For custom transformations, supply a prediction-aware R transformation or precompute predictors with fixed training constants. Older saved fits can reconstruct transformed terms only from retained person data that reproduce the training design; otherwise scoring requests a new calibration fit.

For an intercept-only model, `population_formula = ~ 1` estimates a single population mean and variance by default. In supported RSM/PCM direct MML, `population_sd = 1` instead fixes the SD and estimates only the mean. Training still requires a one-row-per-person ID table in `person_data`. Adjusting these two moments retains a normal population shape; it does not learn skewness or establish that the training population represents the people who will receive scores. New-Person intervals condition on the estimated calibration and population parameters and exclude uncertainty from estimating them. Current population-model scoring remains a review workflow; see `predict_mfrm_units()`.

Latent-regression standard-error caveat

`summary(fit)$population_coefficients` reports point estimates of $\hat{\beta}$; `population_overview` reports the estimated population variance. These tables do **not** provide standard errors, confidence intervals, or asymptotic z / Wald statistics for the population parameters. For an estimated normal RSM/PCM population, `mfrm_population_intervals()` separately requests experimental pointwise coefficient, prespecified coefficient-contrast and log-variance intervals after full joint numerical checks. Its result has a `vcov()` method on coefficient/log-variance coordinates; the ordinary fit summary does not acquire automatic intervals. The internal MML observed-information calculation

includes $(\boldsymbol{\beta}, \log \sigma^2)$ when computing joint covariance for structural-parameter SEs. With explicit `population_sd`, variance is fixed and its log-variance coordinate is excluded from that covariance; coefficient and structural uncertainty are conditional on the supplied SD. Population intervals for GPCM remain unavailable through this new helper. Sampling coverage is not established by numerical checks. Treat the population summary tables as descriptive point estimates; use the separately saved interval result when reporting the experimental approximation and its limitations.

The default latent-regression identification uses `noncenter_facet = "Person"`. Without anchors, this sum-to-zero-centers all non-Person facets. With anchors, `centering_scope = "facet"` includes their contributions in each facet's zero sum; `"remainder"` retains the earlier restriction on the unconstrained remainder. The population model is never changed implicitly to accommodate anchors. The restricted anchor-origin alternative below uses an explicitly fixed population SD.

Anchor origin and cohort means

When an old rater panel is anchored, leaving Rater uncentered allows new raters to have a nonzero mean severity. Fixing the Person mean to zero can nevertheless distort estimates when the receiving cohort differs. For RSM/PCM direct MML, `population_formula = ~ 1` with `population_sd = 1` estimates the receiving mean on an explicitly fixed-SD normal population. Set `noncenter_facet = "Rater"` and supply a retained direct Rater anchor. An anchor on another facet does not resolve the mean/Rater alias. The complete constrained design must still have full rank; one anchor is sufficient only in an otherwise identified design. Group anchors, dummy facets, population covariates and interactions are outside this alternative.

For example, begin with the bundled ratings and an externally supplied anchor value (0.3 below is illustrative, not a validated calibration):

```
ratings <- load_mfrmr_data("example_core")
fixed_rater <- data.frame(Facet = "Rater", Level = "R01", Anchor = 0.3)
cohort_fit <- fit_mfrm(
  ratings, person = "Person", facets = c("Rater", "Criterion"), score = "Score",
  model = "RSM", method = "MML", anchors = fixed_rater,
  noncenter_facet = "Rater", population_formula = ~ 1,
  person_data = unique(ratings["Person"]), population_sd = 1
)
summary(cohort_fit)$population_coefficients # Estimated receiving-cohort mean
summary(cohort_fit)$population_overview     # Fixed versus estimated variance
```

Use `mfrm_population_intervals()` for the mean and `mfrm_facet_intervals()` for rater uncertainty after their independent numerical checks. The fixed variance remains visible without an inferential CI. Covariance includes all free calibration coordinates and the population mean, but excludes uncertainty in the supplied anchors and SD. No general coverage guarantee follows from agreement with another program.

Other facets retain their declared centering. Rater anchors do not establish item invariance: a changed item pool or its difficulty distribution can change the mean's substantive interpretation. Report the anchored panel, item/criterion reference, fixed SD, and assumed stability across cohorts.

Anchors and scale constraints

Anchor inputs are optional and have distinct roles:

- anchors contains facet/level/fixed-value information and imposes direct equality constraints on selected parameters.
- group_anchors contains facet/level/group/group-value information and constrains each declared group mean. Its interpretation is conditional on the externally justified target or equal-mean assumption.
- Common Persons, raters, items, or rating events are properties of the observed design. Neither constraint type creates empirical overlap or proves that disconnected subsets are substantively comparable. Both are normalized internally, so column names can be flexible (facet, level, anchor, group, groupvalue, etc.).

Direct anchors fix facet locations, not thresholds or GPCM slopes. Values must use the source model's internal metric and orientation, not transformed display units. make_anchor_table() exports this metric. For reuse of a complete fixed calibration, see extract_mfrm_calibration() and score_mfrm_calibration().

With the default noncenter_facet = "Person", a partial Rater anchor such as R01 = 0.7 makes the other Rater levels sum to -0.7 under the default centering_scope = "facet". With one remaining level, it is fixed at -0.7. Group means contribute their target multiplied by the group's size, including directly fixed members. If anchors/groups cover all levels, their total must agree with the facet's zero sum. Contradictions stop review and fitting even with anchor_policy = "silent".

Use centering_scope = "remainder" to reproduce earlier partial anchoring: the ungrouped, unanchored levels sum to zero and a singleton is fixed at zero. Saved results retain their original constraints; reloading does not migrate estimates. Changing conventions requires refitting.

For ordinary additive RSM/PCM JML, centering_scope = "anchor_origin" implements the FACETS convention that a directly or group-anchored facet is not additionally centered. With a Person anchor, choose an unanchored facet such as noncenter_facet = "Rater"; other unanchored facets remain centered. A Rater anchor can instead retain the default noncenter Person facet. The noncenter facet must be active and have neither direct nor group anchors. A single direct anchor or group mean supplies its facet's origin; multiple fixed values can impose additional substantive constraints. Without Person group constraints, if all retained responses of the directly anchored Persons are at the same declared minimum or maximum, fitting stops before optimization, without requiring lpSolve: free Person and noncenter-facet locations can diverge together while preserving every fixed value. FACETS 4.5.0 can report finite values for such inputs, but these are not an ordinary finite JML solution and are not reproduced by this route. An extreme anchor does not alone trigger this rule when another Person anchor supplies interior or opposite-boundary responses. Other detected joint additive recession directions affecting calibration also stop this route; passing the checks does not certify every boundary configuration. General extreme-score exclusion and every FACETS option are not equivalent. See the FACETS centering specification. Match category support, signs, the response design and population assumptions for TAM/ConQuest comparisons; an anchor value alone does not establish equivalence. All these routes condition on supplied anchor values and do not propagate error from an external estimated calibration.

If an old zero-sum panel is fully anchored and new raters are added, whole-facet centering also forces the new raters to sum to zero. This is an additional constraint, not evidence that the new panel has the same average severity. In supported fixed-population MML models, choosing noncenter_facet = "Rater" can leave that facet uncentered. This is not a general remedy: only one facet can be noncentered. The RSM/PCM intercept-only, fixed-SD alternative is described above. Other latent regression, including the usual one-family GPCM population model, requires noncenter_facet =

"Person". Review the complete identification and population assumptions before using anchors to compare cohorts.

A group target is an unweighted arithmetic mean of its declared levels, including any directly anchored members. It is not weighted by rating counts. FACETS G supplies per-element contributions to a group total; convert their arithmetic mean to one repeated GroupValue. For example, contributions 0.2 and 1.0 become GroupValue = 0.6 for both members. Unequal GroupValue entries are conflicting targets, not contributions; use anchor_policy = "error" to stop on review issues. FACETS group 0 denotes direct anchors and must be translated to anchors; mfrmr treats Group = "0" as an ordinary group identifier. The FACETS X convention that removes extreme members is not applied automatically. See FACETS element specifications.

Group-origin support retains constrained maximum likelihood, not every native FACETS update. Matched interior RSM/PCM examples agree when a single group only sets the origin. With additional restrictions, FACETS 4.5.0 can satisfy the group means while leaving a nonzero likelihood derivative along a constraint-preserving direction. mfrmr optimizes that likelihood rather than reproducing those numerical differences. Matching means alone does not establish parameter, SE or interval equivalence; group-anchor extreme extrapolation remains unsupported. In MML, individual abilities are integrated out: Person direct and group anchors are unsupported and rejected even with anchor_policy = "silent". Sharing Persons across raters requires no fixed Person values.

Ordinary fit summaries condition on supplied anchor values. Independent source-anchor covariance can be propagated explicitly in the bounded fixed-SD RSM/PCM route through mfrm_facet_intervals() and mfrm_population_intervals(); it is not added automatically. By default, downstream SEs and intervals condition on those values and retain the selected estimator's other limitations. A missing SE for a fixed value does not mean its source calibration was error-free. See mfrmr_linking_and_dff for the theoretical background and a loading-first example, and review_mfrm_anchors() for receiving-data checks.

Anchor review behavior:

- fit_mfrm() automatically runs an anchor review.
- invalid rows are removed before estimation.
- duplicate rows keep the last occurrence for each key.
- anchor_policy controls whether detected issues are warned, treated as errors, or kept silent.
- the review checks table/data compatibility and local support counts; it does not establish source-fit readiness, cross-run identity, parameter invariance, or the validity of a group-mean assumption.

Facet sign orientation:

- facets listed in positive_facets are treated as +1
- all other facets are treated as -1 This affects interpretation of reported facet measures.

Estimator-specific checks before fitting

Before optimization, mfrmr builds a sparse adjacent-category-logit design in the same constrained free coordinates used by the optimizer. The check includes Person coordinates for JML, integrates them out for MML, and includes facet anchors, group constraints, signs, supported two-way interactions, and RSM/PCM step coordinates.

An exactly rank-deficient design stops with a structured `mfrmr_estimability_error`; its `estimability` field records rank, nullity, parameter blocks, tolerance checks, and a bounded null-direction explanation. Optimization is not run. A full-rank MML fixed-effect design whose corresponding free-Person JML design is rank deficient returns a fit with an `mfrmr_estimability_warning`: its cross-panel contrasts rely on the common latent-population assumption and remain review-only.

Inspect `fit$data_review$estimability`. RSM and PCM use the full linear free-coordinate check. For GPCM and an active latent-regression residual variance, the additive block is audited before fitting. A retained vector also records the analytic free-to-expanded log/natural-scale transformation Jacobians and a central-difference check in `fit$data_review$estimability$nonlinear_transformation`. This verifies the parameterization only; it is not a response-likelihood Jacobian or a structural-identification result. A stationary retained solution of modest free dimension also receives a local observed-information Hessian and a recorded eigenvalue-tolerance ladder in `fit$data_review$estimability$fitted_inf` Nonstationary or larger fits retain an explicit not-evaluated status. This fitted-information layer is diagnostic only: it does not yet classify weak information, make the nonlinear check complete, or turn full additive rank into a full-model estimability claim. Eligible nonlinear MML fits also receive bounded observed-pattern and all-response-pattern score checks. The latter operates on each Person's retained observation design under unit row weights and records probability-normalization, zero-expected-score, expected-information, and selected numerical-derivative summaries. Missing rows are not imputed; nonunit weights and excessive pattern grids retain explicit not-evaluated states. Mathematically identical Person observation designs are evaluated once and reconstructed by exact multiplicity; active latent-regression covariate rows are part of this identity. Only conceptual and evaluated workload summaries are retained. These retained-point diagnostics do not by themselves establish global structural identification, weak-information status, or readiness.

`fit$data_review$estimability$nonlinear_local_estimability` interprets only the first-order local rank that these maps support. For JML GPCM, full column rank of the complete conditional adjacent-logit Jacobian is a sufficient retained-point local certificate. For fixed-quadrature MML with unit row weights and finite parameters, the Person-specific observed- pattern score vectors are part of the positive finite response-pattern support. If those vectors span every optimizer free coordinate, the full expected score information is positive definite; exhaustive enumeration is unnecessary for this sufficient direction. A rank-deficient observed subset is inconclusive and is classified only when the all-pattern enumeration is available. The record explicitly leaves continuous-integral and global identification, boundary status, weak information, and inference readiness unclassified.

Choosing maxit without result-driven tuning

Treat `maxit` as a predeclared computational budget, not as a value to tune until preferred estimates appear.

1. Choose the model, estimation method, optimizer, tolerance, quadrature rule, and initial `maxit` before examining coefficient or fit results. The default `maxit = 400` is the package starting point for an analysis.
2. Inspect `FitReadiness` and `InferenceReady`, together with the eligibility and reasons for the specific calculation. RSM/PCM model intervals can pass their own singleton-category and information checks while the fit retains its category review; see `mfrm_facet_intervals()`. This exception does not waive numerical convergence. Also inspect purpose-specific Design, Stability, Diagnostics, and Reporting workflow rows. Optimizer code zero alone is insufficient. A missing, nonfinite or incomplete terminal gradient produces `ConvergenceReason = "code_zero_unavailable_gradient"` and review status, even when the optimizer reports

code zero. Resolve the numerical evaluation before using the fit for inference; more iterations alone do not establish a valid gradient.

3. If `ConvergenceStatus == "iteration_limit"`, keep that fit review-only. Refit the same data, model, method, anchors, optimizer, tolerance, and quadrature rule with the next ceiling in a prespecified sequence, such as 400, 800, then 1600. Do not choose among runs by coefficient size, statistical significance, fit statistics, or agreement with an expected answer.
4. The first run in that sequence that satisfies the numerical-readiness criteria becomes eligible for interpretation. If separately ready runs differ materially, treat the difference as numerical instability and review the model, identification, data support, and optimizer rather than selecting the preferred result.
5. Report the requested `maxit`, actual iteration/evaluation counts, convergence status and reason, optimizer, terminal gradient, and any polishing stages. These are retained in `fit$summary` and `fit$opt$optimizer_polish`.

This rule applies to both JML and MML. JML can require a larger computation budget because it estimates one fixed effect per person; increasing `maxit` does not make JML equivalent to MML and must not be used to switch the estimand after seeing results.

Performance tips

When JML is the prespecified estimand, it is often faster than MML but may require a larger `maxit` on larger datasets. Do not switch from MML to JML only to shorten runtime: integrating over a population distribution and treating person parameters as fixed effects are different analysis choices.

For MML runs, `quad_points` is the main accuracy/speed trade-off. The `@param quad_points` tier table is the authoritative reference; in short:

- `quad_points = 7` is a lightweight screening setting; do not use its IC values for automatic model selection.
- `quad_points = 15` is an intermediate review option when runtime matters; automatic IC ranking remains disabled.
- `quad_points = 31` is the package default and a starting point, not a guarantee that integration error is negligible.
- `quad_points = 61` (or higher) supplies candidate denser grids at additional computational cost; no fixed order is sufficient for every response pattern.
- Use `mml_quadrature_sensitivity()` to inspect same-data movement without an automatic stable/unstable decision.
- `mml_engine = "direct"` remains the most stable general-purpose path.
- `mml_engine = "em"` or `"hybrid"` currently target `RSM` / `PCM` fits without a latent-regression population model.
- Benchmark your own workload before using `mml_engine = "em"` or `"hybrid"` for final reporting; `direct` remains the documented default when you have not compared engines for your data.
- When a direct code-zero stage stops ahead of the terminal-gradient check, bounded polishing is automatic. Inspect `fit$opt$optimizer_polish$Stages` rather than repeatedly lowering `reltol` without reviewing the retained objective, gradient, and parameter changes.

Downstream diagnostics can also be staged:

- use `diagnose_mfrm(fit, residual_pca = "none")` for a quick first pass
- add residual PCA only when you need exploratory residual-structure evidence

Downstream diagnostics report `ModelSE` / `RealSE` columns and related reliability indices. For `MML`, non-person facet `ModelSE` values are based on the observed information of the marginal log-likelihood and person rows use posterior SDs from EAP scoring. For `JML`, these quantities remain exploratory approximations and should not be treated as equally formal.

For `GPCM`, residual-based mean-square fit screens are also best treated as exploratory diagnostics rather than strict Rasch-style invariance tests, because the discrimination parameter is free.

Interpreting output

Start with `summary(fit)` and the estimator selected above. For ordinary MML/JML results, read its `$decision` before interpreting the Person and facet overviews. For CML, read `$locations` and `$steps`, or `$comparisons` for a comparison-only fit; those results do not contain Person abilities. A reported estimate, numerical convergence and an eligible confidence interval are different claims. Read the associated checks and limitations.

Typical workflow

1. Fit the prespecified model and inspect `summary(fit)`.
2. For ordinary MML/JML RSM/PCM, collect `res <- mfrm_results(fit)` and inspect `summary(res)$triage` and `$next_actions`. Pass already computed diagnostics through `diagnostics = diagnostics` to reuse them.
3. Choose `plot(res, ...)`, `mfrm_report()`, `export_mfrm_results()` or `saveRDS(res, file)` according to the output you need. See mfrmr_reports_and_tables for the complete example.
4. For CML, use `mfrm_facet_intervals()` and its own summary, plot, `apa_table()` and RDS saving. For GPCM or extended models, use `gpcm_capability_matrix()` and `mfrm_results()` to select supported outputs. Shared function names do not make ordinary diagnostics or scoring available for every estimator or result class.

Information-criterion contract

For an eligible fixed-facet `MML` fit, let `D = -2 * LogLik`, let `k` be `Npar` (the retained free optimization-vector dimension after constraints), and let `N_person` be the number of unique prepared Persons. The canonical panel is `AIC = D + 2 * k`, `BIC = D + log(N_person) * k`, and `SABIC = D + log((N_person + 2) / 24) * k`.

`ResponseRows`, `WeightedResponseTotal`, `Persons`, and `ICSampleSize` are separate fields. The compatibility field `N` retains its earlier response-row or summed-observation-weight meaning and is not the BIC sample size. Explicit all-unit weights remain eligible; every non-unit observation-weight fit, JML fit, and object without the current contract identity is excluded from the common MML panel. Its canonical `AIC`/`BIC`/`SABIC` fields are `NA`, while any retained raw values are explicitly named `LegacyAIC` and `LegacyBIC`. At 22 or fewer Persons, SABIC is displayed only as sensitivity evidence and `SABICSelectable = FALSE`.

Integration adequacy is recorded separately from formula eligibility. `ICIntegrationTier` is `"coarse_screening"` below 15 points, `"intermediate_review"` at 15–30, `"standard_start"` at 31–60, and `"dense_sensitivity"`

at 61 or more. Raw canonical criteria remain visible in every eligible MML tier, but `ICSelectable = FALSE` below 31 points; automatic deltas, criterion weights, preferences, and LRT are suppressed. A close or consequential q>=31 comparison should still be reevaluated on a denser common grid.

References

The ordered-category many-facet formulation follows Linacre (1989), with the `RSM` and `PCM` branches grounded in Andrich (1978) and Masters (1982). The `GPCM` branch follows the generalized partial credit formulation of Muraki (1992) under a package-specific positive log-slope identification convention. The `MML` route follows the quadrature-based marginal-likelihood framework of Bock and Aitkin (1981).

- Akaike, H. (1974). *A new look at the statistical model identification*. IEEE Transactions on Automatic Control, 19(6), 716-723.
- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Dhaene, G., & Weidner, M. (2023). *Approximate functional differencing*. SERIEs, 14, 379-416. doi:10.1007/s13209023002831.
- Haberman, S. J. (2004). *Joint and conditional maximum likelihood estimation for the Rasch model for binary responses*. ETS Research Report RR-04-20. doi:10.1002/j.23338504.2004.tb01947.x.
- Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement*, 4(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement*, 5(2), 189-227.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176.
- Uto, M., & Ueno, M. (2020). *A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo*. Behaviormetrika, 47, 469-496.
- Robitzsch, A., & Steinfeld, J. (2018). *Item response models for human ratings: Overview, estimation methods, and implementation in R*. Psychological Test and Assessment Modeling, 60(1), 101-139.
- Schwarz, G. (1978). *Estimating the dimension of a model*. Annals of Statistics, 6(2), 461-464.
- Sclove, S. L. (1987). *Application of model-selection criteria to some problems in multivariate analysis*. Psychometrika, 52(3), 333-343.

Conditional maximum likelihood

Set `method = "CML"` and `category_policy = "preserve"` for native many-facet RSM/PCM estimation. Declare the rubric range with `rating_min` and `rating_max`; PCM requires `step_facet`.

Non-person facets and category steps have the ordinary sum-to-zero constraints. Supported two-way facet interactions keep zero marginal sums. Unit observation weights are required. The default `fit_scope = "full"` estimates the full identified structural model; the comparison-only route below reports selected contrasts in deficient models. The suggested package `lpSolve` supplies the finite-MLE check. Repeated Person-by-facet cells require correction or an explicit justified `repeated_ratings = "independent"` declaration before CML intervals are supplied.

For Person p, let $s_p = \sum_j y_{pj}$ and let x_{pjk} be the cumulative structural design vector for category k, with $x_{pj0} = 0$. Conditioning on the observed roster and s_p gives

$$\ell_c(\beta) = \sum_{p,j} x_{pj,y_{pj}}^T \beta - \sum_p \log[t^{s_p}] \prod_j \sum_{k=0}^{K} \exp(x_{pjk}^T \beta) t^k.$$

A generalized ESF recursion computes the coefficients in log scale, conditional sufficient-statistic means and covariances. The score is observed minus conditional expected statistics; information is their conditional covariance. It contains cross-information for all estimated structural coordinates. No ability quadrature or ability population model is used.

Person-constant effects disappear on conditioning. The information at zero checks identification for the actual rosters and totals; an unidentified or numerically rank-sensitive full model is refused. A dynamic-program graph represents conditional support for a linear-program recession check without enumerating response patterns. A detected or inconclusive recession check, or failed stationarity check prevents delivery of a full fit. If those checks pass but the fitted information cannot be evaluated or safely inverted, the finite numerical point estimates and conditional likelihood are retained with a warning. `checks$information_status` is `"unavailable"` and `checks$information_reason` explains why; the joint covariance is an `NA` matrix, not a regularized substitute. Non-fixed targets have missing SE and bounds, `CIEligible = FALSE` and an `InferenceReview` reason in summaries and facet intervals. Small score residuals alone do not certify accurate points in weak directions; examine the information warning before interpreting them. Inverse information, when available, is unregularized. Extreme-total Persons have zero conditional likelihood contribution and remain in `person_totals`.

The full-scope result is an `mfrm_cml_fit`. `summary(fit)` contains `locations`, `steps`, `interactions` and checks; these tables use pointwise 95 percent normal intervals. `mfrm_facet_intervals()` accepts locations or named contrasts at another confidence level, with `method = "model"`. Plot the interval object with `plot(ci)` and retain it with `saveRDS()`. `vcov(fit)` is the covariance in free structural coordinates, or an `NA` matrix with an `unavailable_reason` attribute when information fails. Default `confint(fit)` retains missing bounds and reasons in its `diagnostics` attribute. `logLik(fit)` is the conditional log likelihood. Conditional and marginal/joint likelihood values are different objectives.

The conditional model must hold for the actual rating roster: conditional on ability and the roster, ratings follow the specified independent RSM/PCM response law. A prespecified incomplete assignment is compatible with this assumption. Response-dependent assignment, stopping or missingness requires a separate justification that the conditional law is preserved. Conditioning alone does not supply that justification or repair local dependence.

One sufficient regular regime has independent Persons, fixed structural dimension and category range, bounded ratings per Person, a finite true parameter, and conditional information $I_N(\beta_0)/N$ converging to a positive-definite matrix. For example, the proportions of roster/total strata can stabilize while their combined support identifies every free direction. Then CML is consistent as Person count N grows, even with fixed ratings per Person, and inverse-information Wald intervals

are asymptotically valid. Increasing the total number of Persons alone is insufficient: extreme totals add no information, and a direction supported by only a few Persons can remain imprecise. Growing numbers of facet levels/parameters require a separate asymptotic argument. Numerical checks on one fit do not establish these sampling conditions or guarantee finite-sample coverage. It does not estimate abilities, population means or a population SD. In particular, it cannot supply a contrast divided by that fit's estimated population SD; the MML delta-method route remains separate.

Anchors, population regression, shrinkage, corrected JML and GPCM are outside this CML route. Full-scope CML also excludes dummy/uncentered facets; the comparison route uses the review's zero origin main facet. Unsupported explicit arguments are rejected even when equal to a default. Ordinary `mfrm_fit` scoring, diagnostics and consolidated reports are not attached to this separate class. These restrictions do not affect imported eRm fits. See Mair and Hatzinger (2007), doi:10.18637/jss.v020.i09, for the linear Rasch-family CML framework.

Comparison-only estimation

An incomplete assignment can identify a contrast even when the full model has aliased parameters. Set `fit_scope = "comparisons"` and supply an explicit `comparison_review` to estimate such contrasts. This route supports the review's unadjusted RSM/PCM models, unit-weight retained ratings, preserved categories, one facet-by-origin interaction, zero origin main effect and centered other facets/interaction margins. JML has free Person locations; MML uses the declared fixed normal or normal population regression model. CML conditions out Person abilities and requires no population distribution. PCM requires an explicit step owner. Repeated ratings retain their declared treatment and every response; a repeated row is not another independent Person.

The retained responses, score map and actual model constraints must match the review. Its map is rebuilt. For JML/MML, a column-scaled SVD removes only null directions of the conditional adjacent-logit map. Every observed probability and invariant contrast is preserved; no substantive effect is fixed to zero to identify a previously unidentified target. For MML the transformation preserves the conditional law at every latent node and leaves the variance coordinate free. It therefore preserves the marginal likelihood, but does not prove global marginal identification. The bounded review requires `nrow(A) * ncol(A)` and `ncol(A)^2` no greater than two million, where `A` is the conditional adjacent-logit design matrix. Numerically sensitive or unevaluated maps yield unavailable estimates.

JML first checks the observed-category recession cone using the optional `lpSolve` package. A detected recession or an inconclusive/unavailable check withholds all estimates. This route does not deliver extended-MLE contrasts when some parameters diverge. Unsupported category information also blocks estimates. Numerical acceptance requires optimizer convergence, full native gradient sup norm at most `max(1e-4, 10 * reltol)` and reduced observed information eigenvalues with minimum/maximum ratio greater than `1e-8` (the maximum is floored at one). No covariance is inferred from this check. MML population fits use three deterministic starting variances, requiring all starts to converge, target agreement within `1e-4` logits and objective agreement per Person within `1e-6`. Fixed-population MML has one start. MML also checks `2 * quad_points + 1` nodes at the retained point, requiring the same gradient tolerance and objective change per Person at most `1e-6`. These numerical criteria are not sampling-precision guarantees or a search proof of the global maximum. There is no automatic finer-grid refit.

`interaction_effect_table(fit)` and `summary(fit)` return the declared comparisons, retaining unidentified and failed rows with `Estimate = NA` and `EstimateReason`. Available JML/MML estimates are `descriptive_only`, in logits. Positive MML conditional ranks keep primary `State =`

"not_evaluated". JML/MML SEs and interval endpoints remain missing; FormalInferenceEligible is false. Save the full object or table as RDS to retain weights, empty cells, assumptions and checks. CSV retains visible columns only.

For CML, declare method = "CML" in the prospective review. Identification uses differences between every possible response pattern at the observed Person totals and rosters, constructed by a dynamic program. A target is identifiable only if its coefficient vector is in that row space. The same dense size bound applies to the support-difference matrix and intermediate state vectors. Extreme totals add no information. The zero origin main facet is coding for an effect eliminated by conditioning, not an equal-ability population assumption. FreePersonState remains a separate response-map audit.

CML optimizes the conditional likelihood in the identified subspace. This preserves probabilities given the Person total, not arbitrary unconditional category probabilities. If $\beta = T\gamma + v$, where v is a conditional null direction, an identified target $C\beta$ satisfies $Cv = 0$. Its covariance is

$$CT(T^T I_N T)^{-1} T^T C^T.$$

This expression is invariant to the choice of basis T for the identified coordinates and includes covariance with all estimable nuisance directions. The asymptotic argument above applies in that subspace when it stabilizes and its normalized information has a positive-definite limit. Numerical identification in one sample does not establish those conditions. It checks finite-MLE existence on the complete conditional support and refuses all estimates if any recession is detected or cannot be checked; extended-MLE target inference is outside scope. Stationarity and unregularized information checks match full-scope CML. If all checks pass, identified targets have EstimateStatus = "conditional_wald", FormalInferenceEligible = TRUE, SE, and pointwise 95 percent Lower/Upper limits. If only information fails, identified finite numerical points instead have EstimateStatus = "conditional_point", FormalInferenceEligible = FALSE, missing SE/bounds and an EstimateReason explaining the failure. Unidentified, boundary or unqualified-stationarity targets do not acquire point estimates. $covariance retains available joint covariance; unidentified or failed covariance entries remain NA. The result also retains person_totals. These are asymptotic intervals under the CML assumptions above, not a finite-sample coverage guarantee or a calibrated DFF classification. CML comparison fits estimate no population SD and cannot standardize by one. Use mfrm_facet_intervals(fit, level = .95) to reuse their saved targets through the existing interval plots and APA tables, preserving unavailable rows; changing the confidence level does not refit the model. Population and quadrature arguments are rejected for CML.

The result is an mfrm_comparison_fit, not an ordinary mfrm_fit. Individual cell estimates, Person scoring, ordinary fit diagnostics and prediction are not exposed through this class. representation and optimization retain numerical coordinates for auditing only; their representatives are not identified parameter estimates. Additional anchors, shrinkage, corrected JML, GPCM, alternate MML engines/integration, checkpoints and attached diagnostics are outside this route. Explicit arguments outside data preparation, the reviewed model and basic optimizer/quadrature controls are rejected, including unsupported arguments set to their default values.

The estimable-function criterion is discussed by Lenth (2015), https://journal.r-project.org/articles/RJ-2015-016/. Recession directions and finite-MLE existence in exponential families are distinguished by Geyer (2009), doi:10.1214/08EJS349. These principles do not establish global MML identification or finite-sample inference here.

See Also

mfrm_fit_extractors, mfrm_fit_responses, diagnose_mfrm(), estimate_bias(), build_apa_outputs(), gpcm_capability_matrix, mfrmr_workflow_methods, mfrmr_reporting_and_apa

Examples

```
# Load the package
library(mfrmr)

# Load example ratings and look at the first six rows
ratings <- load_mfrmr_data("example_operational")
head(ratings)

# Fit the model
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

# Plot the results (Wright map)
plot(fit)

# Save the summary, then display its tables
results <- summary(fit)
results$person_overview # One row summarizing person ability estimates
results$facet_overview  # One row per facet: number of levels, mean, SD, range

# Check the interpretation status and recommended next step
results$decision

## Not run:
# Optional repeated fitting: compare specified starting values.
reviewed <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  mml_starts = c(0, 0.5, -0.5))
summary(reviewed)$starting_values # Every attempt, including warnings/failures
reviewed$opt$start_review$parameter_map # Meaning and constraints of coordinates
write.csv(summary(reviewed)$starting_values, "starting-values.csv", row.names = FALSE)

## End(Not run)
```

---

fit_mfrm_imputed *Fit the same MFRM to every completed rating data set*

Description

Fit a separate MFRM to each completed version of the ratings reviewed by review_mfrm_imputations(). All fits use the same model and measurement scale so eligible estimates can be combined with pool_mfrm_imputed(). Failed fits and their messages are retained for review.

Usage

```
fit_mfrm_imputed(x, model = c("RSM", "PCM"), step_facet = NULL, ...)

## S3 method for class 'mfrm_imputed_fits'
print(x, ...)

## S3 method for class 'mfrm_imputed_fits'
summary(object, ...)
```

Arguments

x	An review_mfrm_imputations() object.
model	"RSM" or "PCM".
step_facet	Required for PCM: the facet with separate step parameters.
...	Shared fit_mfrm() arguments, such as quad_points, anchors, facet_interactions or optimizer controls. They must be named. Data columns, category scale and MML identification are set by this workflow. Observation weights, latent regression, shrinkage, checkpointing and adaptive integration are not supported by this workflow.
object	An object returned by fit_mfrm_imputed().

Details

Every fit uses the same original category ladder, facet levels, anchors and other supplied constraints, with fixed-standard-normal person MML identification. keep_original = TRUE prevents per-completion category collapsing. An unsupported category contrast, insufficient observed information or failed optimization is retained as a failed or ineligible analysis; subsequent pooling requires every imputation to qualify.

A common coordinate system does not establish model adequacy. Review the imputation model, rating design and numerical integration. The fit objects retain conditional person scores for individual review; those EAPs and posterior SDs are not ordinary complete-data parameter estimates and standard errors for Rubin pooling.

Value

An mfrm_imputed_fits object with the imputations review, all fits (including NULL for failures), analysis_summary recording every fit's status, error and warnings, and common settings. Use pool_mfrm_imputed() for eligible non-person facet targets.

See Also

review_mfrm_imputations(), pool_mfrm_imputed()

`fit_mfrm_random_rater` *Fit a rating-scale model with shared random rater severity*

Description

Estimate rater severity when the raters are modeled as draws from a specified population. Each rater has one effect shared across everyone they rate. The rating-scale model (RSM) uses approximate marginal maximum likelihood, with optional additive fixed facets. For a local effect within one person's performance instead, see `fit_mfrm_testlet()`.

Usage

```
fit_mfrm_random_rater(
  data,
  person,
  rater,
  score,
  facets = character(),
  score_levels,
  rater_sd = NULL,
  quad_points = 31L,
  maxit = 300L,
  missing = c("fail", "omit"),
  person_sd = NULL,
  person_variance_max = 16
)

## S3 method for class 'mfrm_random_rater'
plot(
  x,
  draw = TRUE,
  style = c("interval", "precision", "distribution"),
  sort = c("input", "estimate", "uncertainty"),
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  reference = 0,
  text_scale = 1,
  point_size = 2.5,
  intervals = c("none", "normal"),
  ...
)

## S3 method for class 'mfrm_random_rater'
```

```
summary(object, ..., calibration_intervals = c("none", "normal"), level = 0.95)

## S3 method for class 'mfrm_random_rater'
print(x, ...)
```

Arguments

`data`	Long-format data, one row per observed or assigned rating.
`person`, `rater`, `score`	Names of the person, rater and numeric score columns.
`facets`	Character vector of fixed-facet columns; default none. Include task or criterion columns explicitly when their difficulties are part of the intended model; their presence in `data` alone does not include them.
`score_levels`	Required consecutive integer score categories, in order.
`rater_sd`	`NULL` estimates the population SD of rater severity. A nonnegative number fixes it as known; zero removes rater heterogeneity.
`quad_points`	Number of standard-normal Gauss-Hermite points for each person integral, from 7 to 241; default 31. The result is also evaluated at `2 * quad_points + 1` points to check this integral's numerical stability. This check does not assess the separate rater Laplace approximation. If `checks$PersonQuadratureStable` is false, refit with more points and recheck all of `checks`; a higher-order evaluation alone does not refit the calibration. No allowed order is universally sufficient.
`maxit`	Maximum optimizer iterations per start; default 300.
`missing`	`"fail"` (default) refuses missing assigned scores; `"omit"` explicitly analyzes observed scores and records omitted rows.
`person_sd`	`NULL` (default) estimates the normal ability SD. A finite positive number fixes it as known; `person_sd = 1` retains the standard normal population. This is a population restriction with Rasch slope one.
`person_variance_max`	Positive upper search bound for estimated ability variance; default 16. Reaching it fails numerical readiness; increase the bound and refit rather than interpreting a capped estimate as a solution.
`x`	A result from `fit_mfrm_random_rater()`. Also for `summary()`: A fitted `mfrm_random_rater` object.
`draw`	Logical; `FALSE` returns plot data without opening a device.
`style`	`"interval"` keeps every labeled row; `"precision"` plots estimates against interval width (upper minus lower bound); `"distribution"` shows the empirical cumulative distribution of finite point estimates, excluding prior-only and unavailable scores. No density is reconstructed.
`sort`	Row order for interval plots: input order, increasing estimate, or increasing interval width. Missing values come last; ties retain input order. Sorting is descriptive, not a test of differences.
`palette`	`"accessible"` uses blue and dark orange with filled/open symbols; `"mono"` uses dark grey with the same symbol distinctions.

`title`, `caption` `NULL` uses the default text; a string replaces it; `""` removes it. Interpretation notes remain in the saved plot data.

`show_title`, `show_notes` Show the title and caption, respectively. `FALSE` hides text without deleting its metadata.

`show_labels` Show IDs on interval and precision plots. Distribution plots summarize estimates and do not label individual IDs. For crowded precision plots use `FALSE` and consult the retained table.

`reference` Vertical reference line in logits; `NULL` omits it. This is an orientation aid, not a quality threshold.

`text_scale` Positive multiplier for text size.

`point_size` Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

`intervals` `"none"` (default) displays point estimates without automatic prediction intervals. `"normal"` explicitly requests the first-order approximation from `confint(x, parm = "raters")`; nominal coverage is not established. Required for the interval-width precision view and sorting by interval width (`sort = "uncertainty"`).

`...` For `summary.mfrm_random_rater()`: Unused by summary and print.

For `plot()`: Unused.

`object` A fitted `mfrm_random_rater` object.

`calibration_intervals` For summaries, `"none"` (default) omits calibration bounds; `"normal"` explicitly requests their pointwise normal approximation. This does not select individual-rater intervals.

`level` Nominal level for explicitly requested summary calibration intervals; default 0.95. Numerical and variance-boundary restrictions remain.

Details

The adjacent-category logit is

$$\log\{P(Y_{prj} = k)/P(Y_{prj} = k-1)\} = \theta_p - u_r - x'_{prj}\beta - \delta_k.$$

Persons are independent $N(0, \sigma_p^2)$ and raters independent $N(0, \sigma_r^2)$, independent of persons and assignment. Fixed-facet level effects sum to zero. Steps are unconstrained adjacent thresholds; they need not be ordered or sum to zero. Their mean sets the overall location. Rater effects are not centered within the observed sample.

This is a frequentist likelihood fit, with no priors on calibration or variance parameters. Person effects are integrated by quadrature conditional on all shared rater effects; the resulting joint rater integral uses the Laplace approximation through the optional RTMB package. Integrating a new rater separately for each person would fit a different model. After marginalization, Persons sharing a rater are dependent. Adding Persons rated by the same fixed panel does not add independent draws from the rater population. Even if those rater effects became known exactly, a fixed number of draws would not consistently estimate a positive population variance. Person count or total rating count alone therefore cannot justify ordinary large-sample population-SD inference. The

full calibration information includes all estimated population SDs, but this does not establish the required asymptotic regime or Laplace accuracy.

The initial scope is RSM, unit weights, a connected person-rater design, multiple persons per rater, and full-rank additive fixed-facet coding. Every declared category must occur. Nonassigned combinations are absent; omitted scores are not imputed. Informative assignment, nonnormal rater populations, PCM, anchors, interactions, testlets and latent regression are not implemented by this route. Connectedness alone does not guarantee adequate information about rater variance. A fixed-facet level with only minimum or only maximum scores has no finite calibration estimate. Such separation fails `checks$NumericalReady` and `checks$InformationPositive`, even if the optimizer reports convergence. `checks$FixedFacetSeparation` records this sufficient separation check; its absence does not exclude separation along combinations of facets.

Both estimated zero variances are considered explicitly. One-sided variance scores check these boundaries; the derivative with respect to SD alone cannot establish them. The ability-variance score uses checked one-sided differences of the complete approximate marginal likelihood, including the rater Laplace determinant. At either estimated boundary, regular calibration and rater intervals are withheld. A fixed zero SD is a known submodel, not a fitted model with arbitrary fixed rater severities.

Fixed-facet and step tables retain estimates and approximate SEs but omit bounds by default. `confint(fit, parm = "calibration", level = 0.95)` or `summary(fit, calibration_intervals = "normal", level = 0.95)` explicitly requests observed-information normal approximations at an interior fit. Their finite-sample coverage is not established. Rebuilding summaries or `mfrm_results()` from an older saved fit applies this policy without refitting or changing the saved fit. Population-SD intervals are requested separately with `confint(fit, parm = "rater_sd")`; no ability-SD interval is supplied. This avoids a Wald interval that excludes zero by construction. Rater estimates are conditional modes. Their `ConditionalSD` holds calibration fixed; `PredictionSE` adds first-order calibration uncertainty through RTMB's generalized delta method. Individual rater interval bounds are missing by default: nominal coverage is not established. `confint(fit, parm = "raters")` explicitly requests the normal approximation from saved estimates without refitting; `plot(fit, intervals = "normal")` displays it. These target realized random effects, not fixed-rater coefficients, and do not classify or exclude raters. This output restriction does not correct interval coverage.

With estimated ability SD, a four-condition study (200 datasets per condition) found conditional individual-rater coverage of 91.6–91.8% with six raters and 94.0–94.1% with 24 raters; finite-interval availability was 99% and 87–88%, respectively. None met the prespecified combined coverage and availability criterion. Six-rater Monte Carlo uncertainty also prevented a conclusive material-undercoverage decision in this study. An earlier known-ability-SD pilot found 84.2% six-rater coverage. See `vignette("mfrmr-random-raters")` for designs and Monte Carlo intervals. These studies do not establish a safe minimum rater count. Numerical checks do not certify statistical validity or shared-rater Laplace accuracy.

`plot()` method: Plot observed-rater severity under a shared-rater RSM

For `plot()`: Points are conditional modes of severity relative to the rater population mean zero; positive values mean stricter ratings. No individual rater intervals are supplied automatically, including for earlier saved fits. Explicit normal whiskers are approximate 95% prediction intervals, not qualified rater classifications. At an estimated zero variance or with unresolved numerical checks, even requested whiskers are unavailable.

Value

An `mfrm_random_rater` object with `calibration_table`, `raters`, full covariance matrices, `checks`, analysis data/omitted-row accounting, model settings and optimizer records. It contains no native pointers; save it with `saveRDS()`. Its summary, plot and prediction methods are specific to this model; it is not an ordinary `mfrm_fit` or portable fixed-facet calibration object. Calibration and summary rater tables retain `IntervalStatus`, `CIEligible` and `InferenceReview` for normal-interval availability. These distinguish unrequested bounds, unsupported targets and unresolved checks or variance boundaries; see `confint.mfrm_random_rater()`. Retained points require the separate fit checks and are not certified by these fields.

For `plot()`: Invisibly, an `mfrm_plot_data` object with the rater table and settings. Use `plot_data()` for custom graphics.

Numerical evidence

An independent comparison evaluated local marginal-likelihood changes at eight saved calibrations for three-category RSMs with 240 Persons, six or 24 raters, and rotating or weakly linked assignments. Displacements spanned all six calibration coordinates and named contrast/SD directions, scaled to one local information unit. All 144 planned comparisons (128 distinct parameter points) met a 0.05-log-likelihood-unit tolerance including Monte Carlo uncertainty. The largest difference was 0.005 log-likelihood units. The reference reused joint posterior samples with an exact change-of-variables identity; the package likelihood and estimator were unchanged.

This supports the tested local likelihood changes, not an exact maximum- likelihood solution, an absolute likelihood normalization, the entire SD profile, variance boundaries or repeated-sampling interval coverage. Person quadrature checks remain distinct from this comparison. See `vignette("mfrmr-random-raters")` for the reference-precision limits and scope.

Model assumptions and related research

A shared rater effect represents differences in usual severity across raters, not inconsistency within a rater or differential severity toward a subgroup. Van den Noortgate et al. (2003, equation 7) describe crossed normal effects for binary responses. The binary case here has that sharing structure, with the rater sign reversed and unit Rasch slope. Huang and Cai (2024) extend crossed effects to item-level ordinal responses, using cumulative graded-response probabilities and estimated slopes. Their model and estimator are not this adjacent-category RSM and Laplace fit.

Fixing ability variance at one while keeping the Rasch slope at one is a population restriction, not merely a change of units. The default estimates this variance, keeping the ability mean zero and step location free. It does not fit group-specific distributions or latent regression. Unequal ability populations across rater panels can violate its assumptions even when the assignment is connected. A successful numerical check cannot assess those population assumptions. Earlier saved fits without ability-population parameters retain their original known N(0,1) meaning for predictions, profiles and bootstrap refits.

Relation to testlets and discrimination

A shared severity effect is not a random discrimination parameter. This RSM does not estimate GPCM slopes. Its rater effect spans persons, whereas `fit_mfrm_testlet()` uses a distinct effect

for each Person/block pair, even when the block is named Rater. These sharing structures lead to different marginal likelihoods and prediction targets; the two routes cannot currently be combined. See `gpcm_capability_matrix()` for the relationship to fixed slope families and local independence.

Comparison with fixed-rater MFRM

Compare the same observed rating events, categories, fixed facets and ability population. Fixed rater effects describe the observed panel; random effects refer to a population and are shrunk toward its mean. Their raw locations need not share an origin. Compare aligned rater contrasts or predictions at the same ability and facet settings. Setting `rater_sd = 0` removes all rater differences; it does not recover a model with freely estimated fixed rater effects. Shared raters induce dependence across Persons after marginalization, so a Person-count BIC or ordinary chi-squared likelihood-ratio test must not be copied from the fixed-facet model. `compare_mfrm()` checks matched events and compares centered facet summaries, retaining each model's assumptions and readiness. It does not rank models or calculate intervals for their differences.

References

Van den Noortgate, W., De Boeck, P. and Meulders, M. (2003). Cross-classification multilevel logistic models in psychometrics. *Journal of Educational and Behavioral Statistics*, 28, 369–386. doi:10.3102/10769986028004369.

Huang, S. and Cai, L. (2024). Cross-classified item response theory modeling with an application to student evaluation of teaching. *Journal of Educational and Behavioral Statistics*, 49, 311–341. doi:10.3102/10769986231193351.

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H. and Bell, B. M. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5), 1–21. doi:10.18637/jss.v070.i05.

See Also

`score_mfrm_random_rater()`, `predict.mfrm_random_rater()`, `plot.mfrm_random_rater()`, `mfrm_response_diagnostics()` for descriptive posterior predictive residuals, `confint.mfrm_random_rater()`, `mfrm_random_rater_intervals()`, `mfrm_results()`

`plot.mfrm_testlet_scores()` for view selection and accessible output. The distribution view summarizes the observed conditional modes, not the assumed rater population.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("fit_mfrm_random_rater", run.dontrun = TRUE)
## Not run:
# Inspecting this saved synthetic fit does not require RTMB.
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$random_rater$fit
# To refit instead, install optional RTMB >= 2.0 and run:
# ratings <- load_mfrmr_data("example_core")
# fit <- fit_mfrm_random_rater(ratings, "Person", "Rater", "Score",
#   facets = "Criterion", score_levels = 1:4, quad_points = 121)
summary(fit)
```

```
plot(fit)
system.file("examples", "extended-models.R", package = "mfrmr")

## End(Not run)
```

`fit_mfrm_testlet` *Fit a rating-scale model with dependence within Person-specific testlets*

Description

Account for extra dependence among ratings of the same person, for example several rubric criteria from one performance. A testlet identifies the ratings sharing this local effect within that person. The rating-scale model (RSM) estimates fixed facets, normal ability variance and one common local variance. For a rater effect shared across people instead, see `fit_mfrm_random_rater()`.

Usage

```
fit_mfrm_testlet(
  data,
  person,
  score,
  testlet,
  facets = character(),
  score_levels,
  testlet_variance = NULL,
  quad_points = 31L,
  maxit = 300L,
  variance_max = 16,
  missing = c("fail", "omit"),
  person_sd = NULL,
  person_variance_max = 16
)

## S3 method for class 'mfrm_testlet'
plot(
  x,
  facet = x$input$columns$facets[1],
  draw = TRUE,
  style = c("interval", "precision", "distribution"),
  sort = c("input", "estimate", "uncertainty"),
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
```

```
  reference = 0,
  text_scale = 1,
  point_size = 2.5,
  intervals = c("none", "normal"),
  level = 0.95,
  ...
)

## S3 method for class 'mfrm_testlet'
summary(object, ..., calibration_intervals = c("none", "normal"), level = 0.95)

## S3 method for class 'mfrm_testlet'
print(x, ...)
```

Arguments

`data`	Long-format ratings; each row is one assigned rating.
`person, score, testlet`	Column names. `testlet` identifies a group within a Person and may also occur in `facets`, for example a fixed Rater effect.
`facets`	Fixed-facet column names; default none. A column used as `testlet` is not automatically a fixed facet. For example, `testlet = "Task"` models local dependence; also specify `facets = "Task"` if the model should estimate fixed task difficulties.
`score_levels`	Consecutive integer categories in increasing order.
`testlet_variance`	`NULL` estimates the common local variance. A nonnegative number fixes it as known; zero removes local dependence.
`quad_points`	Standard-normal quadrature order for both integrals, from 7 to 121; default 31. Results are checked at `2 * quad_points + 1`.
`maxit`	Maximum iterations per optimization start; default 300.
`variance_max`	Upper search bound for an estimated variance; default 16. A fit reaching this bound is not numerically ready.
`missing`	`"fail"` (default) refuses missing assigned scores; `"omit"` analyzes observed rows and records the omissions.
`person_sd`	`NULL` (default) estimates the SD of a mean-zero normal ability population. A positive number fixes the SD as known. Use `1` to reproduce the earlier fixed N(0,1) model; this is a population assumption, not merely a choice of units.
`person_variance_max`	Upper search bound for estimated ability variance (not SD); default 16. Reaching it withholds numerical readiness.
`x`	An object from `fit_mfrm_testlet()`. Also for `fit_mfrm_testlet()`: An `mfrm_testlet` result.
`facet`	A fixed facet to display; defaults to the first fixed facet.
`draw`	`FALSE` returns plot data without opening a device.

`style` "interval" keeps every labeled row; "precision" plots estimates against interval width (upper minus lower bound); "distribution" shows the empirical cumulative distribution of finite point estimates, excluding prior-only and unavailable scores. No density is reconstructed.

`sort` Row order for interval plots: input order, increasing estimate, or increasing interval width. Missing values come last; ties retain input order. Sorting is descriptive, not a test of differences.

`palette` "accessible" uses blue and dark orange with filled/open symbols; "mono" uses dark grey with the same symbol distinctions.

`title, caption` NULL uses the default text; a string replaces it; "" removes it. Interpretation notes remain in the saved plot data.

`show_title, show_notes` Show the title and caption, respectively. FALSE hides text without deleting its metadata.

`show_labels` Show IDs on interval and precision plots. Distribution plots summarize estimates and do not label individual IDs. For crowded precision plots use FALSE and consult the retained table.

`reference` Vertical reference line in logits; NULL omits it. This is an orientation aid, not a quality threshold.

`text_scale` Positive multiplier for text size.

`point_size` Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

`intervals` "none" (default) shows points only; "normal" explicitly requests observed-information normal-approximation bounds. Interval-width views or sorting require this explicit selection.

`level` For `summary.mfrm_testlet()`: Nominal level for explicitly requested summary calibration intervals; default 0.95. Numerical and variance-boundary restrictions remain.

For `plot()`: Nominal level of explicitly requested calibration intervals; default 0.95. This does not establish finite-sample coverage.

`...` Unused.

`object` An `mfrm_testlet` result.

`calibration_intervals` For summaries, "none" (default) omits calibration bounds; "normal" explicitly requests their pointwise normal approximation. Person scoring remains a separate conditional output.

Details

The adjacent-category logit is

$$\log\{P(Y_{pbi} = k)/P(Y_{pbi} = k-1)\} = \theta_p + \gamma_{pb} - x'_{pbi}\beta - \delta_k.$$

Abilities are independent N(0,σ_p^2). Local effects are independent N(0,v), independent of abilities and assignment. A label reused by another Person represents a new local effect, not a shared random

rater. Every row has one non-overlapping membership. Fixed-facet effects sum to zero; free steps determine overall location. There are no calibration priors.

Frequentist MML uses nested one-dimensional Gaussian quadrature: integrate each local effect conditional on ability, multiply its block likelihoods, then integrate ability. Unequal block sizes and incomplete assignments are supported. At least two Persons must have multiple observed testlets and at least two Persons must have a testlet with repeated ratings. These initial design requirements do not prove estimability. Unit weights, RSM, additive fixed facets and one common variance are the supported scope; overlapping memberships, PCM, correlated effects, random rater effects shared across Persons, covariate-dependent ability means and nonnormal ability populations are not included. Estimating one ability variance does not establish homogeneous populations across assignment groups. After integrating both random effects, different Persons are independent under the model; ratings within a Person are not. Regular likelihood inference requires an identified interior model and growing information across independent Persons. The full calibration information includes every estimated variance; treating an estimated variance as known would omit nuisance-parameter uncertainty. A positive numerical Hessian alone does not prove identification, and fixed-order quadrature agreement does not establish an asymptotic approximation rate.

Zero is evaluated explicitly for both estimated variances using their one-sided variance scores, including the joint-zero submodel. All optimization starts are retained. Fixed coordinates are searched within `[-20,20]`; reaching a bound withholds readiness. Integration differences, the projected gradient and information are separate checks. A converged start is preferred when its objective differs from the best failed start only within floating-point rounding precision. Increasing quadrature order may be necessary; no order is universally sufficient. Calibration tables retain estimates and approximate SEs but omit bounds by default. `confint(fit, parm = "calibration", level = 0.95)` or `summary(fit, calibration_intervals = "normal", level = 0.95)` explicitly requests observed-information normal approximations. Their finite-sample coverage is not established. Numerical/information failures and estimated variance boundaries retain missing bounds. A regular variance interval is not supplied. Rebuilding summaries or `mfrm_results()` from older saved fits applies this policy without refitting or changing the source fit. If estimated ability variance is zero, Person scoring returns unavailable rows rather than degenerate zero-width intervals. Positive information for the remaining fixed coordinates does not resolve variance-boundary inference. `predict.mfrm_testlet()` supplies conditional Person scores and continuous equal-tail intervals; these exclude calibration-estimation uncertainty. Numerical agreement does not establish coverage or model fit.

`plot()` method: Plot testlet-model fixed-facet estimates

For `plot()`: Requested whiskers are observed-information normal approximations for calibration parameters; finite-sample coverage is not established. Missing intervals remain missing; points do not classify rater quality. `plot_data()` and `plot_data_components()` provide the same table without drawing. `as_ggplot()` preserves the interval meaning and missing whiskers.

Value

An `mfrm_testlet` object with calibration, covariance, calibration table, checks, all optimization runs, input/omission accounting and settings. It contains no native pointers; use `saveRDS()` and `readRDS()`. Calibration tables retain `IntervalStatus`, `CIEligible` and `InferenceReview` for missing or explicitly requested normal bounds; see `confint.mfrm_testlet()` for the status meanings. A retained point is not certified by interval availability. Inspect `checks` before interpreting it.

For `plot()`: Invisibly, an `mfrm_plot_data` object; use `plot_data()`.

Model assumptions and related research

Wang and Wilson's Rasch testlet model (2005) motivates dependence within a Person-specific block. With `testlet = "Rater"` and a fixed Rater facet, the more direct reference is their random-effects facet model (2005, equations 14–15): usual rater severity is distinct from an effect shared only within a Person/rater pair. Here the local variance is common to all testlets; it cannot rank raters by individual inconsistency.

Both papers allow ability variance to be estimated. This function estimates one normal ability variance by default while fixing its mean at zero and keeping free step location. This avoids adding a second location parameter. Unlike models with separate local variances, it retains a common testlet variance. The cited results do not guarantee coverage for a new assessment design or for this implementation.

Relation to discrimination and shared rater effects

Discrimination changes response sensitivity; it is not a substitute for dependence within a performance. This function fits an RSM, not a GPCM with additional slopes. Its local effect is shared only within a Person/block pair. A Person/task block shared across assessors differs from a Person/task/assessor rating-occasion block. Select the identifier according to the assessment process; only one nonoverlapping membership is supported. A nonzero local variance does not by itself establish halo. For one rater effect shared across persons, see `fit_mfrm_random_rater()`. The two models cannot currently be combined. See `gpcm_capability_matrix()` for the relationship to fixed discrimination and local independence.

Comparison with ordinary MFRM

Hold the observed events, categories, fixed facets, omissions and ability population constant. A known `testlet_variance = 0` removes local dependence while retaining the fixed Rater facet, if supplied. The corresponding ordinary model is an RSM MML with a matching normal population and location convention. For `person_sd = 1` this is the fixed N(0,1) model; estimated-population comparisons must also align location constraints rather than compare raw coefficients. An estimated zero variance lies on a boundary; ordinary chi-squared likelihood-ratio calibration cannot be assumed. `compare_mfrm()` checks matched events and compares centered facet effects with an ordinary RSM MML fit; it does not provide automatic ranking. Descriptive score or interval changes do not establish which model is better.

Assessment applications

With `testlet = "Task"`, the local effect is shared by a Person's ratings on that task. `vignette("mfrmr-testlet-applicati` provides runnable examples of allocating a fixed rating budget across tasks and comparing score sensitivity when tasks have unequal numbers of criteria. Its small enumeration reports model-conditional posterior precision and marginal EAP reliability while holding fitted calibration fixed; it is not a general design optimizer, a G/D study or a coverage guarantee. Ordinary MFRM does not generally assign equal information to arbitrary added criteria or tasks.

A common local variance cannot identify halo, establish that criteria are indistinguishable, or rank tasks by dependence. Task-specific variances are not estimated by this API. Fitting tasks separately does not remedy this: with one local block per Person, ability and local variances are confounded.

Unequal block sizes are handled through the likelihood, not by imposing equal task weights or removing every source of bias. Assignment, content and the origin of dependence require substantive review.

Earlier saved results

Saved fits without an ability-variance parameter retain their original N(0,1) interpretation when scored or reported. Refitting now estimates ability variance unless `person_sd = 1` is supplied. Reprinting or rescoring does not refit a population. Save a new fit and regenerate its scores together when changing the population assumption.

References

Wang, W.-C. and Wilson, M. (2005). The Rasch testlet model. *Applied Psychological Measurement*, 29, 126–149. doi:10.1177/0146621604271053.

Wang, W.-C. and Wilson, M. (2005). Exploring local item dependence using a random-effects facet model. *Applied Psychological Measurement*, 29, 296–318. doi:10.1177/0146621605276281.

See Also

`predict.mfrm_testlet()`, `plot.mfrm_testlet()`, `mfrm_results()`, `mfrm_response_diagnostics()` for descriptive posterior predictive residuals.

mfrmr_workflow_methods, `mfrmr_output_guide()`, `mfrmr_interval_guide()`, `plot.mfrm_testlet_scores()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("fit_mfrm_testlet", run.dontrun = TRUE)
## Not run:
# Saved fit for the synthetic example_core ratings.
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$testlet$fit
# To refit instead (this takes longer than inspecting the saved result):
# ratings <- load_mfrmr_data("example_core")
# fit <- fit_mfrm_testlet(ratings, "Person", "Score", "Rater",
#   facets = c("Rater", "Criterion"), score_levels = 1:4, quad_points = 121)
fit$checks
plot(fit, facet = "Rater")
# The complete regeneration recipe is included with the package:
system.file("examples", "extended-models.R", package = "mfrmr")

## End(Not run)
```

`fit_rater_accuracy` *Estimate criterion-referenced rater accuracy from exact matches*

Description

Fit a binary many-facet Rasch model to agreement with supplied reference ratings. Raters are the objects of measurement; higher accuracy logits indicate a greater probability of matching the reference, after adjusting for the difficulty facets.

Usage

```
fit_rater_accuracy(
  data,
  rater,
  facets,
  score,
  reference,
  method = c("JML", "MML"),
  maxit = 400,
  reltol = 1e-09,
  quad_points = 31
)
```

Arguments

`data`	A data frame with one row per rater and benchmark combination.
`rater`	Name of the rater identifier column.
`facets`	Names of one or more difficulty-facet columns. Their joint combination identifies a benchmark, for example `c("Essay", "Domain")`. For a separate difficulty for every performance-domain combination, supply a single preconstructed benchmark identifier instead. These are different models; choose the structure substantively.
`score`, `reference`	Names of aligned observed and reference rating columns. Both must be numeric, or both character/factor category labels. Numeric equality is exact; factors are compared by labels. Scores must already be categorized. No rounding, averaging or partial credit occurs.
`method`	Ordinary `"JML"` (default) or `"MML"`; see Details.
`maxit`, `reltol`, `quad_points`	Optimization and quadrature controls passed to `fit_mfrm()`. Quadrature applies only to MML.

Details

Following the dichotomous example in Engelhard, Wang and Wind (2018, pp. 42-43), each complete pair becomes `Match = 1` for equality and `0` otherwise. The additive model is `logit Pr(Match = 1) = accuracy[rater]`

Each difficulty facet is sum-to-zero centered; accuracy locations are free. There is no free binary threshold. This function reuses the native RSM estimator on the declared 0/1 scale, with unit weights, no interactions, anchors, shrinkage or JML correction. It does not reproduce the paper's interaction tests or its FACETS results.

References are treated as fixed supplied benchmarks. Their quality, selection and adjudication require substantive justification; their uncertainty is not estimated. A benchmark must have at most one distinct nonmissing reference across raters, even on rows with missing observed scores. Repeated rater-by-benchmark rows are rejected, including repetitions with a missing score. Choose independent observations and an appropriate facet structure before fitting. Identifiers must be present and nonblank. Distinct numeric identifiers that become the same text label are rejected; provide unambiguous character identifiers instead. Missing observed/reference ratings are excluded with separate indicators in `cases`; missing references are never filled from other rows. Both matches and mismatches must remain. Raters with no retained pairs still appear in `table` with unavailable estimates.

JML estimates a fixed accuracy location per rater and has the usual incidental-parameter and sparse-design limitations. All-match/all-mismatch raters retain primary `Inf`/`-Inf` estimates, not the finite optimizer proxies. MML instead estimates a normal population of rater accuracies and returns rater EAPs with `PosteriorSD`. Its effective population sample size is the number of raters, not the number of benchmark ratings. Posterior SD conditions on fitted calibration/population parameters; it is not a frequentist SE or a confidence interval. Small rater pools and population misspecification need particular review. No JML SE, interval, significance test or automatic accuracy cutoff is computed here. MML explicitly uses `population_formula = ~ 1` to estimate the rater population mean and variance; it does not use the RSM default fixed N(0,1).

Exact agreement proportions describe the retained benchmark mix. Accuracy logits adjust for the specified difficulty structure under the response model, local independence and a connected identifiable design. They are neither ordinary rater severity nor examinee proficiency. Binary coding discards the direction and magnitude of disagreements. The native fit's readiness and estimate-use labels remain visible; finite numbers do not establish reportability, model fit or transport to other benchmarks. High separation reliability for these accuracy locations would describe differences in accuracy, not agreement or differences in severity.

Value

A plain list with `table` (rater counts, exact agreement, accuracy, posterior SD and source status), `difficulty` (facet locations), `cases` (all input rows, binary matches and missingness), `fit` (native `mfrm_fit`), `settings` and `notes`. Internal `Person` identifiers in `fit` refer to raters; `Facet1`, `Facet2`, etc. refer to the requested difficulty facets. `settings$column_map` records this mapping, also used by `cases`. Interpret native fit reports by these roles, not their default examinee/rater labels. `difficulty$Facet` restores the input facet names. Save the complete result with `base::saveRDS()`; the data frames can be exported with `utils::write.csv()`. Reading saved output does not refit.

References

Engelhard, G., Jr., Wang, J., & Wind, S. A. (2018). A tale of two models: Psychometric and cognitive perspectives on rater-mediated assessments using accuracy ratings. *Psychological Test and Assessment Modeling, 60*(1), 33-52. Equations and binary coding: pp. 42-43.

See Also

fit_mfrm(), score_comparison_table(), rater_centrality_table()

Examples

```
# A small illustrative benchmark exercise, not a validation sample.
d <- expand.grid(Judge = paste0("J", 1:4), Benchmark = paste0("B", 1:6))
d$Reference <- rep(c(1, 2, 3, 1, 2, 3), each = 4)
matched <- c(1,1,0,0, 1,0,1,0, 0,1,0,1,
             1,1,0,0, 0,1,1,0, 0,0,1,1)
d$Rating <- ifelse(matched == 1, d$Reference, (d$Reference %% 3) + 1)
accuracy <- fit_rater_accuracy(d, "Judge", "Benchmark", "Rating", "Reference")
accuracy$table
accuracy$difficulty
# Review SourceFitReadiness and EstimateUse, especially for small samples.
```

---

fit_rater_network *Fit an ERGM to a declared binary rater network*

Description

Model network configurations and rater covariates after examining shared rating coverage with rater_network_analysis() or rater_halo_network_analysis(). Estimation uses the optional ergm package.

Usage

```
fit_rater_network(
  x,
  terms,
  threshold,
  relation = c("greater", "less"),
  node_data = NULL,
  missing = c("error", "model"),
  control = NULL,
  gof = NULL,
  gof_control = NULL,
  seed = NULL
)

## S3 method for class 'mfrm_network_ergm'
summary(object, ...)

## S3 method for class 'mfrm_network_ergm'
coef(object, ...)

## S3 method for class 'mfrm_network_ergm'
```

```
vcov(object, ...)

## S3 method for class 'mfrm_network_ergm'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'mfrm_network_ergm'
plot(x, type = c("coefficients", "gof"), draw = TRUE, ...)
```

Arguments

x	For `fit_rater_network()`: A rater or halo network result. All eligible pair metrics are used, before graph/plot filtering; missing comparisons never become zero ties. For `plot()`: Output from `fit_rater_network()`.
terms	One-sided ERGM formula, for example `~ edges + mutual` for a directed network, or `~ edges + nodecov("Experience")`. See `ergm::ergmTerm` for terms appropriate to the declared graph.
threshold	Required finite cutoff for the chosen pair metric.
relation	`"greater"` makes a tie when the metric is strictly greater than the threshold; `"less"` makes one when strictly smaller. Equal values are observed non-ties. Negative alpha is not automatically a positive tie.
node_data	Optional data frame with `Node` IDs and complete node covariates. IDs must match all nodes exactly; order is matched by ID.
missing	`"error"` requires all dyads to be observed. `"model"` explicitly permits unobserved dyads, represented by `NA`, using ergm's observed-data likelihood under an ignorable missingness assumption.
control	Optional `ergm::control.ergm()` object.
gof	Optional one-sided formula for simulation-based model checks, e.g. `~ degree + distance` for an undirected graph. See `ergm::gof.ergm`.
gof_control	Optional `ergm::control.gof.ergm()` object.
seed	Optional integer seed; the caller's RNG state is restored.
object	An `mfrm_network_ergm` result.
...	For `summary.mfrm_network_ergm()`: Unused; unexpected arguments are rejected by extractors. For `plot()`: For GOF, arguments passed to ergm's `plot.gof`; for coefficients, arguments passed to base `graphics::plot()`.
parm	Coefficient names or positive integer positions; NULL selects all.
level	Confidence level strictly between zero and one.
type	`"coefficients"` or `"gof"` (requires saved GOF).
draw	Whether to draw; `FALSE` returns plot data only.

Details

This is a binary ERGM, not a valued/signed ERGM or a rating-response model. A coefficient is a conditional network log-odds coefficient for its change statistic; it is not an MFRM severity, a causal rater effect or an independent rating-level regression. Thresholds, sampling of raters, overlap and node covariates define the question. Prespecify them and assess sensitivity by refitting at substantively plausible thresholds; do not select a threshold for significance. Different thresholds define different response networks: their AIC values do not select a best threshold.

A missing dyad includes insufficient overlap and an undefined association. Such missingness need not be ignorable (e.g. constant scores or assignment related to rater characteristics). `missing = "model"` asserts an assumption, not a correction for informative missingness. Self-ties are excluded. Unobserved dyads are not structural zeroes. For structural constraints or valued ERGMs, use the returned `network` with `ergm::ergm()` explicitly.

Estimation requests MLE. For dyad-independent terms MPLE is also the MLE; dependent terms require MCMC. A returned fit is not proof of convergence or model adequacy. Inspect `ergm::mcmc.diagnostics(result$fit` and the saved `gof` object; review warnings, degeneracy, effective sample size and mixing. GOF tail areas are simulation diagnostics, not tests of halo. Intervals in the coefficient table are approximate pointwise normal intervals, conditional on the chosen network model and adequate estimation. Small rater networks may not support complex models. Neither mfrmr nor this wrapper validates arbitrary ERGM specifications or corrects estimated-edge uncertainty.

`confint()` preserves unavailable coefficients as missing bounds. Its matrix attributes retain `level`, `method`, `diagnostics` and `notes`. These intervals remain conditional on the selected binary network and model; they do not propagate estimated-edge or model-selection uncertainty. Changing `level` does not change the fitted object or its stored plot.

`plot()` method: Plot ERGM model checks or coefficient intervals

For `plot()`: Coefficient intervals are approximate pointwise normal intervals. They are not evidence of MCMC convergence. The GOF plot delegates to ergm; its graphical options remain available through ....

Value

An `mfrm_network_ergm` object with the backend `fit`, binary `network`, complete `dyads`, `nodes`, coefficient/diagnostic/settings tables, backend warnings and optional `gof` with long-form simulation tables. `summary()` retains model settings and diagnostic cautions. `coef()` and `vcov()` extract backend coefficient coordinates; `confint()` returns approximate pointwise normal intervals without refitting. `plot()` uses the originally stored 95% intervals or saved simulation GOF. Use `mfrm_report()` and `export_summary_appendix()` for stored reporting.

For `plot()`: Invisibly, an `mfrm_plot_data` object containing coefficient rows or GOF summaries and long-form simulation draws. Use `plot_data()` to export.

References

Lamprianou, I. (2018). Investigation of rater effects using social network analysis and exponential random graph models. Educational and Psychological Measurement, 78, 430–459. doi:10.1177/0013164416689696.

See Also

`rater_network_analysis()`, `rater_halo_network_analysis()`, `plot.mfrm_network_ergm()`

Examples

```
# Optional integration example (loads additional modelling or rendering packages).
# example("fit_rater_network", run.dontrun = TRUE)
## Not run:
if (requireNamespace("ergm", quietly = TRUE) &&
    requireNamespace("igraph", quietly = TRUE)) {
  ratings <- load_mfrmr_data("example_core")
  fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
    method = "JML", maxit = 300)
  rn <- rater_network_analysis(fit, weight_metric = "KrippendorffAlpha")
  # Four synthetic raters: illustrate the handoff, not a complex ERGM.
  # Tie = pairwise interval alpha strictly above the declared 0.35 cutoff.
  model <- fit_rater_network(rn, ~ edges, threshold = 0.35, seed = 41)
  model$dyads
  summary(model)$diagnostics
  confint(model, level = 0.90) # Approximate, conditional on this binary network
  mfrm_report(model)$tables$coefficients
  plot(model, draw = FALSE)
}

## End(Not run)
```

---

gpcm_capability_matrix

GPCM workflow and score availability

Description

Check which GPCM workflows can be used in mfrmr, the limits that apply to each workflow, and the recommended alternative when a route is not available.

The table is intended for route selection before or after fitting and is limited to workflow availability, interpretive constraints, and the route to use next. Except for the multiple-slope and EM rows, the table describes one facet supplying relative discriminations. The provisional two-family route has a separate scope below. The one-family model uses one slope facet. MML permits a different facet for category steps; JML requires `slope_facet == step_facet`. It has one substantive ability dimension. Separate-owner MML supports fitted-object scoring, information, slope/curve uncertainty, matched PCM comparison and saved inference. Weighting reviews and simulation/design workflows still require the same owner; same-design `bootstrap_mfrm_gpcm()` supports separate owners. Numerical eligibility is not a coverage guarantee. These structural choices are stated separately from the availability of each output. An available probability or descriptive comparison does not establish eligibility for a confidence interval or model-selection rule.

JML fits estimate relative slopes but do not support the MML slope interval, bootstrap, curve-uncertainty or inferential comparison routes described here. Their available facet/location SEs are exploratory observation-table approximations, not slope SEs or nuisance-adjusted joint-information SEs. Local curvature checks do not establish bias control or interval coverage. Fitted-object JML scoring uses post-hoc EAP with a default standard-normal reference prior or an explicit

scoring_prior; it is not ML/WLE scoring. Portable GPCM extraction supports MML and shared-owner JML within their distinct source-check scopes. JML local checks leave incomplete global audits unchanged. See fit_mfrm() for JML estimation and boundary conventions and predict_mfrm_units() for conditional scoring.

Usage

```
gpcm_capability_matrix(
  status = "all",
  view = c("workflows", "scores", "unavailable")
)
```

Arguments

status	Which rows to return. For view = "workflows" or "unavailable", use "all" (default), "supported", "supported_with_caveat", "blocked", or "deferred". For view = "scores", use "all", "available_with_caveat", "supporting_route", or "unavailable".
view	The table to read: "workflows" (default) gives the general map; "scores" describes package-native versus FACETS-style score quantities; "unavailable" lists blocked/deferred routes and alternatives.

Details

Status has the following user-facing meanings:

- supported: the helper is available within the stated boundary;
- supported_with_caveat: the helper runs, but its interpretation is restricted as described in Boundary;
- blocked: the helper intentionally stops for a GPCM fit;
- deferred: no public mfrmr route is currently available.

Read Boundary before interpreting a caveated result. For a blocked or deferred row, use RecommendedRoute to choose a supported analysis or a Rasch-family alternative.

In the score view, available_with_caveat denotes an available quantity within its Limitation, supporting_route denotes related evidence or a reference model rather than a replacement for that quantity, and unavailable denotes a quantity not supplied. Package-native GPCM scores and uncertainty are not FACETS score-side equivalents. These statuses are kept separate from the general workflow statuses to preserve their meaning.

In 0.2.4, this is the single public entry for all three tables. Replace gpcm_score_side_contract(status) with gpcm_capability_matrix(status, view = "scores"), and gpcm_runtime_guard_coverage() with gpcm_capability_matrix(view = "unavailable"). Each view retains its former column schema and status meanings. These tables describe supported routes; they do not inspect a particular fit or certify its inferential eligibility.

Value

For `view = "workflows"`, a data.frame of class `mfrmr_gpcm_capabilities` with one row per workflow family and columns:

- `Area`
- `Helpers`
- `Status`
- `Boundary`
- `RecommendedRoute`

For `view = "scores"`, an ordinary data.frame with `Capability`, `Status`, `Limitation` and `Alternative`. For `view = "unavailable"`, an ordinary data.frame with `Area`, `Helper`, `Status`, `Boundary` and `RecommendedRoute`.

Connected outputs and unavailable extensions

For the current one-slope-family model, use `summary(fit)` to review the model and numerical status, `plot(fit, type = ...)` for available location, fit and category views, and `stats::confint()` / `mfrm_curve_intervals()` for separately checked MML uncertainty. An interval result has its own `print()` and `plot()` methods; attach selected results to `mfrm_results()` for `mfrm_report()` and `export_mfrm_results()`. A plot of locations or fit statistics is not a plot of slope uncertainty. `mfrm_facet_intervals()` separately supplies native-scale model-information Wald intervals for one-family MML locations and within-facet contrasts. Direct fixed/adaptive MML, centered additive facets/steps and unit weights are required, with an estimated intercept-only normal population or the explicit fixed-N(0,1) restriction. Shared/separate slope and step owners are supported. Full-information and finer-grid checks preserve refusal reasons; regular asymptotic validity assumes a correctly specified, identified interior marginal model with nonsingular limiting information. Finite-sample coverage is not guaranteed. Explicit zero-sum contrasts can be standardized by the same fit's population SD using the full delta method; individual standardized-location intervals are not supplied.

Portable calibration has its own row and `mfrm_calibration_capabilities()` gives estimator-specific source restrictions. Two slope families are available provisionally through `fit_mfrm()` with fixed-standard-normal MML, exactly two facets, no anchors and unit weights. Choose fixed-grid EM or adaptive direct MML explicitly; see `fit_mfrm()`. Use `summary(fit)`, then inspect adaptive initialization in `fit$opt$mml_initialization`: the default compares neutral and EM-derived starts with the same adaptive likelihood, retaining failed starts and the selected candidate's convergence status. The saved initialization policy also follows quadrature-order refits. Use `curves <- mfrm_curve_intervals(fit, newdata)` and `plot(curves)`. Despite the function name, this route returns fitted values with unavailable intervals. Attach them with `mfrm_results(fit, intervals = curves)` for reports and saved exports. Without attachments, `mfrm_results(fit)` collects the saved fit and its numerical status; neither call calculates new diagnostics. Separately use `ci <- confint(fit, parm = "slopes")` for experimental component log-Wald intervals and attach `intervals = list(slopes = ci, curves = curves)`. Local numerical checks do not establish global identification or sampling coverage; failed checks retain missing bounds. An explicit `confint(fit, parm = "slopes", method = "profile", slope = c(Task = "t1"))` profiles one two-family component, using the actual owner/level names. It preserves the fitted fixed or adaptive integration method; adaptive nodes, scales and Jacobians are recomputed throughout the constrained searches.

It retains nuisance reoptimization, numerical checks, unavailable endpoints and a same-target Wald comparison. Neither method has qualified coverage. Use `mml_quadrature_sensitivity()` to compare refits with the same engine and integration method. Both routes retain each order's experimental log-Wald checks. Optional `adaptive_quad_points` also adds fixed-calibration integration checks for two families; the posterior moments in that review are numerical diagnostics, not a Person-scoring workflow. Person-score comparisons remain unavailable for two families. For both engines, `mfrm_response_diagnostics()` supplies same-data posterior predictive residuals with fixed calibration, including descriptive Infit/Outfit without reference cutoffs. It retains the fitted integration method and complete Person conditioning record, including when selecting output rows. Attach the saved object through `response_diagnostics` to the results call above for plots, reports and exports. `predict_mfrm_units()` separately supplies experimental conditional new-Person EAP, posterior SD and continuous posterior intervals under the retained N(0,1) prior, with source/batch numerical checks and no calibration uncertainty. `extract_mfrm_calibration()` provides portable two-family format 6 with the same fixed prior and separate source/batch checks. `mfrm_facet_intervals()` supplies separately checked experimental normal intervals for either owner's locations or prespecified within-facet contrasts. It uses the constrained location block of the inverse full marginal information; failed numerical checks retain point estimates with missing intervals. Attach the result as `intervals = list(locations = ci)` for saved reports and plots. Coverage is unqualified; location differences are not uniform rating differences when slopes or steps vary. Ordinary fit diagnostics, step/curve intervals and model ranking remain unavailable. Shared-owner corrected JML has an explicit experimental point-estimation and reporting route through `jml_correction_order`; formal structural intervals remain unavailable. Its descriptive response diagnostics and conditional EAP scoring use their own saved identities and source checks. Portable corrected calibration uses file format 5; no ordinary JML likelihood checks or structural intervals are inherited. Separate slope and step owners alone still define one slope family.

Model, estimation and algorithm

RSM, PCM and GPCM describe response probabilities, not a particular fitting algorithm. MML integrates over an ability distribution; JML estimates the training persons' abilities jointly with the other parameters. EM and direct optimization are numerical ways to fit an MML model. RSM/PCM can therefore also use MML–EM. In this package, MML defaults to direct optimization; RSM/PCM EM and hybrid support additive, fixed-population models with fixed integration (`population = NULL`). One-family GPCM EM/hybrid requests fall back to direct and record the engine in `fit$summary`.

The provisional two-slope-family GMFRM route uses fixed-standard-normal MML with fixed-grid generalized EM (`mml_engine = "em"`) or adaptive direct optimization (`mml_engine = "direct", mml_integration = "ada`
`mml_integration = "auto"` preserves the selected engine and increases integration points up to `mml_accuracy_control = list(max_points = ...)`, including reference rules, while retaining the numerical and information checks. Fixed integration remains the package default; two-family direct fitting requires explicit adaptive or automatic integration. Adaptive EM and two-family hybrid fitting are unavailable. EM is not a defining property of GMFRM. See the GPCM scope vignette for the distinction between the model, estimation method and algorithm, with links to Muraki, TAM and sirt.

Applications of two slope families

In a speaking assessment, task difficulty and assessor severity describe shifts in ratings, whereas task and assessor slopes describe how responses change with ability. Their product is the effective

slope for a rating. For example, 0.8 times 1.2 gives 0.96; it is a log-odds multiplier, not an accuracy percentage or a score weight. The same roles can use piece/judge or station/examiner labels, provided the single-ability assumption is appropriate. Two slope families do not create two latent abilities or a separate free slope for every task-by-assessor pair.

Potential uses include task review and assessor feedback. A high assessor slope is not a competence threshold, and a low task slope is not a rule for deleting content. Read category and information curves over the relevant ability range, inspect uncertainty and assignment overlap, and bring in reference ratings or substantive evidence for accuracy claims. The vignette explains these uses and a provisional two-family fitting example. It does not yet provide a two-family model comparison or qualified feedback decision. It also shows how to import the empirical writing table `sirt::data.ratings1` from that separately installed package, preserve its categories, and review unequal assignment. The example distinguishes new fitted response curves from evidence of predictive accuracy. The original fixed-grid fits had integration-sensitive curves and incomplete residual summaries. A subsequent adaptive fit resolved the retained numerical example; that agreement does not validate model fit or feedback decisions. All score datasets bundled with mfrmr are synthetic.

Rankings and consequential decisions

An official competition result, highest latent ability and a future winner are different targets. The scope vignette reviews actual figure-skating score protocols, changing judge panels and advancement to a final. Individual score intervals, high rank correlation or a high G coefficient do not establish the probability of selecting the correct champion. Differences need covariance and selection-aware uncertainty; independently drawing from printed SEs omits shared calibration uncertainty. There is no winner-probability or simultaneous Person-rank confidence-set API. Available one-family and experimental two-family new-Person EAP intervals condition on the calibration and prior; they do not provide a validated winner-selection procedure. Preserve competition-specific aggregation, rounding, tie rules and advancement separately from modelled ability. Do not interpret sensitivity to a judge's marks as proof of bias.

Local independence, testlets and random effects

Two fixed slope families change response sensitivity; they do not by themselves remove dependence between ratings from the same performance. A testlet effect is itself a random effect, defined by its sharing unit. A testlet model assumes independence conditional on ability and its local effect, with dependence remaining after the effect is integrated out. One substantive ability can therefore coexist with additional latent dependence variables.

`fit_mfrm_testlet()` supplies a Person-local RSM block effect; reusing a block label for another Person creates a different effect. `fit_mfrm_random_rater()` instead supplies a rater severity effect shared across persons. These separate RSM routes neither add effects to GPCM nor jointly estimate shared-rater and testlet effects. Random discrimination is also different from estimating two fixed slope families. The GPCM scope vignette explains effect-sharing units, observed versus new block/rater prediction and the distinction from multivariate G-theory. A testlet variance does not diagnose halo or enforce equal task weights.

Interpreting the former bounded GPCM label

Use GPCM as the model name and state the structure and output restrictions explicitly. The older label described limited implementation scope, not a distinct unidimensional response model. Positive log-parameterized relative slopes retain their geometric-mean-one identification; there is no

extra user-facing finite slope box. Rejection of numerical overflow/underflow is not clipping to a valid estimate. Separate owners, MML inference/comparison and portable calibration are available within their respective scopes; two-family fitting provides provisional estimates and conditional curves, with separately checked experimental component-slope intervals. JML slope intervals remain unavailable. A data-specific boundary or unbounded interval is a separate statistical issue, and retirement of the label does not certify interval coverage.

Estimates, intervals and comparisons

For free slopes, these are different questions:

- **What did the numerical fit return?** `fit$slopes$OptimizerEstimate` retains the fitted relative slopes for descriptive sensitivity analysis. The compatibility column `Estimate` contains the same numerical values; it does not override `ParameterStatus` or `PrimaryEstimate`.
- **How uncertain is a slope?** `confint(fit, parm = "slopes")` returns approximate pointwise intervals for eligible GPCM MML fits. `diagnose_mfrm(fit)$parameter_uncertainty$slopes` supplies the same 95% calculation with `CIEligible` and `InferenceReview`. Ineligible solutions retain missing ordinary bounds and explicitly labelled `Optimizer*` diagnostic quantities. Old eligibility flags do not authorize an interval.
- **Which model should be selected?** MML information criteria may be retained numerically. `compare_mfrm()` ranks GPCM MML candidates when its separate solution and comparison checks pass. `ICSelectable` describes likelihood and integration requirements; `ICFitEligible` describes the fit's solution check; `ICComparable` is the final comparison decision. The weighting review preserves this decision without making an operational-scoring recommendation. With `nested = TRUE`, it can also request the separately checked PCM/GPCM equal-slope test.

The inference restrictions above concern the current package implementation; they are not a claim that GPCM inference is impossible in general. A local rank or curvature check does not by itself assess competing solutions, numerical integration error or the performance of an interval procedure. Relative-slope intervals have their own checks. A PCM/GPCM LRT requires the matched comparison described below; more iterations or quadrature points alone do not establish its structural assumptions.

Different requirements for intervals and model comparison

These decisions are separate, and do not follow from unidimensionality:

- **Slope intervals:** `confint.mfrm_fit()` with `parm = "slopes"` uses the inverse joint observed information, including estimated population parameters, and the sum-zero log-slope transformation. It checks likelihood consistency, convergence, positive unregularized information, unit weights and a grid of at least 31 points. The exponentiated log-Wald limits are pointwise model-based approximations for geometric-mean-one relative slopes by default. Explicit options add population-SD-standardized slopes, named ratios or differences, Bonferroni adjustment and independent-cluster sandwich covariance. An experimental `method = "profile"` with one named `slope` reoptimizes all nuisance parameters, including the normal population, and retains endpoint failures and a saved likelihood plot. Its initial scope excludes anchors, interactions, covariates and standardized targets. Profile coverage and superiority over Wald have not been established. Small samples and misspecification can still affect coverage. Failed checks retain missing limits and a reason. `bootstrap_mfrm_gpcm()` provides

fitted-model bootstrap intervals or a matched PCM/GPCM test; `mfrm_curve_intervals()` propagates calibration uncertainty to curves. Probability-curve intervals showed undercoverage in a saved-fit study of small incomplete designs, including finite-grid Bonferroni families. Numerical availability and multiplicity adjustment do not certify nominal coverage; bootstrap coverage requires separate evidence too. Saved results support `apa_table()`, `plot_data()` and `as_ggplot()`; attach selected intervals to `mfrm_results()` for reports and exports. Their targets remain separate from Wright/Pathway location and fit displays.

- **Information criteria:** AIC/BIC compare maximized likelihoods on the same response data, with appropriate free-parameter counts and numerical accuracy. They do not require a slope confidence interval or nested models. The GPCM MML solution check reevaluates the retained likelihood, terminal gradient and positive unregularized local information without refitting. It uses the existing numerical-gradient tolerance (at most 10^{-4}) and information-inversion eigenvalue tolerance. The existing joint information calculation is shared with slope intervals and governed by `options(mfrmr.max_information_bytes = 256 * 1024^2)` rather than an 80-coordinate cutoff. The budget estimates dense matrix workspace, not total process memory. An unavailable check gives a reason, not permission to rank. Local checks do not prove global optimality or integration accuracy; inspect different starts and `mml_quadrature_sensitivity()` when the decision is close.
- **PCM/GPCM likelihood-ratio test:** with the same population model, step structure and other constraints, setting all relative slopes to one gives PCM. One is an interior positive slope value, not a variance-zero boundary. With G slope levels and no other differing free parameters, the null imposes $G - 1$ independent log-slope restrictions. A chi-square reference additionally requires identified, regular solutions and adequate sample information. `compare_mfrm()` with `nested = TRUE` checks the matched model settings, G-1 free dimensions and regular local MML solutions before reporting an asymptotic chi-square p-value. Default PCM and GPCM calls can use different population models: supply `population_formula = ~1` and the same person data to both fits to compare estimated-normal models. Small or sparse samples can give inaccurate asymptotic p-values; examine starting-value and quadrature sensitivity and report the test's assumptions.

The official R AIC documentation describes likelihood comparability. The mirt model documentation documents GPCM and information-matrix SEs, and its model-comparison documentation describes likelihood-ratio and information-criterion comparisons. These are examples of supported statistical methods, not validation of mfrmr's many-facet implementation.

Comparing models with ConQuest and TAM

Match the response formula before comparing estimates. In mfrmr, the selected positive slope multiplies ability, facet locations and the category step together. A slope multiplying ability alone, with separately additive rater severity, generally specifies a different many-facet model.

TAM's `tam.mml.mfr()` does not estimate slopes itself, but its documented Example 14, Model 14c combines a facet intercept design with grouped slopes in `tam.mml.2pl(irtmodel = "GPCM.design")`. ConQuest estimates GPCM scores with `scoresfree`; its default slopes belong to combinations of facets (generalized items), with further grouping available through a scoring design. Neither construction automatically reproduces mfrmr's single slope family and complete-predictor multiplication. See the TAM fitting documentation and ConQuest Note 8.

A matched item-only, positive-slope GPCM with an estimated normal population can be expressed on either scale. mfrmr fixes the geometric mean of relative slopes α_i to one and estimates the

population SD σ conditional on any population covariates. On the unit-variance scale the slopes become $a_i = \sigma\alpha_i$. Locations, steps and any population regression also need transformation. This equivalence does not include imposing both unit variance and geometric-mean-one slopes: together they impose an additional restriction.

Transformed intervals require the joint parameter covariance. Multiplying relative-slope interval endpoints by an estimated population SD omits its uncertainty and covariance with the slopes. Use `confint.mfrm_fit()` with `parm = "slopes", scale = "standardized"` for the full transformation; with covariates the SD is the residual population SD. TAM's documented `tam.se()` omits parameter covariances; ConQuest distinguishes the covariance of parameter estimates from the latent-population covariance. Neither marginal SEs nor a latent-population covariance matrix replace the required joint matrix. Numerical replication requires matching data, model and identification. IC selection can compare different, nonnested models on compatible likelihoods; verify each model's free-parameter count and integration accuracy. A likelihood-ratio test additionally requires a nested null. Consult `vignette("mfrmr-gpcm-scope")` for examples, sources and the scope of existing numerical comparisons. These do not supply a general GPCM import or comparison API.

Conditional Person uncertainty

Person posterior SDs and intervals, where returned, condition on the fitted calibration; they are not slope intervals or calibration-aware confidence intervals. An unstable calibration also limits their interpretation.

Typical workflow

1. Call `gpcm_capability_matrix()` before using `GPCM` in a new workflow.
2. For `supported_with_caveat`, read `Boundary` before interpreting output.
3. For `blocked` or `deferred`, follow `RecommendedRoute` instead.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `compute_information()`, `predict_mfrm_units()`, `sample_mfrm_plausible_values()`, `reporting_checklist()`, mfrmr_workflow_methods, mfrmr-package

Examples

```
gpcm_capability_matrix()
gpcm_capability_matrix("supported")
gpcm_capability_matrix("blocked")
gpcm_capability_matrix(view = "scores")
gpcm_capability_matrix("unavailable", view = "scores")
gpcm_capability_matrix(view = "unavailable")
```

`gwet_ac2_table` *Compute Gwet's AC2 or unweighted AC1*

Description

Compute observed-score agreement across two or more raters, with a declared category universe and quadratic, linear or unweighted agreement.

Usage

```
gwet_ac2_table(
  data,
  rater,
  unit,
  score,
  categories,
  weights = c("quadratic", "linear", "unweighted")
)
```

Arguments

`data`	Long-format ratings with at most one row per rater and unit.
`rater`	Name of the rater identifier column.
`unit`	Names of columns jointly identifying a rated unit, such as `c("Person", "Task")`.
`score`	Name of the score column. Values may be numeric, character, factor or logical, with `NA` for missing scores. Factors use their labels.
`categories`	Required distinct permitted category values, including unused categories, in their substantive order. At least two are required. They must have the same numeric/character/logical representation as scores. With linear/quadratic weights, numeric values use their actual spacing; character/logical values use equally spaced positions in this vector. Factor levels never silently determine the order. To treat numeric codes as equally spaced ranks, explicitly convert scores and categories to character. Choose spacing on substantive grounds.
`weights`	One of `"quadratic"` (default), `"linear"` or `"unweighted"`. Agreement weights are 1 minus squared normalized distance, 1 minus normalized absolute distance, or the identity matrix, respectively.

Details

Observed agreement is the equally weighted mean of within-unit weighted agreement over distinct ordered rating pairs, using only units with at least two nonmissing ratings. Category proportions are the equally weighted mean of within-unit category proportions over all units with at least one rating. Thus singletons contribute to expected agreement but not observed agreement; units with no ratings contribute to neither. This is Gwet's raw-rating convention, not Krippendorff's pairable-rating margins or Cohen's separate rater margins. Unequal rating counts do not give units greater weight.

With q declared categories, agreement matrix W and mean category proportions pi, expected agreement is `sum(W) * sum(pi * (1 - pi)) / (q * (q - 1))`. AC is `(observed - expected)/(1 - expected)`. Quadratic/linear weights give AC2; identity weights give AC1. At two categories the supplied linear and quadratic weights equal identity weights, so the numerical results coincide. The complete category universe is required: adding unused categories can change expected agreement and AC. It must not be selected to improve a result.

Negative estimates are retained. No pairable units or a nonpositive chance denominator gives `NA` with a status. With all raters always using the same category, AC can be 1 when the declared universe has at least two categories; this does not establish discrimination, validity or reference accuracy. Unlike kappa, its chance model does not become undefined in that case.

Repeated rater-unit cells are rejected, even when scores are missing. Audits distinguish explicit missing scores from absent assignments relative to the observed rater roster. Units with no input rows are unknown. Missingness and assignment may still bias interpretation. Results are descriptive point estimates: no confidence intervals, tests, quality bands or invariance to prevalence are claimed. Different chance models answer different questions; AC2 is not an automatic replacement or correction for kappa. Dense weight storage is quadratic in the declared category count. Custom weights and circular, bipolar, radical and ratio weights are not implemented.

Value

A plain list with `table` (coefficient, observed/expected agreement, status and counts), `counts` (category totals and unit-mean proportions), `categories`, `weights`, `cases` (input rows and their two inclusion flags), `unit_summary`, `units`, `settings` and `notes`. `Unit` indexes `units`. Results can be saved with `base::saveRDS()` without refitting on reload.

References

Gwet, K. L. Author implementation of AC1/AC2, `gwet.ac1.raw`, and linear/quadratic weights, in *irrCAC*. https://github.com/kgwet/irrCAC/blob/master/R/agree.coeff3.raw.r https://github.com/kgwet/irrCAC/blob/master/R/weights.gen.r

See Also

`pairwise_kappa_table()`, `krippendorff_alpha_table()`, `pairwise_icc_table()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("gwet_ac2_table", run.dontrun = TRUE)
## Not run:
data("mfrmr_example_agreement", package = "mfrmr")
ratings <- mfrmr_example_agreement
ac <- gwet_ac2_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", categories = 0:3)
ac$table
ac$unit_summary # Number of observed ratings and pairability for each response.
ac$notes # Read the panel coefficient together with its assumptions.

## End(Not run)
```

`import_erm_fit` *Import an* `eRm` *fit to an mfrmr-compatible bundle*

Description

Extracts item / person parameters from an `eRm::PCM()`, `eRm::RM()` or `eRm::RSM()` fit. Source cumulative easiness coefficients are converted to absolute adjacent-category difficulties. One item location is returned per item: the mean of its thresholds. Source identification is retained. Current `eRm` person tables use `Person Parameter` and `Std.Error`; historical `theta` / `thetapar` estimate labels are also accepted. Unknown or internally misaligned person-table schemas stop with an explicit error rather than silently recycling rows. Same caveats as `import_mirt_fit()`.

Usage

```
import_erm_fit(fit, model = c("RSM", "PCM", "GPCM"), item_facet = "Item")
```

Arguments

`fit`	An object returned by `eRm::PCM()`, `eRm::RM()`, or `eRm::RSM()`.
`model`	Matching `"PCM"` or `"RSM"` label; either is accepted for a binary `RM`. `"GPCM"` and linear extensions are unsupported.
`item_facet`	Name to assign to the item facet.

Details

Item-location SEs use the corresponding scalar transformation of the source cumulative coefficient SE. Person maximum-likelihood estimates and conditional SEs retain source conventions, including labelled extreme-score extrapolations. The original coefficient table is retained in `source$native_parameters`.

Value

An `mfrm_imported_fit` object.

Imported uncertainty

Imported SEs retain the source package's interpretation. The measurement-side diagnostics do not reconstruct the joint parameter covariance, so joint facet chi-square statistics, degrees of freedom and p-values are unavailable. Posterior SDs do not supply sampling SEs for separation reliability. Other separation summaries require valid SEs for every finite estimate and remain descriptive. Imported Wright maps show points only: source uncertainty conventions do not establish one common confidence-interval calculation. Re-import older saved bundles from the existing source-package fit to update difficulties, thresholds and uncertainty labels. The mirt and TAM importers accept `compute_fit = TRUE` when source fit statistics are needed; no model re-estimation is required.

Scope

Use summary() for source-scale tables and plot() for a point-only Wright map. Available source fit statistics remain in the facet and diagnostic tables. Native model curves, comprehensive mfrm_results() reports, response-level diagnostics, run_qc_pipeline(), bias/DIF analysis, anchoring and portable calibration are unavailable for imported bundles. This is a one-way fitted-object import of the documented fields.

See Also

import_mirt_fit(), import_tam_fit()

Examples

```
if (requireNamespace("eRm", quietly = TRUE)) {
  response_matrix <- matrix(sample(0:3, 60, replace = TRUE), nrow = 20)
  colnames(response_matrix) <- paste0("Item", seq_len(ncol(response_matrix)))
  fit <- eRm::PCM(response_matrix)
  imported <- import_erm_fit(fit, model = "PCM")
  imported$summary
}
```

---

import_mirt_fit *Import an* mirt *fit to an mfrmr-compatible bundle*

Description

Extracts item, step, and person parameters from a mirt::mirt() fit and returns an mfrm_imported_fit object. The returned object has the public slots summary, facets$person, facets$others, steps, config, and source that the mfrmr plot and table helpers expect. Only unidimensional Rasch and partial-credit response models with positive slopes and ordinary category scores are supported. Graded-response, guessing and multidimensional models are refused. With compute_fit = TRUE, source Infit / Outfit statistics are attached.

Usage

```
import_mirt_fit(
  fit,
  model = c("RSM", "PCM", "GPCM"),
  item_facet = "Item",
  compute_fit = FALSE
)

## S3 method for class 'mfrm_imported_fit'
summary(object, digits = 3L, ...)

## S3 method for class 'summary.mfrm_imported_fit'
print(x, ...)
```

Arguments

fit	An object returned by mirt::mirt() (a SingleGroupClass).
model	One of "RSM", "PCM", "GPCM". The importer does not reconstruct all source constraints; pass the model that was estimated. Non-unit slopes require "GPCM". A polytomous "RSM" import requires source item type "rsm".
item_facet	Name to assign to the item facet in the imported bundle (default "Item").
compute_fit	Logical. When TRUE, run mirt::itemfit() and mirt::personfit() to populate Infit / Outfit / OutfitZSTD columns on the returned facet tables, plus build a measurement-side diagnostics bundle. Person fit uses source EAP scores. Default FALSE extracts parameters without calculating fit statistics.
object, x	An imported measurement bundle.
digits	Number of digits for displayed estimates.
...	Additional arguments (unused by imported summaries).

Value

An mfrm_imported_fit object. Slots:

summary Model / method / N / LogLik / AIC / BIC.

facets$person Person ID, Estimate, SE, Extreme, plus Infit / Outfit / OutfitZSTD / Zh when compute_fit = TRUE.

facets$others Item-level estimates and slopes; with compute_fit = TRUE, also available Infit / Outfit statistics.

steps Absolute adjacent-category thresholds on the source ability scale, labelled in Parameterization; these are not centered step deviations. Rating-scale offsets are included.

config List with the declared model and facet names used for the import; downstream plot and table helpers consult this to dispatch correctly on the imported bundle.

diagnostics mfrm_diagnostics-shape bundle when compute_fit = TRUE; NULL otherwise.

source Imported-from metadata.

Source scale

Item difficulty is the mean of its absolute adjacent-category thresholds. Source identification and slopes are retained without rescaling. For mirt gpcmIRT and rsm, the category offset is included as b - c / a. Binary and polytomous items may be mixed; thresholds are read using each item's response model and category count. Inapplicable padding columns in the source coefficient table are not missing estimated thresholds. Person estimates are EAP; the SE column contains conditional posterior SDs, not sampling SEs. Person labels use retained source row names or P-prefixed row positions. Original identifiers discarded by mirt cannot be recovered. Imported summaries describe these conventions without assuming a native mfrmr population distribution or slope normalization.

Imported uncertainty

Imported SEs retain the source package's interpretation. The measurement-side diagnostics do not reconstruct the joint parameter covariance, so joint facet chi-square statistics, degrees of freedom and p-values are unavailable. Posterior SDs do not supply sampling SEs for separation reliability. Other separation summaries require valid SEs for every finite estimate and remain descriptive. Imported Wright maps show points only: source uncertainty conventions do not establish one common confidence-interval calculation. Re-import older saved bundles from the existing source-package fit to update difficulties, thresholds and uncertainty labels. The mirt and TAM importers accept `compute_fit = TRUE` when source fit statistics are needed; no model re-estimation is required.

Scope

Use `summary()` for source-scale tables and `plot()` for a point-only Wright map. Available source fit statistics remain in the facet and diagnostic tables. Native model curves, comprehensive `mfrm_results()` reports, response-level diagnostics, `run_qc_pipeline()`, bias/DIF analysis, anchoring and portable calibration are unavailable for imported bundles. This is a one-way fitted-object import of the documented fields.

See Also

`import_tam_fit()`, `import_erm_fit()`

Examples

```
# Optional integration example (loads additional modelling or rendering packages).
# example("import_mirt_fit", run.dontrun = TRUE)
## Not run:
if (requireNamespace("mirt", quietly = TRUE)) {
  response_matrix <- matrix(sample(0:1, 120, replace = TRUE), nrow = 40)
  colnames(response_matrix) <- paste0("Item", seq_len(ncol(response_matrix)))
  fit <- mirt::mirt(response_matrix, 1, itemtype = "Rasch", verbose = FALSE)
  imported <- import_mirt_fit(fit, model = "RSM")
  imported$summary
}

## End(Not run)
```

`import_tam_fit` *Import a* `TAM` *fit to an mfrmr-compatible bundle*

Description

Extracts item / step / person parameters from a unidimensional `TAM::tam.mml()` or `TAM::tam.mml.mfr()` fit. Difficulties and absolute adjacent-category thresholds are derived from the source category logits, rather than inferred from coefficient names. Source slopes and scale identification are retained.

Usage

```
import_tam_fit(
  fit,
  model = c("RSM", "PCM", "GPCM"),
  item_facet = "Item",
  compute_fit = FALSE
)
```

Arguments

fit	An object returned by TAM::tam.mml() or TAM::tam.mml.mfr().
model	Declared response model: "RSM", "PCM", or "GPCM". Non-unit source slopes require "GPCM"; import does not reconstruct all source constraints or certify equivalence to a native mfrmr model.
item_facet	Name to assign to the item facet for the single-facet path. Ignored when the input is a multi-facet tam.mml.mfr fit, whose combined response conditions are labelled "DesignCell".
compute_fit	Logical. When TRUE, run TAM::msq.itemfit() and TAM::tam.personfit() to populate Infit / Outfit columns on the returned facet tables, plus build a measurement-side mfrm_diagnostics bundle. Default FALSE.

Details

Each item difficulty is the mean of its absolute adjacent-category thresholds. Positive constant adjacent-category slopes are required. For multi-facet fits, each returned difficulty combines all facet effects for that response condition. Separate facet coordinates are not reconstructed; the original coefficient table remains in source$native_parameters.

A transformed item SE is retained only when the location is a scalar multiple of one source coefficient and its slope is fixed. Otherwise it is missing: marginal coefficient SEs cannot replace the required joint covariance. Persons retain source EAP and conditional posterior SD. Item fit is averaged over source posteriors; TAM::tam.personfit() uses WLE scores. FitBasis records this distinction without replacing the imported EAP estimates. Person Extreme flags use observed responses and each item's fitted category support, including unobserved categories. NObserved counts responses used for this check. ResponseStatus distinguishes observed, entirely missing, and unavailable response information. Entirely missing or unverifiable rows have Extreme = NA; finite posterior scores do not imply nonextreme data. Re-import older TAM objects to recover these flags; saved imported objects without the source responses cannot reconstruct them.

The public imported-fit surface is deliberately unidimensional and MML-only. A tam.jml object is not silently relabelled as MML, and a TAM fit with ndim > 1 is rejected rather than flattened into one mfrmr scale. Keep multidimensional TAM fits in a separate validation workflow.

TAM-native AIC, BIC, and adjusted BIC are retained as explicitly named Native* fields. Compatibility AIC and BIC columns still mirror the native TAM values, but the imported object has ICStatus = "imported_native_descriptive", ICEligible = FALSE, and cannot enter compare_mfrm() as a current mfrmr IC contract. In particular, TAM's native aBIC is not relabelled as the package's Sclove SABIC. The source metadata also retains the TAM version, dimension count, iterations, and iteration ceiling used for the conservative imported convergence status.

Value

An mfrm_imported_fit object. Slots mirror import_mirt_fit(), with explicit TAM-native IC provenance in summary and source.

Imported uncertainty

Imported SEs retain the source package's interpretation. The measurement-side diagnostics do not reconstruct the joint parameter covariance, so joint facet chi-square statistics, degrees of freedom and p-values are unavailable. Posterior SDs do not supply sampling SEs for separation reliability. Other separation summaries require valid SEs for every finite estimate and remain descriptive. Imported Wright maps show points only: source uncertainty conventions do not establish one common confidence-interval calculation. Re-import older saved bundles from the existing source-package fit to update difficulties, thresholds and uncertainty labels. The mirt and TAM importers accept compute_fit = TRUE when source fit statistics are needed; no model re-estimation is required.

Scope

Use summary() for source-scale tables and plot() for a point-only Wright map. Available source fit statistics remain in the facet and diagnostic tables. Native model curves, comprehensive mfrm_results() reports, response-level diagnostics, run_qc_pipeline(), bias/DIF analysis, anchoring and portable calibration are unavailable for imported bundles. This is a one-way fitted-object import of the documented fields.

See Also

import_mirt_fit(), import_erm_fit()

Examples

```
if (requireNamespace("TAM", quietly = TRUE)) {
  response_matrix <- matrix(sample(0:3, 60, replace = TRUE), nrow = 20)
  colnames(response_matrix) <- paste0("Item", seq_len(ncol(response_matrix)))
  fit <- TAM::tam.mml(resp = response_matrix, irtmodel = "PCM")
  imported <- import_tam_fit(fit, model = "PCM")
  imported$summary
}
```

---

interaction_effect_table

Extract model-estimated facet interaction effects

Description

interaction_effect_table() returns the fixed-effect interaction block estimated by fit_mfrm() when facet_interactions is supplied. These are model-estimated deviations from the additive main-effects MFRM, not the residual screening statistics returned by estimate_bias().

Usage

```
interaction_effect_table(fit, comparison_review = NULL)
```

Arguments

fit	An mfrm_fit or mfrm_comparison_fit returned by fit_mfrm().
comparison_review	Optional result of describe_mfrm_data() with comparisons, or its $comparison_review component. Returns the declared interaction contrasts from a matching joint fit instead of individual cells.

Details

mfrmr supports two-way interactions between non-person facets, for example facet_interactions = "Rater:Criterion". Each interaction matrix is identified by zero marginal sums across both participating facets, so the interaction estimates are separable from the two main effects. Positive values indicate higher-than-expected scores for the facet-level combination under the additive model; negative values indicate lower-than-expected scores.

Use this table for confirmatory model review after specifying the facet pair of substantive interest. For exploratory screening without adding parameters to the fitted model, use estimate_bias() or estimate_all_bias().

With comparison_review, first fit the declared model using the review's fit_arguments and the original ratings. The fit must match the retained responses, origins, score coding, model, constraints, facet orientation, step owner and population design. Only native, unadjusted RSM/PCM fits without shrinkage are supported. The contrast is the declared weighted sum of Group1-minus-Group2 interaction effects, in logits. Positive means a relative shift toward higher coded scores; it is not an absolute school effect or evidence of unfairness. Location and step effects are not added.

This route reevaluates the retained likelihood and full gradient without optimization. Objective agreement must be within 1e-6 and the gradient sup norm within max(1e-4, 10 * reltol), matching the ordinary numerical stopping tolerance. For MML it also evaluates 2 * q + 1 quadrature nodes, requiring the finer-grid gradient to meet that tolerance and the absolute negative-log-likelihood change per Person to be at most 1e-6. These are numerical checks, not statistical precision criteria. A failed check retains the row with Estimate = NA and an explanation; the separate OptimizerEstimate is a diagnostic trace. No refit or partial fit is run.

EstimateStatus = "descriptive_only" requires full conditional-map rank, numerical convergence, current readiness information and no recorded blocking condition or boundary exclusion. It does not certify finite-MLE existence, global marginal identification or sampling performance. MML retains State = "not_evaluated" for positive conditional ranks even when a descriptive fitted contrast is available. BoundaryState, FitReadiness, NumericalState and QuadratureState remain separate. SE, Lower, Upper remain missing and FormalInferenceEligible remains false. A target-only identified contrast in a rejected full model remains unavailable through this route. Save the returned table with saveRDS() to preserve its review and checks without recalculation; CSV retains the visible columns but not the attributes.

For a mfrm_comparison_fit created with fit_scope = "comparisons", omit comparison_review: the table already contains the declared targets and their partial-fit checks. Extraction does not fit again. See fit_mfrm() for its likelihood-preserving parameterization, availability requirements and limits. For full RSM/PCM MML fits, mfrm_facet_intervals() with the same comparison_review

separately supplies model-information intervals and supported population-SD standardization; extraction here stays descriptive. CML comparison fits add conditional-information SEs and pointwise 95 percent Wald limits for identified targets, with `conditional_wald` status; unidentified rows remain missing. The ordinary full-fit route above remains unchanged.

Value

A tibble with one row per interaction cell. Returns an empty tibble when the fit has no model-estimated facet interactions. With `comparison_review`, one row per requested contrast with availability, numerical and readiness fields; attributes `comparison_review`, `fit_checks` and `notes` preserve the prospective question, source fingerprint and numerical evidence, and interpretation limits.

See Also

`fit_mfrm()`, `describe_mfrm_data()`, `summary.mfrm_fit()`, `estimate_bias()`, `compare_mfrm()`, mfrmr_linking_and_dff

`mfrm_facet_intervals()` with `interaction` for supported fixed-population RSM/PCM MML cell intervals from full joint information.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("interaction_effect_table", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(
  toy, person = "Person", facets = c("Rater", "Criterion"),
  score = "Score", method = "MML", model = "RSM",
  facet_interactions = "Rater:Criterion",
  quad_points = 7, maxit = 30
)
head(interaction_effect_table(fit))

## End(Not run)
```

---

interrater_agreement_table

Build an inter-rater agreement report

Description

Build an inter-rater agreement report

Usage

```
interrater_agreement_table(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  context_facets = NULL,
  exact_warn = 0.5,
  corr_warn = 0.3,
  include_precision = TRUE,
  top_n = NULL
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
rater_facet	Name of the rater facet. If NULL, inferred from facet names.
context_facets	Optional context facets used to match observations for agreement. If NULL, all remaining facets (including Person) are used.
exact_warn	Warning threshold for exact agreement.
corr_warn	Warning threshold for pairwise correlation.
include_precision	If TRUE, append rater severity spread indices from the facet precision summary when available.
top_n	Optional maximum number of pair rows to keep, after ordering by flag, exact agreement and correlation. The summary still describes all candidate pairs. Leave NULL to retain the complete pair audit.

Details

This helper computes pairwise rater agreement on matched contexts and returns both a pair-level table and a one-row summary. The output is package-native and does not require knowledge of legacy report numbering. When fitted category probabilities are available, expected exact agreement for a matched context is the model-implied quantity $\sum_k P_{r1}(X = k)P_{r2}(X = k)$. It is not a marginal-frequency chance agreement statistic. This product assumes conditional independence at the saved Person point score and fitted structural parameters. MML uses the saved EAP: it does not integrate either the product or its factors over the Person posterior. Calibration and population uncertainty are not propagated. Nonfinite primary Person/facet/step estimates or unavailable GPCM slopes withhold the affected expectations; finite internal optimizer proxies are not substituted and limiting boundary probabilities are not calculated here. ExpectedReason records unavailable primary estimates, repeated ratings, unavailable probabilities or absent matched scores. Observed exact agreement uses equality of the package's observed score categories. Repeated ratings in a rater/context cell are averaged using observation weights; comparisons then describe those means, and expected exact agreement is withheld for the repeated cells. The current function does not translate category positions across multiple independent scales, apply an agreement-based SE inflation, or establish numerical equivalence with FACETS Table 7. For two-way observed-score ICCs without an MFRM fit, use pairwise_icc_table(). That function requires one rating per rater/unit and

an explicit choice of agreement/consistency and single/average ratings. The raw-score kappa, alpha and AC2 helpers also reject repeated rater-unit cells; they do not inherit this function's averaging convention. See mfrmr_methods_reference for choosing among these targets.

Within a pair, every matched context with two finite scores contributes one opportunity, after any within-cell averaging. Observation weights affect that averaging, not the weight assigned to the resulting context. Overall exact/adjacent agreement divides the corresponding counts by the total opportunities across pairs. A context with finite compared scores from m judges contributes m(m-1)/2 opportunities, so this is not equal weighting of distinct contexts or independent evidence from that many people. Pairwise correlation is computed from the matched scores, and `MeanCorr` weights each available correlation by its pair's matched-context count. It is not a panel ICC.

Value

An `mfrm_interrater` bundle (a named list) with:

- `summary`: one-row inter-rater summary
- `pairs`: pair-level agreement table
- `notes`: interpretation and availability qualifications
- `settings`: applied options and thresholds

Use `summary(ir)` or `ir$summary` to inspect a result and `plot(ir)` for its saved agreement figure. Save the full bundle with `base::saveRDS()` and reopen with `base::readRDS()`; export `ir$pairs` and `ir$summary` separately with `utils::write.csv()`. Reopening performs no fitting. `plot(ir, type = "agreement", metric = "correlation")` selects the correlation view; `metric = "difference"` selects observed-score gaps. These plots retain the saved flag thresholds. Changing them requires recreating this result from the existing fit, not fitting a new model. `as_ggplot()` converts a saved plot payload without recomputation.

Interpreting output

- `summary`: overall agreement level, number/share of flagged pairs.
- `pairs`: pairwise exact agreement, correlation, and direction/size gaps.
- `settings`: applied facet matching and warning thresholds.

Flags indicate configured review thresholds, not rater quality, fairness or an automatic training priority. A missing rule remains unavailable; a known cutoff crossing still flags the pair. Overall flag rates require complete classification. Expected agreement is withheld if any matched context lacks valid category probabilities; availability counts are retained. Recreate older agreement/network results from the existing fit and matching diagnostics with the original settings, then regenerate plots and exports.

Typical workflow

1. Run with explicit `rater_facet` (and `context_facets` if needed).
2. Review `summary(ir)` and top flagged rows in `ir$pairs`.
3. Visualize with plot(x, type = "agreement").

Output columns

The `pairs` data.frame contains:

Rater1, Rater2 Rater pair identifiers.

N, OpportunityCount Number of matched contexts with two finite scores for this pair, after any repeated-cell averaging.

Exact Proportion of exact score agreements.

ExpectedExact Model-implied expected exact agreement from the two raters' fitted category-probability vectors. `NA` when those probabilities are unavailable.

Adjacent Proportion whose absolute compared-score difference is at most 1, including exact agreements.

MeanDiff Signed mean score difference (Rater1 - Rater2).

MAD Mean absolute score difference.

Corr Pearson correlation between paired scores.

Flag Logical; `TRUE` when Exact < `exact_warn` or Corr < `corr_warn`.

ExactCount, AdjacentCount Observed counts behind the proportions.

ExpectedExactCount Sum of model-implied exact-agreement probabilities over matched contexts; it may be fractional. Unavailable if any such context lacks valid probabilities.

ExpectedAvailableContexts, ExpectedUnavailableContexts Counts of matched contexts with and without valid fitted probabilities.

ExpectedReason `available`, or semicolon-separated reasons: `primary_estimate_unavailable`, `repeated_ratings`, `fitted_probabilities_unavailable`, or `no_matched_scores`.

The `summary` data.frame contains:

RaterFacet Name of the rater facet analyzed.

Pairs, AvailablePairs, UnavailablePairs All candidate rater pairs, those with matched scores, and those without.

TotalPairs, OpportunityCount Total matched-context opportunities across rater pairs, not the number of distinct people or contexts.

Contexts Distinct contexts with at least two finite compared scores.

ExactAgreement Exact agreements divided by all matched-context opportunities.

AgreementMinusExpected Observed exact agreement minus expected exact agreement.

MeanCorr Mean available pairwise correlation weighted by each pair's matched-context count; unavailable correlations are excluded.

FlaggedPairs, FlaggedShare Known flagged pairs and their share of all pairs; the share is unavailable if classification is incomplete.

ClassifiedPairs, UnclassifiedPairs Available and unavailable combined flag decisions. Correlation availability is recorded separately.

RaterSeparation, RaterReliability Severity-spread indices for the rater facet, reported separately from agreement.

See Also

diagnose_mfrm(), facets_chisq_table(), plot(x, type = "agreement"), pairwise_icc_table(), pairwise_kappa_table(), gwet_ac2_table(), krippendorff_alpha_table(), mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("interrater_agreement_table", run.dontrun = TRUE)
## Not run:
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Compare ratings of the same person on the same criterion
agreement <- interrater_agreement_table(fit, rater_facet = "Rater")
agreement$summary
agreement$pairs[, c("Rater1", "Rater2", "N", "Exact", "Corr", "MeanDiff")]
# N is the number of matched ratings; Exact is the fraction with identical scores
# MeanDiff = Rater1 minus Rater2: positive means Rater1 assigned higher scores
# These observed-score comparisons are distinct from fitted rater severity

## End(Not run)
```

---

krippendorff_alpha_table

Compute Krippendorff's alpha across raters

Description

Estimate observed-score agreement across two or more raters, allowing unequal numbers of ratings per unit. This is Krippendorff's agreement coefficient, not Cronbach's internal-consistency alpha or MFRM separation reliability. No measurement-model fitting is performed.

Usage

```
krippendorff_alpha_table(
  data,
  rater,
  unit,
  score,
  level = c("nominal", "ordinal", "interval", "ratio"),
  categories = NULL
)
```

Arguments

`data`	Long-format data with at most one row per rater and unit.
`rater`	Name of the rater identifier column.
`unit`	Names of columns jointly identifying a rated unit, for example `c("Person", "Task")`. They match observations, not variance components.
`score`	Name of the score column. Missing scores are `NA`. Nominal/ordinal scores may be numeric, character, factor or logical; factors compare by labels. Interval/ratio scores must be finite numeric values, and ratio scores must be nonnegative with a meaningful zero.
`level`	Measurement level: `"nominal"` (default), `"ordinal"`, `"interval"` or `"ratio"`. Choose on substantive grounds; numeric category codes alone do not establish an interval or ratio scale.
`categories`	For ordinal scores, the required distinct category values in their substantive order. Factor levels and numeric spacing never silently set this order. For nominal scores, optional permitted values; if `NULL`, observed nonmissing values are used. Values must have the same numeric/character/logical representation as `score` (factors use labels). Unused categories are allowed. Leave `NULL` for interval/ratio scores.

Details

The coefficient is $1 - D_o/D_e$. Only units with at least two nonmissing scores contribute to either disagreement. A unit with m ratings contributes its ordered rating pairs divided by m-1 to the coincidence counts, hence contributes m pairable ratings overall. Units are not equally weighted when m differs. The expected disagreement draws two distinct observations without replacement from the pooled pairable ratings, with denominator n(n-1), where n is the number of pairable ratings. This is not Cohen's kappa chance model or MFRM model-implied expected agreement.

The squared difference function is 0/1 mismatch for nominal scores, frequency-based rank distance for ordinal scores, squared numeric difference for interval scores, and $((c-k)/(c+k))^2$ for ratio scores (0 when both values are zero). Ordinal positions use cumulative pairable-category counts minus half the category count, divided by n. The squared difference between these positions is proportional to Krippendorff's ordinal distance. It is not quadratic weighting of category numbers or equally spaced category ranks. Unused ranks and singleton-only categories do not alter pairable distances.

Interval distances are calculated after subtracting `score_offset`, dividing by `score_half_range`, and dividing by 2, using only pairable values to set the offset and half-range. Constant pairable values map to zero. This avoids squaring very large or tiny raw scores. The common normalization leaves alpha unchanged; reported disagreements are on this normalized scale, not original squared score units. Ordinal positions are similarly normalized by n. Nominal and ratio disagreements use their stated distance directly. Computation avoids a dense category-by-category matrix. Ratio computation still requires quadratic time in the number of distinct pairable values.

No rows are imputed and repeated rater-unit cells are rejected, including repetitions with missing scores. All input rows remain in `cases`. `unit_summary` distinguishes missing scores on assigned rows from absent assignments relative to the observed rater roster. Units with no input rows at all are unknown to the function. Missingness and selective assignment may affect representativeness; numerical handling does not correct that bias.

Negative estimates are retained. With no pairable ratings or zero expected disagreement, alpha is unavailable with a reason rather than set to 0 or 1. A finite coefficient describes this pooled rating sample, not each rater's accuracy against a reference or a guarantee of interchangeable ratings. There are no confidence intervals, tests, quality cutoffs or bootstrap claims. Dependencies among several tasks from one person require attention in any subsequent sampling inference. Circular, bipolar and custom metrics are not implemented.

Value

A plain list with `table` (alpha, observed/expected disagreement, availability and sample counts), `counts` (category values, nonmissing and pairable counts, plus ordinal positions), `cases` (input rows and exclusion reasons), `unit_summary` (assignment counts and unit disagreements), `units` (unique input unit keys), `settings` and `notes`. `Unit` indexes rows of `units`. Save with `base::saveRDS()`; export data frames with `utils::write.csv()`. Reopening the saved list performs no calculation.

References

Krippendorff, K. (2011; literature updated 2013). *Computing Krippendorff's Alpha-Reliability*. Sections A-E, pp. 1-10. https://www.asc.upenn.edu/krippendorffs-alpha-reliability

See Also

`pairwise_icc_table()`, `interrater_agreement_table()`, `score_comparison_table()`

Examples

```
data("mfrmr_example_agreement", package = "mfrmr")
ratings <- mfrmr_example_agreement
agreement <- krippendorff_alpha_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", level = "ordinal", categories = 0:3)
agreement$table
agreement$unit_summary # Each response needs at least two observed ratings.
agreement$notes
# Ordinal alpha uses category ordering; it is not quadratic weighted kappa.
```

---

launch_mfrmr_viewer *Launch a local Shiny viewer for an mfrm_results object*

Description

`launch_mfrmr_viewer()` opens a local Shiny app for reading the object returned by `mfrm_results()`. It is intentionally a viewer over an existing comprehensive results object, not a new estimation interface.

Usage

```
launch_mfrmr_viewer(
  x,
  top_n = 100L,
  launch.browser = TRUE,
  port = NULL,
  host = getOption("shiny.host", "127.0.0.1"),
  display.mode = c("auto", "normal", "showcase"),
  return_app = FALSE,
  ...
)
```

Arguments

x	An mfrm_results() object.
top_n	Maximum number of table-index rows shown in the summary payload.
launch.browser	Passed to shiny::runApp().
port	Optional port passed to shiny::runApp(). NULL lets Shiny choose its default.
host	Host passed to shiny::runApp(). Defaults to the local host.
display.mode	Passed to shiny::runApp().
return_app	Logical; if TRUE, return the Shiny app object without running it. This is useful for embedding or testing.
...	Additional arguments passed to shiny::runApp() when return_app = FALSE.

Details

Testlet and random-rater results are not supported by this viewer. Use mfrm_report(res, output = "html") or export_mfrm_results() for their static results and model-specific figures.

The viewer assumes that fitting, diagnostics, and section selection have already happened through mfrm_results(). This keeps GUI exploration separate from reproducible analysis setup: the Replay tab displays the mfrm_results() replay script stored in the result object.

The app includes tabs for overview/triage, QC evidence, APA-style report text when include = "publication" or "apa" was used, available bias screens, pathway plotting, unexpected-response inspection, generic tables, generic plot routes, and replay code. QC, Report, Bias, and Pathway/Misfit tabs show local section-status tables so unavailable or not-requested sections are visible where users look for them. Bias-interaction follow-up still requires an explicit facet-pair decision outside the viewer.

shiny is an optional dependency. Install it before using this viewer: install.packages("shiny").

Value

Invisibly returns the value from shiny::runApp(), or the Shiny app object when return_app = TRUE.

See Also

mfrm_results(), mfrm_results_interactive()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "JML",
  maxit = 30
)
res <- mfrm_results(fit, include = c("fit", "diagnostics", "tables"))

if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  launch_mfrmr_viewer(res)
}
```

`list_mfrmr_data` *List packaged ordered-response MFRM datasets*

Description

List packaged ordered-response MFRM datasets

Usage

```
list_mfrmr_data(details = FALSE)
```

Arguments

`details`	If `FALSE` (default), return the backward-compatible character vector of dataset keys. If `TRUE`, return a catalog describing the intended teaching role and design of each dataset accepted by `load_mfrmr_data()`.

Details

Use this helper when you want to select packaged data programmatically (e.g., inside scripts, loops, or interactive-application wrappers). For the separately packaged G-theory and agreement tables loaded with `data()`, see mfrmr_gtheory_data and mfrmr_agreement_data. These use different input structures and are not keys for `load_mfrmr_data()`.

Typical pattern:

1. call `list_mfrmr_data()` to see available keys.
2. pass one key to `load_mfrmr_data()`.

Value

A character vector of dataset keys accepted by `load_mfrmr_data()`, or, when `details = TRUE`, a data.frame with `Key`, `Rows`, `Persons`, `Raters`, `Criteria`, `CountBasis`, `PrimaryUse`, `Design`, and `Empirical`.

Interpreting output

With `details = FALSE`, returned values are canonical dataset keys accepted by `load_mfrmr_data()`. With `details = TRUE`, use `PrimaryUse` and `Design` to distinguish the applied teaching example from idealized, planted-effect diagnostic, and larger sparse-design datasets. `CountBasis` states whether person/rater counts use raw labels or Study-prefixed labels. Every bundled dataset is synthetic rather than empirical.

Typical workflow

1. Capture keys in a script (`keys <- list_mfrmr_data()`).
2. Select one key by index or name.
3. Load data via `load_mfrmr_data()` and continue analysis. Treat a `combined` key as a design-review object, not as a direct-fit example.

See Also

`load_mfrmr_data()`, ej2021_data

Examples

```
library(mfrmr)

# See the names accepted by load_mfrmr_data()
list_mfrmr_data()

# Compare the purpose and design of the available datasets
catalog <- list_mfrmr_data(details = TRUE)
catalog[, c("Key", "PrimaryUse", "Design")]

# Choose one dataset
ratings <- load_mfrmr_data("example_operational")
head(ratings)
```

`load_mfrmr_data` *Load a packaged ordered-response MFRM dataset*

Description

Load a packaged ordered-response MFRM dataset

Usage

```
load_mfrmr_data(
 name = c("example_core", "example_bias", "example_operational", "study1", "study2",
    "combined", "study1_itercal", "study2_itercal", "combined_itercal")
)
```

Arguments

name	Dataset key. One of the values from list_mfrmr_data(). If omitted, the backward-compatible default is "example_core"; new code should pass a key explicitly.

Details

For a first MFRM analysis, run ratings <- load_mfrmr_data("example_operational"). This reads data already included in mfrmr; no CSV, download or data-generation code is needed. ratings is a name you choose for the returned table. It is not a second dataset to create. Older examples may name the same object toy or d. Start by reading head(ratings), names(ratings) and table(ratings$Score).

load_mfrmr_data("<key>") is the canonical loader for packaged ordered-response MFRM datasets and the entry point used across the package help and vignettes. The equivalent base-R alternative data("<object-name>", package = "mfrmr") remains available for users who prefer the full data() spelling; both paths return identical long-format data frames.

All returned datasets include the core long-format columns Study, Person, Rater, Criterion, and Score. Some datasets, such as the packaged documentation examples, also include auxiliary variables like Group for DIF/bias demonstrations. The supplied ratings are synthetic teaching data, not empirical records. For observed-score G-theory examples with multiple score columns, see mfrmr_gtheory_data; for agreement examples, see mfrmr_agreement_data. Those datasets use data() because their row/column roles differ from the ordered-response MFRM tables returned here.

Value

A data.frame in long format: each row is one rating, not one Person.

Interpreting output

The return value is a plain long-format data.frame. The example and study-specific keys are ready for fit_mfrm() after checking role and score mappings. The combined keys are design-review objects: overlapping IDs or simple Study-based prefixes do not establish a common measurement scale, so an explicit identity, anchor, or linking design is required before a joint fit is interpretable.

Typical workflow

1. list valid names with list_mfrmr_data().
2. load one dataset key with load_mfrmr_data(name).
3. fit a model with fit_mfrm() and inspect with summary() / plot().

See Also

list_mfrmr_data(), ej2021_data

Examples

```
library(mfrmr)

# Load synthetic ratings and inspect their columns and score frequencies
ratings <- load_mfrmr_data("example_operational")
head(ratings)
names(ratings)
table(ratings$Score)

# Optional: base R's data() loads the same dataset under its full name
data("mfrmr_example_operational", package = "mfrmr")
identical(ratings, mfrmr_example_operational) # TRUE: both contain the same data
```

---

make_anchor_table *Build a candidate direct-anchor table from fitted estimates*

Description

Build a candidate direct-anchor table from fitted estimates

Usage

```
make_anchor_table(
  fit,
  facets = NULL,
  include_person = FALSE,
  digits = NULL,
  readiness_policy = c("error", "review")
)
```

Arguments

fit	Output from fit_mfrm().
facets	Optional subset of facets to include.
include_person	Include person estimates as candidate anchors. Use only when cross-run person identity and the intended longitudinal constraint are substantively justified. Fixed Person values are supported only by a receiving JML fit; MML rejects Person anchors.
digits	NULL (default) preserves the fitted numeric values. A nonnegative integer explicitly rounds them for display or external use.
readiness_policy	How a source fit that is not inference-ready is handled. "error" (default) refuses anchor export or reuse. "review" permits explicit review-only extraction; those values must not be used as operational anchors.

Details

This function performs a mechanical conversion from fitted estimates to the Facet/Level/Anchor schema accepted by fit_mfrm(). The returned rows are candidate direct constraints, not an approved anchor set. Anchor uses the internal, untransformed location metric. The table does not include category thresholds, GPCM slopes, model settings or source SEs; it is not a complete saved calibration. See extract_mfrm_calibration() for that separate workflow. Fixing an estimated value does not propagate its source uncertainty into the new fit.

By default, the function refuses export when the source fit is not inference-ready under the current readiness contract. Set readiness_policy = "review" only to inspect candidate values; this does not make them eligible for reuse. The function cannot verify cross-run element identity or invariance. Before reuse, document why selected elements retain the same meaning, check the observed design's connectedness, and run review_mfrm_anchors() on the receiving data. That review checks schema and receiving-data support; it does not validate the source fit or the substantive invariance assumption.

Keep digits = NULL for recalibration. Rounding individual anchors can break their joint sum or group-mean constraint, even at six decimal places. Explicitly rounded values are different constraints and must be reviewed before reuse; the fitter does not silently recenter supplied anchors.

Value

A data.frame with Facet, Level, and Anchor.

Interpreting output

- Facet: facet name to be anchored in later runs.
- Level: specific element/level name inside that facet.
- Anchor: fixed logit value (rounded only when digits is supplied).

Typical workflow

1. Fit and diagnose a defensible reference run with fit_mfrm().
2. Confirm source readiness, element identity, and the intended link.
3. Export candidates with make_anchor_table(fit) and review them.
4. Pass selected rows back into fit_mfrm(..., anchors = ...).

See Also

fit_mfrm(), review_mfrm_anchors()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("make_anchor_table", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(
  ratings, "Person", c("Rater", "Criterion"), "Score",
```

```
  method = "MML", rating_min = 1, rating_max = 4,
  category_policy = "preserve"
)
# Select a facet deliberately; matching labels alone do not prove invariance.
anchors_tbl <- make_anchor_table(fit, facets = "Rater")
anchors_tbl

## End(Not run)
```

`mantel_haenszel_table` *Compare binary responses between matched groups*

Description

Compute a Mantel–Haenszel (MH) odds ratio for each declared target, with conditional exact or asymptotic inference. Matching strata are supplied explicitly; this function does not estimate ability or fit an MFRM.

Usage

```
mantel_haenszel_table(
  data,
  person,
  target,
  score,
  group,
  strata,
  groups,
  inference = c("exact", "asymptotic"),
  correct = TRUE,
  conf_level = 0.95,
  p_adjust = "holm"
)
```

Arguments

`data`	Long-format data with at most one row per Person and target.
`person`	Name of the independent Person identifier column.
`target`	Name of the item or other target identifier column.
`score`	Name of a numeric column containing only 0, 1 or missing values. No automatic dichotomization or category recoding is performed.
`group`	Name of the group column. Each Person must belong to one group.
`strata`	Names of one or more columns jointly defining exact matching strata, for example `c("Total", "Plan")`. Numeric values are not binned. Strata can differ across targets. Missing strata exclude that response.

groups	Two distinct group labels in reference, focal order. Data containing other groups are rejected rather than silently filtered.
inference	"exact" (default) uses the conditional distribution given each stratum's margins. "asymptotic" uses the MH chi-squared approximation.
correct	Logical; request the continuity correction convention of stats::mantelhaen.test() for asymptotic inference. Ignored for exact inference; the resolved setting is saved. This does not correct for repeated ratings, matching errors or multiple testing.
conf_level	Confidence level strictly between 0 and 1. Intervals are pointwise, not simultaneous across targets.
p_adjust	A stats::p.adjust() method, default "holm". The family includes every observed target, including targets with unavailable tests.

Details

An odds ratio above 1 means higher odds of score 1 in the reference group. MHOddsRatio is always sum(A*D/N)/sum(B*C/N), with reference successes A, reference failures B, focal successes C and focal failures D. No .5 is added to zero cells. Zero and infinite estimates are retained.

OddsRatio and its interval follow the selected inference: conditional maximum likelihood and an exact equal-tail interval for "exact", or the MH estimate and a large-sample log-odds interval for "asymptotic". Exact two-sided p-values sum null probabilities no greater than the observed statistic's probability, as in stats::mantelhaen.test(). They need not invert the equal-tail interval. Asymptotic boundary estimates can have an unavailable interval even when the test and point estimate are available. No method is substituted automatically. Computation failures and warnings remain visible in Status/Message. Exact refers to the conditional distribution; numerical inversion uses the standard R routine's root-finding precision. Its asymptotic continuity correction subtracts .5 only when the absolute count contrast is at least .5. Exact p-values exceeding 1 by at most 64 machine epsilons from rounding are returned as 1; larger violations remain unavailable. Asymptotic count products use double precision to avoid integer overflow in large strata. This preserves the MH formulas and does not change the independent-Person or large-sample assumptions.

A stratum contributes information only when both groups and both score categories occur. Other strata are retained with their counts and a reason but contribute neither odds-ratio terms nor inference. This also removes empty and singleton strata before calling the standard R calculation. A single informative stratum is supported. cases retains every input row, distinguishing missing responses/matching from uninformative strata. Duplicate Person-target cells, including missing-score duplicates, are rejected. Multiple targets may use the same Persons; dependence across targets does not invalidate Holm adjustment of valid individual p-values.

The inferential model requires independent Persons within each target and a common conditional odds ratio across its strata. Small cells alone do not invalidate the exact conditional calculation, but exact inference does not establish these assumptions. The separate stratum odds ratios help reveal opposing effects that a common MH effect can conceal; no test of homogeneity, power guarantee, ETS label or automatic DIF/bias flag is given. The number of independent Persons is not the number of rating rows.

Interpreting conditional association as DIF additionally requires adequate matching and comparable exposure. With a common binary Rasch item set and invariant other items, the total including the studied item eliminates the group ability distribution from that item's conditional response.

A rest score generally does not have this property. Totals from different rater/task plans are not automatically comparable: supply justified joint strata or restrict to a defensible common plan. Strata without group overlap are visibly excluded. The function verifies data structure, not the validity of a matching variable, invariance of other items, absence of informative missingness, or independence beyond the supplied Person IDs.

A rater scoring several criteria for the same Person produces repeated Person-rater observations and cannot be pooled through this interface. Renaming those observations as new Persons does not establish independence. This is a separate conditional association analysis, not a new `method` for `analyze_dff()`. Residual and linked-refit contrasts have different targets; their agreement does not certify robustness or fairness.

Value

A plain list with `table` (one row per target), `strata` (2-by-2 counts and stratum odds ratios), `cases` (input-row inclusion audit), `matching` (unique complete stratum keys), `settings` and `notes`. `Stratum` indexes rows of `matching`. `Included` means inclusion in the conditional calculation, not proven substantive comparability. `PAdjusted` uses the family specified above; interval endpoints are not multiplicity-adjusted. Numerical availability does not certify a DIF inference. Save with `base::saveRDS()` and export each data frame with `utils::write.csv()`; reading these outputs performs no calculation.

References

Holland, P. W., & Thayer, D. T. (1986). *Differential item functioning and the Mantel-Haenszel procedure*. ETS Research Report RR-86-31, pp. 7–18. doi:10.1002/j.23308516.1986.tb00186.x.
See `stats::mantelhaen.test()` for the numerical inference conventions.

See Also

`ordinal_logistic_table()`, `analyze_dff()`, mfrmr_linking_and_dff

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mantel_haenszel_table", run.dontrun = TRUE)
## Not run:
# One binary target per Person; both groups share two declared strata.
d <- data.frame(Person = paste0("P", 1:16), Item = "A",
  Group = rep(c("Reference", "Focal"), each = 8),
  Matching = rep(rep(c("Low", "High"), each = 4), 2),
  Score = c(0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1))
mh <- mantel_haenszel_table(d, "Person", "Item", "Score", "Group",
                            "Matching", c("Reference", "Focal"))
mh$table
mh$strata

## End(Not run)
```

`measurable_summary_table` *Build a measurable-data summary*

Description

Build a measurable-data summary

Usage

```
measurable_summary_table(fit, diagnostics = NULL)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.

Details

This helper consolidates measurable-data diagnostics into a dedicated report bundle: run-level summary, facet coverage, category usage, and subset (connected-component) information.

`summary(t5)` is supported through `summary()`. `plot(t5)` is dispatched through `plot()` for class `mfrm_measurable` (`type = "facet_coverage"`, `"category_counts"`, `"subset_observations"`).

Value

A named list with:

- `summary`: one-row measurable-data summary
- `facet_coverage`: per-facet coverage summary
- `category_stats`: category-level usage/fit summary
- `subsets`: subset summary table (when available)

Interpreting output

- `summary`: overall measurable design status.
- `facet_coverage`: spread/precision by facet.
- `category_stats`: category usage and fit context.
- `subsets`: connectivity diagnostics (fragmented subsets reduce comparability).

Typical workflow

1. Run `measurable_summary_table(fit)`.
2. Check `summary(t5)` for subset/connectivity warnings.
3. Use `plot(t5, ...)` to inspect facet/category/subset views.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `summary` data.frame (one row) contains:

Observations, TotalWeight Total observations and summed weight.

Persons, Facets, Categories Design dimensions.

ConnectedSubsets Number of connected subsets.

LargestSubsetObs, LargestSubsetPct Largest subset coverage.

The `facet_coverage` data.frame contains:

Facet Facet name.

Levels Number of estimated levels.

MeanSE Mean standard error across levels.

MeanInfit, MeanOutfit Mean fit statistics across levels.

MinEstimate, MaxEstimate Measure range for this facet.

The `category_stats` data.frame contains:

Category Score category value.

Count, Percent Observed count and percentage.

Infit, Outfit Descriptive category residual mean squares.

InfitZSTD, OutfitZSTD Unavailable (NA); conditioning on the observed category does not preserve a universal unit-residual reference.

ExpectedCount, DiffCount, LowCount Expected-observed comparison and low-count flag.

See Also

diagnose_mfrm(), rating_scale_table(), describe_mfrm_data(), mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("measurable_summary_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
t5 <- measurable_summary_table(fit)
summary(t5)
p_t5 <- plot(t5, draw = FALSE)
p_t5$data$plot

## End(Not run)
```

`mfrmr_agreement_data` *Synthetic ratings illustrating agreement, offsets and missing scores*

Description

Ready-to-load data from the rating-agreement tutorial. Six Persons complete two Tasks, with up to three Raters on each response. Load with `data("mfrmr_example_agreement", package = "mfrmr")`.

Format

A data frame with 35 rows and four columns: `Person`, `Task`, `Rater`, and `Score`. Each row is one rater's rating of one Person's response to one Task. Scores use the declared categories 0–3.

Details

Rater R2 always scores one category above R1, so their rankings agree but their scores differ. R3 has two deliberately different scores, one missing score (P06/T2) and one absent row (P02/T2). There are 34 observed scores. Missing ratings are not zero scores; an absent row alone does not identify whether a rating was planned. Use `unit = c("Person", "Task")` to match responses. `unit = "Person"` would mix two different responses.

These are constructed teaching data, not empirical records, expert standards or evidence of a method's performance in a population.

Source

Existing data from the rating-agreement tutorial. The deterministic construction is retained in `data-raw/make-teaching-data.R` in the source repository.

See Also

`pairwise_kappa_table()`, `krippendorff_alpha_table()`, `gwet_ac2_table()`, `pairwise_icc_table()`, `load_mfrmr_data()`

Examples

```
data("mfrmr_example_agreement", package = "mfrmr")
ratings <- mfrmr_example_agreement
head(ratings)
with(ratings, table(Rater, Score, useNA = "ifany"))
agreement <- pairwise_kappa_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", categories = 0:3)
agreement$table
# Each pair uses its own shared responses; inspect its denominator as well.
```

mfrmr_compatibility_layer

mfrmr Compatibility Layer Map

Description

Guide to the legacy-compatible wrappers and text/file exports in mfrmr. Use this page when you need continuity with older compatibility-oriented workflows, fixed-width reports, or graph/score file style outputs.

This compatibility layer currently applies mainly to diagnostics-based RSM / PCM workflows. Supported GPCM fits also provide graph and package-native scorefile exports through facets_output_file_bundle(), with explicit score-side limitations. Check each helper's model scope. Treat this layer as a presentation/contract surface, not as a claim of FACETS or ConQuest numerical equivalence.

SPSS is treated differently from FACETS and ConQuest: mfrmr currently supports table/data-frame/CSV handoff for SPSS-oriented reporting workflows, but it does not generate SPSS syntax, write native SPSS system files, execute SPSS estimators, or claim SPSS numerical equivalence.

When to use this layer

- You are reproducing an older workflow that expects one-shot wrappers.
- You need fixed-width text blocks for console, logs, or archival handoff.
- You need graphfile or scorefile style outputs for downstream legacy tools.
- You are checking column coverage and metric consistency against a FACETS-style output contract.

When not to use this layer

- For standard estimation, use fit_mfrm() plus diagnose_mfrm().
- For report bundles, use mfrmr_reports_and_tables.
- For manuscript text, use build_apa_outputs() and reporting_checklist().
- For visual follow-up, use mfrmr_visual_diagnostics.

Compatibility map

facets_positioning_guide() User-facing wording for the package's relationship to FACETS. Use before describing compatibility outputs in a report or migration note.

run_mfrm_facets() One-shot legacy-compatible wrapper that fits, diagnoses, and returns key tables in one object.

mfrmRFacets() Alias for run_mfrm_facets() kept for continuity.

build_fixed_reports() Fixed-width interaction and pairwise text blocks. Best when a text-only compatibility artifact is required.

facets_output_file_bundle() Graphfile/scorefile style CSV and fixed-width exports for legacy pipelines.

`write_mfrm_residual_file()` Package-native observation-level residual CSV/TSV export for reviewer, spreadsheet, or external QC handoff.

`write_mfrm_subset_file()` Package-native connected-subset summary and node-membership CSV/TSV export for linking review handoff.

`facets_output_contract_review()` Column and metric review against the FACETS-style output-contract specification. Use only when an explicit output-contract review is part of the task; it checks the package output contract and does not imply external FACETS equivalence.

Preferred replacements

- Instead of `run_mfrm_facets()`, prefer: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()`.
- Instead of `build_fixed_reports()`, prefer: `bias_interaction_report()` -> `build_apa_outputs()`.
- Instead of `facets_output_file_bundle()`, prefer: `category_curves_report()` or `category_structure_report()` plus `export_mfrm_bundle()`.
- For residual or subset handoff, prefer `write_mfrm_residual_file()` and `write_mfrm_subset_file()` over reusing graph/score compatibility exports.
- Instead of `facets_output_contract_review()` for routine QA, prefer: `reference_case_review()` for package-native completeness review or `reference_case_benchmark()` for packaged benchmark cases.

Multiple-imputation review and intervals in 0.2.4

Re-run `pool_mfrm_imputed()` on saved, valid `fit_mfrm_imputed()` analyses to obtain `IntervalReason` and saved inference notes, then recreate plots and exports. No new imputations or fitted analyses are needed for the interval arithmetic update. Numerically unresolved intervals keep their estimates and variances with missing bounds. Older saved tables lack this audit.

`review_mfrm_imputations()` now rejects numeric IDs that collapse to the same text and detects changes to observed numeric predictors without text rounding. Numeric predictors must stay numeric in completed data; integer and double storage are interchangeable. Pooling rechecks the original roster and saved assignment/support review. Correct changed inputs from their original evidence and recreate affected reviews/analyses; do not overwrite original values to make an invalid completion pass.

Expected agreement and accuracy identifiers in 0.2.4

Regenerate `diagnose_mfrm()` from the saved fit, then recreate `interrater_agreement_table()` and derived reports or exports. Calibration refitting is unnecessary. Observed agreement remains available when primary fitted estimates are unavailable; model-expected agreement is withheld instead of using finite optimizer proxies. New `ExpectedReason` fields distinguish these cases from repeated ratings and unavailable probabilities. Older diagnostics do not contain this audit; reports identify its absence and request regeneration before interpreting model expectations. The MML baseline remains a conditional EAP plug-in quantity, not posterior-integrated agreement.

`fit_rater_accuracy()` rejects numeric identifiers whose conversion to text would merge distinct values. Replace them with unambiguous character IDs from the original data before fitting; already rounded text cannot recover the original identities. Previously affected accuracy fits need to be regenerated from corrected input.

Halo-network inference in 0.2.4

Recreate `rater_halo_network_analysis()` results from the original fit and matching saved diagnostics before summary, plotting or export. No model refit is needed. Older results lack the Person-level availability audit and are rejected by summary and plot methods. Correlations remain available when Persons repeat across shared contexts, but ordinary independent-pair p-values are withheld with counts and reasons. Context definitions omitting `"Person"` also provide descriptive correlations only. Use `alpha = 1` for an explicitly descriptive graph without a p-value filter; this does not supply missing inference. Regenerate derived plots and CSV files.

Simulation, assignment review and GPCM guidance in 0.2.4

These old public function names have been replaced, rather than retained as callable aliases. Their help aliases lead to the new topics.

Former call	Current call	What it does
`evaluate_mfrm_recovery(...)`	`simulate_mfrm_recovery(...)`	Generate data repeat
`build_peer_review_design_review(x)`	`review_peer_design(x)`	Read existing peer-as
`gpcm_score_side_contract(status)`	`gpcm_capability_matrix(status, view = "scores")`	Read score availabili
`gpcm_runtime_guard_coverage()`	`gpcm_capability_matrix(view = "unavailable")`	Read blocked/deferre

`assess_mfrm_recovery(x, ...)` keeps its name: it applies review thresholds to an existing simulation result without generating data or refitting. Unused arguments are now rejected, so misspelled options cannot silently leave a requested threshold unapplied. Review thresholds are aids to interpretation, not universal validity criteria.

The simulation and peer-design replacements retain their argument lists, defaults, numerical calculations and result classes. Existing RDS objects still work with `summary()`, supported `plot()` methods and reporting helpers; no re-estimation or class relabelling is needed for these naming changes. Each GPCM view retains its previous columns and status meanings. The default workflow view is unchanged apart from updated route names. Existing scorefile bundles keep the component `gpcm_score_side_contract` and its CSV filename for saved-output compatibility; this field name is not a callable function.

CSV export name in 0.2.4

Replace `export_mfrm(fit, ...)` with `export_mfrm_tables()` using the same arguments. The old function is no longer exported; its help alias opens the replacement's help. CSV filenames, numeric values and the returned file table are unchanged by the rename. Enriched facet CSVs additionally retain diagnostic availability, SE/inference qualifications and reasons, and follow the source facet-table order. Existing fits do not need re-estimation; regenerate older CSVs from the fit and matching diagnostics to include these fields. Do not infer approval from their absence in an older CSV.

`export_mfrm_results()` and `export_mfrm_bundle()` retain their names because they serve different tasks: the former archives an existing results object and reloads its RDS, while the latter prepares fit-level artifacts with an optional refitting recipe. Neither is an alias for the CSV-only exporter. ZIP replacement now includes only files written by the current call; older members are not carried forward. If ZIP creation fails, the prior archive is retained but is not reported as a successful current export. See mfrmr_reports_and_tables for the saving routes.

Practical migration rules

Recompute older `mantel_haenszel_table()` results from their original input data if large strata produced integer-overflow warnings or unavailable odds ratios/tests/intervals. Merely reopening the saved list does not repair it. The corrected calculation preserves the statistical method, matching and multiplicity settings; no MFRM fitting is involved. Regenerate CSV exports from the new result.

- Start FACETS-facing reports with `facets_positioning_guide()` when readers might otherwise assume FACETS numerical reproduction.
- Keep compatibility wrappers only where a downstream consumer truly needs the old layout or fixed-width format.
- For new scripts, start from package-native bundles and add compatibility outputs only at the export boundary.
- Treat compatibility outputs as presentation contracts, not as the primary analysis objects.
- Use `compatibility_alias_table()` when you need to check which aliases are still retained and which package-native names should be used in new code.
- Use `reporting_checklist(fit)$software_scope` to review the current FACETS, ConQuest, and SPSS relationship wording for a fitted analysis.

G/D-study entry points in 0.2.4

Use `mfrm_g_study()` for one or several observed score columns and `mfrm_d_study()` for the resulting assessment plans. The former public names are replaced as follows:

Former call	Current call
`mfrm_multivariate_gstudy(data, scores, ...)`	`mfrm_g_study(data, scores, ...)`
`mfrm_multivariate_d_study(g, plans, weights)`	`mfrm_d_study(g, plans, weights = weights)`
`mfrm_d_study(main_effects, plans, residual_scaling = ...)`	Same call; the main-effects approximation is retained.

The two former long names are no longer exported functions; their help aliases lead to the replacement pages. Constructor arguments, ANOVA/MINQUE defaults, result classes and stored estimates are preserved. Existing G/D RDS files can be reloaded and passed to `mfrm_d_study()`, `summary()`, `plot()` and the same supported interval methods without re-estimation. This naming change alone does not require recreating a saved analysis.

Both D-study routes accept facet labels or `n_<facet>` count columns, for example `n_Rater` and `n_Task`. Existing task/rater `Raters`/`Tasks` names remain accepted for results that use them. Supply one count per facet; duplicate aliases, extra columns and ambiguous matches are errors. Returned count columns retain the source result's convention so plot selections and saved interval identities stay unchanged. A nested child count still means children per parent, not the total number of children.

`summary(d)` returns the coefficient data frame for either route. For a main-effects D-study this replaces generic per-column descriptive summaries; use `summary(as.data.frame(d))` if those summaries were intended. Unsupported arguments to an already estimated G-study are now rejected instead of ignored. Composite weights require the score-component route; `residual_scaling` belongs only to the main-effects approximation. Supplying a fitted MFRM still estimates a separate observed-score main-effects G-study first. Dispatch does not change an estimator or add interval support to that approximation.

Standard R extraction in 0.2.4

Native fits now have documented coefficient, covariance, likelihood and count methods. Previously, `coef(fit)` could fall through to `NULL`, and ordinary MML fits lacked `vcov()` and `logLik()`. Use `coef(fit)` for free coordinates matched to `vcov(fit)`, or `coef(fit, type = "locations")` for labelled saved facet locations. These vectors answer different questions; do not combine a component vector with the free-coordinate covariance without its constraint map. Existing `summary(fit)` tables remain the route to estimates with status and interpretation. CML's existing covariance and likelihood values retain their meanings; the new `coef()` uses the same conditional coordinates.

`confint(fit)` now returns pointwise intervals for those default coefficients for supported native MML/CML fits. Use names or integer positions in `parm` to select coefficients. The matrix carries estimates, SEs, availability reasons and checks in its attributes; comparison-only CML retains missing rows for unidentified targets. JML structural-truth intervals remain unsupported. CML uses saved conditional information; MML applies the same covariance qualification as `vcov()`.

Breaking change for GPCM: replace `confint(fit, ...)` with `confint(fit, parm = "slopes", ...)` when requesting slope intervals, ratios/differences, standardization, sandwich or profile inference, or preparing slope plots/report attachments. The default coefficient matrix uses free log-slope coordinates and is a different target. Supplying slope options without `parm = "slopes"` is rejected. Existing saved slope-interval objects remain reusable; no refit or recomputation is needed merely to change the call syntax. Bootstrap-result and extended-model methods retain their own defaults. See `confint.mfrm_fit()`.

`nobs(fit)` counts Persons for MML/CML and retained ratings for JML. Use `nobs(fit, type = "ratings")` for a response-row count in every method. `logLik(fit)` retains the likelihood basis and information-criterion reason; unqualified ordinary results give `df = NA`, so AIC/BIC remain unavailable. Multiple-model `AIC()`/`BIC()` calls now direct users to `compare_mfrm()` rather than allowing conditional and marginal likelihoods to be mixed. Current native saved fits can use these methods after reloading; source objects are not rewritten. Legacy likelihood metadata, corrected/penalized objectives and unsupported model classes require their documented summaries.

Native ordinary fits also support fitted rating and residual methods. Previously `fitted(fit)` and `residuals(fit)` could silently return `NULL`. They now return retained-row vectors; `fitted(fit, type = "prob")` returns category probabilities. `predict(fit)` has the same source-rating target. `predict(fit, newdata = ...)` is explicitly unsupported; retain `predict_mfrm_units()` or `score_mfrm_calibration()` for ability scoring from supplied responses. These targets must not be interchanged.

MML probabilities are evaluated at the saved EAP, not averaged over its posterior distribution. JML uses primary estimates, including limiting probabilities for extreme Persons. Zero-variance Pearson residuals are missing with a reason. Legacy `diagnose_mfrm()` helpers retain finite JML optimizer proxies and their small-variance cutoff, so values can differ at these limits. Diagnostic observation/score/residual tables now include `PrimaryPersonMeasure`, `PersonMeasureBasis`, `ConditionalVariance`, `PearsonAvailable` and row-specific reasons. Overall/element mean squares are unavailable when any contributing standardized residual, variance or weight is unavailable; `FitAvailableRows`, `FitUnavailableRows` and `FitReason` expose that coverage. Older diagnostics could effectively count missing residuals as zero misfit. Regenerate them and their exports from the saved fit, without refitting. MML response calculations now match Persons by identifier even when the saved Person table is reordered.

Read the `diagnostics`, `settings`, `score_map` and `source_readiness` attributes; values alone do not establish fit or inference readiness. Outputs align with `fit$prep$data`, not an unfiltered

input table. Existing saved fits can be reused without refitting. When old objects lack omitted-row identities, their original `InputRow` positions remain `NA`; retained-row order is still available. CML and comparison-only results now explicitly refuse this unsupported Person plug-in target instead of letting `fitted()`/`residuals()` fall through to `NULL`.

Input normalization corrections in 0.2.4

The shared preparation used by `fit_mfrm()`, `describe_mfrm_data()` and anchor review now rounds scores within `1e-6` of an integer to that integer. Earlier preparation accepted that tolerance but then truncated toward zero: for example, `1.9999999` could become category 1 rather than category 2. The new `score_roundoff_normalized` preparation note records affected retained rows. Genuine fractional responses still require explicit recoding. Corrected JML and two-family GPCM retain their stricter exact-integer input checks; explicitly normalize roundoff before using those specialized fits. Numeric observation weights no longer pass through character formatting; their supplied precision is retained. Out-of-range integer category codes, nonrepresentable category spans and overflowing retained-weight totals are refused before likelihood construction.

Grouped and rowwise tibbles now behave like their ungrouped versions. Only selected model columns enter preparation; grouping on an unused column does not add that column to the design or remove rows for its missing values. This is one combined analysis, not a separate model for each dplyr group.

Facets named `Person`, `Score`, `Weight`, `score_k`, `theta`, `steps`, `interactions`, `log_slopes`, `beta` or `log_sigma2` are now refused before they can collide with internal data or parameter blocks. Rename the facet and update the `facets` argument and any step/slope-owner or anchor references to it. This restriction does not prevent using those names for their intended person, score or observation-weight roles. For example, rename a facet column `steps` to `Stage` and replace `facets = c("Rater", "steps")` with `facets = c("Rater", "Stage")`, also updating any `step_facet`, `slope_facet`, interaction and anchor references. Similar names such as `log_sigma2_group` remain supported. Internal parameter lookup now matches names exactly, preventing those columns from being mistaken for population or slope parameters when those blocks are absent. Older fits using such prefixes should be refitted with the same settings if their calculation entered an affected path, including adaptive MML, where the former partial matching could alter the gradient.

If an older fit was affected by category truncation or a column collision, refit from the original corrected data and regenerate its derived results. Editing `prep$data`, `score_map`, parameter names or identity metadata in a saved fit cannot repair estimates computed from a different response coding. Exact integer inputs with ordinary facet names keep their existing preparation behavior.

Likelihood and convergence corrections in 0.2.4

JML/MML category and posterior normalization now preserve probabilities and log likelihoods when finite log weights share a large offset. Earlier paths could lose the normalizing term through cancellation, even though no exponential overflow occurred. Fits whose estimates or numerical checks entered this regime need refitting with the same data/model and a new numerical review. Saved estimates and derived outputs do not update when the package is updated; do not edit their stored likelihood or readiness flags.

Optimizer code zero no longer gives a numerical pass when the terminal gradient is absent, incomplete or contains NA/NaN/Inf. In that case the result reports `code_zero_unavailable_gradient` and remains review-only. Older saved fits with missing or nonfinite gradient diagnostics must not

be treated as qualified merely because their stored convergence code is zero. Refitting under the current checks is required before relying on their numerical-readiness claim. A finite recorded norm alone cannot establish that an older full gradient was valid, because earlier code discarded nonfinite entries when calculating that norm. These corrections do not change the response model, estimator, quadrature order or tolerance.

Optimizer polishing now rejects candidates with a worse likelihood beyond floating-point rounding error, even if their terminal gradient is smaller. Earlier acceptance used a relative allowance that grew with the objective magnitude. If a saved fit selected such a worse polishing stage, refit with the original data/model/settings and regenerate dependent inference; editing its saved optimizer history does not repair the retained estimates. MML solution checks now return an ineligible result with a reason when likelihood, gradient-norm or information summaries are missing or invalid.

Two-family GPCM/GMFRM EM also uses the stable Person-posterior normalization: large common log likelihoods no longer erase quadrature weights or inflate expected response counts. Its log-slope derivatives and the ordinary GPCM JML/MML derivatives avoid cancellation at large, nearly tied category logits. Refit calibrations affected by these numerical defects with their original data and model settings before reusing inference.

GPCM profile intervals now require positive nuisance curvature at both integration orders, in addition to their existing optimizer, gradient and start-agreement checks. A shared stationary point need not maximize the constrained likelihood. Recompute saved profile intervals with `confint(fit, parm = "slopes", method = "profile", slope = ...)` to apply this additional check; stored intervals do not update automatically. The source calibration can be reused if it remains eligible and was unaffected by the fitting corrections above. An unresolved curvature check leaves missing bounds and a reason; it does not prove that an interval does not exist. Passing these local checks does not establish a global optimum or sampling coverage.

Updating saved ability scores in 0.2.4

New-Person and portable-calibration scoring now remove common log-likelihood offsets before applying quadrature weights. This prevents distorted EAPs and posterior SDs when finite likelihood terms are very large in magnitude. Continuous posterior intervals and cut-score probabilities likewise center the likelihood before adding the prior density; otherwise even an ability-independent likelihood could change a normal-prior interval.

Recompute affected scores with `predict_mfrm_units()` or `score_mfrm_calibration()`, preserving the input rows, calibration, scoring prior, weights, interval level and any cut scores. Regenerate plausible values with the original seed and settings using `sample_mfrm_plausible_values()`. Saved summaries and draws do not update automatically. A scoring-only correction can reuse an eligible unchanged calibration; if its fitted likelihood or convergence checks were affected by the corrections above, refit and review that calibration first. Intervals remain conditional on the point calibration and supplied prior; this correction does not establish frequentist coverage or include calibration uncertainty.

The same likelihood-centering correction applies to ordinary source-Person, testlet and shared-rater continuous scoring. Recompute source scores with `score_mfrm_persons()`. For a supplied roster, use `predict.mfrm_testlet()` or `score_mfrm_random_rater()` with the original complete roster and settings. Recompute affected response diagnostics with `mfrm_response_diagnostics()`. Testlet nested integration and the shared-rater likelihood also correct cancellation at large finite category ties; calibrations affected during estimation require refitting and renewed numerical checks

before reuse. Testlet derivatives now handle a single latent point without dropping a matrix dimension. None of these corrections certify the separate shared-rater Laplace approximation or calibration-adjusted interval coverage.

Updating saved diagnostics in 0.2.4

When a report, plot, bias screen or export accepts both `fit` and `diagnostics`, the diagnostic object must describe that same fitted analysis. Matching person/facet labels or location estimates alone is insufficient: thresholds, population specification, integration settings and retained observations must also agree. A mismatched pair now raises an error instead of combining results or silently substituting newly computed diagnostics.

Corrected diagnostic and bias bundles carry `calculation_version = 2`. Earlier bundles must be recreated from the saved fit, including `estimate_bias()` and derived reports. Corrections affect PTMEA, ZSTD, displacement, chi-square df, PCM fair scores, bias expectations and summary tests, and unexpected-response screening. The diagnostic default is now `fit_df_method = "facets"`. Fit-measure tables and ZSTD fit-pathway plots use this default too; request the explicit engine option for historical df. `whexact = TRUE` now means FACETS WHEXACT=NO; use `"legacy_linear"` for its former calculation. Reproducing the former engine convention also requires `fit_df_method = "engine"`. The former unexpected-response rule is explicit `abs_z_min = 2, rule = "either"`. Pairwise reports now use modified Rasch-Welch df rounded down by default; `df_method = "classical"` retains the former continuous reference. Recreate those reports to obtain `CalculationVersion = 3`. Recompute rather than inserting a version field into an old object. These are diagnostic changes and do not require a main calibration refit.

Older native diagnostics that lack a saved calibration identity are also refused when paired with a fit. For native models supported by `diagnose_mfrm()`, regenerate diagnostics from the original saved fit; the model does not need to be refitted. Reuse the diagnostic options from your original analysis, such as `diagnostic_mode` and `residual_pca`. For example, if the original options were `"legacy"` and `"none"`:

```
fit <- readRDS("fit.rds")
diagnostics <- diagnose_mfrm(
  fit, diagnostic_mode = "legacy", residual_pca = "none"
)
saveRDS(diagnostics, "diagnostics-0.2.4.rds")
report <- facet_quality_dashboard(fit, diagnostics = diagnostics)
```

Function signatures and result-table columns are unchanged by this identity check. Helpers that explicitly accept diagnostics alone, such as `fit_measures_table(diagnostics)`, still support that input; it does not establish a match to another fit. Imported diagnostics must be the bundle stored in the matching imported fit. These checks do not expand the model families supported by ordinary diagnostics; for example, native CML retains its separate interval/reporting workflow in `mfrm_facet_intervals()`.

Updating saved shrinkage results in 0.2.4

Reapply `apply_empirical_bayes_shrinkage()` to the saved fit using the original prior settings, then regenerate plots and exports. No MFRM refit is needed. Updated reports record `CalculationVersion = 2`; calculations avoid artificial zero SEs and retained weights caused by subtracting a shrinkage factor rounded to one.

`shrink_person = TRUE` now requires ordinary JML. For MML or other estimators, use `shrink_person = FALSE`: this removes any previous extra Person adjustment and retains the original Person estimates. MML EAP and posterior SD already describe a posterior distribution; treating them as an observation and sampling SE for another normal update is unsupported. Non-Person facet adjustment remains available within its existing scope.

Updating saved screening results in 0.2.4

Recompute `detect_anchor_drift()` and `build_equating_chain()` from saved fitted waves to apply stable offset/SE calculations and explicit unavailable states. No model refit is required. Singleton or uncomputable residual spreads now remain missing, and summaries distinguish unavailable flags from zero flagged elements. The result records fallback and actual offset contributors; chains also record unavailable cumulative sums. Earlier saved tables are not repaired by loading them. Rebuild chains before requesting offset sensitivity if the saved offsets no longer match the current arithmetic. Regenerate plots and exports: `as_ggplot()` now uses the selected drift, wave estimate, cumulative offset or common-element count instead of the first numeric column.

Rerun `analyze_dff()` with `method = "refit"` from the saved baseline fit and original data to repair earlier subgroup results. This reruns subgroup estimation: failed/skipped groups no longer borrow baseline estimates, absent target levels remain visible, and sparsity/precision screens use retained target-cell counts and shared-Person checks. Current summary, report and plotting methods reject earlier refit objects; do not add the version field by hand. Regenerate exported tables and figures. Location-direction labels now describe parameter differences rather than asserting that entire subgroup response curves are harder or easier.

Recompute older `estimate_bias()` / `estimate_all_bias()` results from the saved fit and matching diagnostics to obtain boundary and complete-cell information/fit checks. Search-boundary estimates remain points, with unavailable SEs and screening tail areas. Small positive response variances contribute to information; missing residuals no longer produce partial-cell Infit/Outfit. Facet labels containing `||` now identify distinct cells correctly. The summary reports missing screening counts and retains unavailable cells in Bonferroni/Holm family sizes. These changes require rescreening, not refitting the calibration; an old saved table is not recalculated by plotting or printing it.

Results from `estimate_bias()`, `estimate_all_bias()`, `unexpected_response_table()`, `unexpected_after_bias_tab` and `displacement_table()` now carry a `fit_identity` field. When a consumer combines a supplied result with a fit, it checks that they describe the same analysis. This includes reports, dashboards, QC, export/replay helpers and `build_misfit_casebook()`. Every retained member and the primary result of a bias collection must match. Matching facet labels alone is insufficient.

Older results without this metadata, and manually assembled bias tables, cannot establish that match. Recreate the result from its saved fit and matching diagnostics; do not add an identity field to an old table by hand. Use the original facets, estimation limits, filtering rules and thresholds. For example, if the original bias screen used these options:

```
bias <- estimate_bias(fit, diagnostics,
  facet_a = "Rater", facet_b = "Criterion",
  max_abs = 10, omit_extreme = TRUE, max_iter = 2, tol = 1e-3)
saveRDS(bias, "bias-0.2.4.rds")
report <- build_apa_outputs(fit, diagnostics, bias_results = bias)
```

The same calls and table columns remain available; the added metadata is stored alongside the tables. Recomputing the screen does not refit the model. Standalone summary/plot methods keep their existing input requirements and do not establish a relationship to a separately supplied fit. These identity checks prevent accidental mixing; they do not validate manually edited numbers or make a screening result eligible for formal inference.

Updating saved predictions in 0.2.4

`predict_mfrm_units()` and `sample_mfrm_plausible_values()` now retain a `fit_identity` field. Fit-level manifests, replay scripts and archive exports require that identity to match the supplied calibration, fitted population coding and source scoring status. Recreate older outputs from the saved fit and original scoring inputs, retaining the original column mappings, `person_data`, prior, integration settings, omission policy and seed. This repeats scoring or drawing plausible values, not estimation of the calibration model. Standalone summary/plot methods retain their existing input rules; a standalone table does not establish a match to another fit.

`predict_mfrm_population()` is a simulation-based design forecast. A result created with `fit = fit` must match that source fit when attached to an archive. Recreating an older fit-derived forecast repeats its simulations and replication fits; the original calibration fit can be reused. A result created with `sim_spec = spec` remains an explicitly supplied scenario and can accompany a different fit. Its notes and the manifest's `population_prediction_source` setting distinguish this from a forecast derived from the attached fit. Do not change the source label of an old result to bypass the check.

Generated replay scripts continue to use the saved simulation specification. They reproduce those forecasting assumptions rather than deriving new ones from the replayed calibration, which could change the forecast. The recreated forecast is therefore labelled as a supplied-specification scenario. Identity checks do not change scoring uncertainty, eligibility or the interpretation of plausible values and Monte Carlo forecast uncertainty.

Updating saved structural reviews in 0.2.4

`analyze_hierarchical_structure()` now saves `data_identity` metadata. `reporting_checklist()` checks the reviewed column roles and retained rows, including scores when ICC was computed. Raw-data reviews and reviews of a subset of fitted facets remain supported when those data match; the checklist names their scope. Row order, factor level order, unused columns and fitted estimates are irrelevant to this descriptive-data match. Different rating counts, assignments or ICC scores are not interchangeable.

Recreate an older review from its saved fit or matching retained rows using the original facets, person/score columns, ICC and connectivity options, missing-value policy and interval settings (including bootstrap seed). This repeats the requested structural/ICC review, not MFRM estimation. Do not add identity metadata to old tables manually. Standalone summary and plot methods retain their existing requirements. Reviews of additional raw rows or grouping columns absent from the fit must be reported separately; they cannot mark this fit's structural-review checklist item as available. Recalculate nesting reviews involving fractional/zero weights or missing facet values: `detect_facet_nesting()` now uses weighted frequencies directly and one common set of observed rows per pair. Earlier results rounded weights and mixed marginal and conditional samples. Weight columns must now be numeric, finite and nonnegative, with some positive weight; missing columns and invalid values are refused instead of ignored or replaced. New `UsedRows`/`ExcludedRows` columns record the pair-specific scope.

Rating-count review corrections in 0.2.4

In facet_small_sample_review(), facet_summary$WorstCategory now selects the lowest count band, correcting the previous highest-band aggregation. Recreate the review from the saved fit to update older review objects. Count thresholds must now be finite, positive and strictly increasing. The table columns and default thresholds remain the same; the returned notes field explains that counts are retained rating rows and cannot establish precision or coverage.

In reporting_checklist(), the item formerly named "Facet sample-size adequacy" is now "Facet rating-count screening". Scripts selecting that item by its text must use the new label. Its availability indicates that a count screen is present, including sparse counts, rather than treating a high count band as proof of sample adequacy.

Retained table-field names

- row_review is the data-quality table field used to document row filtering. FACETS-style column and metric contract results are exposed as column_review and metric_checks.
- Prediction outputs expose row-preparation and person-omission traceability as row_review and population_review.
- hierarchical_review, shrinkage_review, and nesting_review are manifest or model-comparison traceability fields. They are not callable helper names; user-facing helper names use review/check terminology.

Typical workflow

- Legacy handoff: run_mfrm_facets() -> build_fixed_reports() -> facets_output_file_bundle() plus write_mfrm_residual_file() / write_mfrm_subset_file() only when those standalone files are needed.
- Mixed workflow: RSM / PCM: fit_mfrm() -> diagnose_mfrm() -> build_apa_outputs() -> compatibility export only if required. GPCM: fit_mfrm() -> diagnose_mfrm() -> reporting_checklist() -> facets_output_file_bundle() only when a graph/score handoff is needed, retaining its package-native GPCM caveats.
- FACETS output-contract review: fit_mfrm() -> diagnose_mfrm() -> facets_output_contract_review().

Companion guides

- For FACETS coverage and boundary wording, see facets_positioning_guide() and facets_feature_coverage()
- For standard reports/tables, see mfrmr_reports_and_tables.
- For manuscript-draft reporting, see mfrmr_reporting_and_apa.
- For visual diagnostics, see mfrmr_visual_diagnostics.
- For linking and DFF workflows, see mfrmr_linking_and_dff.
- For end-to-end routes, see mfrmr_workflow_methods.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrmr_compatibility_layer", run.dontrun = TRUE)
## Not run:
```

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]

run <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
summary(run)
compatibility_alias_table("functions")

fixed <- build_fixed_reports(
  estimate_bias(
    run$fit,
    run$diagnostics,
    facet_a = "Rater",
    facet_b = "Criterion",
    max_iter = 1
  ),
  branch = "original"
)
names(fixed)


## End(Not run)
```

`mfrmr_example_data` *Synthetic many-facet rating examples*

Description

Compact synthetic many-facet datasets sized to exercise the documented workflow. Their dimensions are not evidence of sample-size adequacy for an applied study.

Format

A data.frame with 6 columns:

Study Example dataset label (`"OperationalExample"`, `"ExampleCore"`, or `"ExampleBias"`).

Person Person/respondent identifier.

Rater Rater identifier.

Criterion Criterion facet label.

Score Observed category score on a four-category scale (1–4).

Group Balanced grouping label (`"A"` / `"B"`). It is neutral in the operational and core examples; the bias example has the planted group structure described below.

Details

Available data objects:

- `mfrmr_example_operational`
- `mfrmr_example_operational_design` (documented separately below)
- `mfrmr_example_core`
- `mfrmr_example_bias`

`mfrmr_example_operational` is the primary applied teaching example. It has a connected but incomplete two-rater assignment, moderately unequal rater workloads, and six planned criterion-level omissions. Scores are sampled directly from stated RSM category probabilities. The six unobserved ratings are absent rows in the long data rather than `NA` scores. Each group contains 24 persons, and three omissions in each group leave 141 observed rows for Group A and 141 for Group B. The balanced `Group` variable has no effect in the generating model; random observed differences may still occur.

`mfrmr_example_core` is an idealized complete crossing generated from a single latent trait plus rater and criterion main effects. It is useful as a fast deterministic example, but it is not representative of routine incomplete operational assignment.

`mfrmr_example_bias` instead uses a balanced partial two-rater assignment. Group A and B latent means are -0.1 and 0.1 logits, respectively, with a common SD of 0.9. It also contains:

- a planted `Group x Criterion` effect (`Group B` is advantaged by 1.2 logits on `Language`)
- a planted `Rater x Criterion` interaction (`R04 x Accuracy` lowers the linear predictor by 1.2 logits)

This lets differential-functioning and bias-analysis help pages demonstrate non-null findings.

Data dimensions

Dataset	Rows	Persons	Raters	Criteria	Groups
example_operational	282	48	6	3	2
example_core	768	48	4	4	2
example_bias	384	48	4	4	2

Suggested usage

- Use `mfrmr_example_operational` for the beginner data-to-report workflow and for inspecting incomplete but connected assignment.
- Use `mfrmr_example_core` for fast, idealized checks and examples that specifically require complete crossing.
- Use `mfrmr_example_bias` for `analyze_dff()`, `analyze_dif()`, `dif_interaction_table()`, `plot(x, type = "heatmap")`, and `estimate_bias()`.

All three objects can be loaded either with `load_mfrmr_data()` or directly with `data()`, for example `data("mfrmr_example_operational", package = "mfrmr")`.

Source

Synthetic documentation data generated from rating-scale Rasch facet designs with fixed seeds. Generator scripts are maintained under `data-raw/` for this release in the public source repository. These are synthetic examples, not empirical records.

Examples

```
data("mfrmr_example_operational", package = "mfrmr")
table(mfrmr_example_operational$Score)
table(mfrmr_example_operational$Group)
```

---

mfrmr_example_operational_design

Planned assignment roster for the operational example

Description

A score-free roster declaring all Person x Rater x Criterion cells planned for `mfrmr_example_operational`. Pass it to the `expected_design` argument of `describe_mfrm_data()` to distinguish the six expected-but-unobserved cells from combinations that were never assigned.

Format

A data.frame with 288 rows and 5 columns:

Study Example dataset label (`"OperationalExample"`).

Person Person/respondent identifier.

Rater Planned rater identifier.

Criterion Planned criterion label.

Group Balanced grouping label (`"A"` / `"B"`).

Details

The roster contains 288 planned cells. The observed `mfrmr_example_operational` table contains 282 rows, so an explicit design comparison identifies six planned omissions. Extra roster columns such as `Study` and `Group` are ignored unless they are named as model facets.

Source

Synthetic assignment roster generated with the operational example by `data-raw/make-operational-example.R`. It contains no empirical records and no fabricated scores.

Examples

```
data("mfrmr_example_operational", package = "mfrmr")
data("mfrmr_example_operational_design", package = "mfrmr")
review <- describe_mfrm_data(
  mfrmr_example_operational,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  expected_design = mfrmr_example_operational_design
)
summary(review)$structural_missingness
```

`mfrmr_gtheory_data` *Synthetic crossed and nested data for learning G theory*

Description

Existing tutorial examples supplied as ready-to-load tables. Load one with `data("mfrmr_example_gtheory_crossed", package = "mfrmr")`, then assign it to a shorter name such as `ratings`. No simulation code is needed to run the G-study or D-study examples.

Format

Two data frames with five columns: `Person`, `Rater`, `Task`, `Content`, and `Organization`. Each row is one Person-by-Rater-by-Task rating cell; the last two columns are continuous numeric scores. `mfrmr_example_gtheory_crossed` has 1,920 rows (40 Persons, eight common raters and six common tasks). `mfrmr_example_gtheory_nested` has 540 rows (30 Persons, six tasks and three raters per task).

Details

In the crossed example, the same rater ID denotes the same rater on every task. In the nested example, `R1`, `R2`, `R3` are local labels: the pair `Task` and `Rater` identifies a rater, giving 18 raters in total. Declare `nesting = c(Rater = "Task")` for that example. Persons are crossed with all tasks and rater teams in both examples.

The scores were constructed from independent normal random effects for every source in the respective design. The nested example correlates the two score components within each source. These are fictional unbounded continuous scores, not ordered rubric categories or MFRM ability estimates. Their design and sample sizes demonstrate the functions; they do not establish that an assessment's assumptions or estimation precision are adequate.

Source

The existing crossed G-theory tutorial (seed 24) and nested G-study help example (seed 2027). Their unchanged generation recipes are retained in `data-raw/make-teaching-data.R` in the source repository.

See Also

mfrm_g_study(), mfrm_d_study(), mfrm_multivariate_d_compare(), load_mfrmr_data()

Examples

```
data("mfrmr_example_gtheory_crossed", package = "mfrmr")
ratings <- mfrmr_example_gtheory_crossed
head(ratings)
g <- mfrm_g_study(ratings, scores = c("Content", "Organization"),
  person = "Person", rater = "Rater", task = "Task")
g$component_diagnostics
# Inspect component estimates before projecting future plans.
d <- mfrm_d_study(g,
  design_grid = data.frame(Raters = c(2, 4), Tasks = 3),
  weights = c(Content = .5, Organization = .5))
summary(d)
```

---

mfrmr_interval_guide *Confidence-interval and uncertainty route guide*

Description

Return a compact map of the public mfrmr routes that can expose confidence intervals or interval-like uncertainty displays. Use this when you need to know which helper accepts show_ci or ci_level, which columns to look for in draw = FALSE output, and how strongly the resulting interval should be interpreted.

Usage

```
mfrmr_interval_guide(
  scope = c("all", "visual", "table", "reporting", "fit", "bias", "linking", "gpcm",
    "equivalence", "hierarchical", "shrinkage")
)
```

Arguments

scope	Which rows to return: "all" (default), "visual", "table", "reporting", "fit", "bias", "linking", "gpcm", "equivalence", "hierarchical", or "shrinkage".

Details

The guide is deliberately conservative. It is a namespace and interpretation map, not a fitted result and not proof that a given interval is available for a particular run. For run-specific availability, call the listed helper with draw = FALSE or inspect the relevant result table.

Most rows use ci_level = 0.95 by default. Some intervals are model-based Wald intervals, some are delta-method intervals, some are profile or profile-like intervals when available, and some are plotting overlays around already-estimated quantities. The Basis and InterpretationBoundary columns are the important guardrails.

Value

A data.frame with columns:

- `Route`
- `Scope`
- `PrimaryHelper`
- `DisplayRoute`
- `DefaultLevel` (missing for an uncertainty-only route without intervals)
- `IntervalColumns`
- `Basis`
- `UseFor`
- `InterpretationBoundary`
- `GPCMStatus`
- `Notes`

Why an estimate or interval is missing

A finite printed value may be a usable estimate, an optimizer value retained for review, a constraint-fixed value, or a prior-only score. Read the result's status and explanation before interpreting it. An `NA` bound is not an infinite limit, zero uncertainty, or evidence of no effect.

- **Fit and parameter readiness:** start with `summary(fit)$decision`, `$readiness` and `$key_warnings` for ordinary fits. For CML inspect `summary(fit)$checks` and `$notes`; comparison-only CML retains `EstimateStatus` and `EstimateReason` in `fit$comparisons`. Identification failure or absence of a finite MLE cannot be repaired merely by increasing optimizer iterations. A refused calculation explains the restriction in its error instead of returning an inferential result.
- **Facet and population intervals:** inspect `ci$table$Status`, `$CIEligible` and `$InferenceReview`, together with the result's checks and cautions. Missing SEs or bounds may coexist with a retained point; that does not certify the point as reliable. `fixed` refers to a target fixed by constraints, not an estimate with perfect measured precision.
- **GPCM slope/profile intervals:** use `attr(ci, "diagnostics")` and the saved endpoint checks. They distinguish an unavailable interval from a partially resolved profile; `confint.mfrm_fit()` describes each method.
- **Testlet and shared-rater normal intervals:** use `attr(ci, "diagnostics")` and `attr(ci, "checks")`; default summary tables also retain `IntervalStatus` and `InferenceReview`. These distinguish intervals not requested, unsupported targets, failed numerical/information checks, and estimated zero-variance boundaries. The boundary restriction applies even when optimization has converged. See `confint.mfrm_testlet()` and `confint.mfrm_random_rater()`.
- **Person scores:** `score_mfrm_persons()` and the extension scoring functions retain row-level `Status` and `Reason`. `prior_only` means no observed responses: the reported mean and interval describe the assumed population. `unavailable` withholds individual estimates and intervals. Review the stated cause; more quadrature nodes cannot supply observations or remedy a zero ability-variance restriction.

- **G/D-study intervals:** read the metric-specific `Status`, diagnostics and notes. A width below numerical resolution is a calculation limitation, not perfect reliability; raw log-ratio or pivot diagnostics may still be retained. An explicitly reported zero endpoint, a constraint boundary and an unresolved numerical interval have different meanings.

Increase integration precision or revise optimizer controls only when the saved diagnostics identify those numerical issues. Model assumptions, insufficient design information and unsupported boundary inference need substantive review. Numerical success alone does not resolve them. Keep explanations with saved results: use `saveRDS()` for the whole object, and export the status/reason table with CSV. Older saved tables do not gain fields merely by being read; recreate summaries or interval displays from their source fit where supported. Do not rewrite historical fit checks.

Availability is not coverage

Read the target, interval method and assumptions before interpreting bounds. Numerical availability, sampling coverage and public API support are separate questions. Where returned, `CIEligible` records the source and method's eligibility checks, which can include model, design and numerical restrictions; `CoverageQualified = FALSE` means nominal sampling coverage is not established for that procedure. It is not a measured coverage rate and does not erase computed bounds. Absence of that field is not evidence of coverage either.

A partial profile retains computed endpoints and the unresolved side's reason. An unresolved search is not an infinite limit or a calibrated one-sided interval. Inspect `apa_table(result, which = "profile_endpoints")` and retain the method's notes when reporting or exporting it.

Corrected-JML `RootSE` describes variation around an adjusted-equation root; residual bias can separate that root from the true structural parameter. It must not be turned into a structural confidence interval by multiplying it by 1.96. This guide includes that uncertainty-only route with no default confidence level. Unavailable public structural intervals do not establish that every interval method is mathematically impossible.

Conditional Person posterior intervals hold calibration fixed. A full external Bayesian analysis, for example through `export_mfrm_model()`, has different assumptions and returns credible intervals; export alone neither runs the sampler nor validates its intervals. Inspect sampler precision, convergence and prior sensitivity. Neither type of posterior interval is automatically a frequentist confidence interval with nominal coverage. See `gpcm_capability_matrix()` for estimator-specific routes and restrictions. For coefficients, conditional population means and residual variance in an estimated RSM/PCM population, use `mfrm_population_intervals()`. Its plot() method displays coefficients or saved contrasts, retaining unavailable intervals and experimental status. Residual variance stays in the table because its units differ. This route is separate from Person scoring and facet-location intervals.

Native RSM/PCM CML intervals use conditional information, with no estimated Person abilities or population SD. The guide gives a dedicated route for saved comparison-only targets and explains the facet argument for full CML results. A separate full-MML interaction route accepts `comparison_review` and optional `scale = "standardized"`; estimated-SD standardization includes the same fit's population-SD uncertainty and its covariance with the contrast. It requires an intercept-only normal population rather than substituting a regression residual SD. Neither numerical eligibility nor a positive conditional identification check establishes global MML identification or finite-sample interval coverage.

See Also

mfrmr_visual_diagnostics, visual_reporting_template(), plot(x, type = "fair_average"), plot(x, type = "bias"), plot(x, type = "displacement"), plot(x, type = "wright", view = "unified"), plot(x, type = "rater_severity"), plot(x, type = "apa_summary"), fit_measures_table(), mfrm_facet_intervals(), mfrm_population_intervals(), pool_mfrm_imputed(), mfrm_random_rater_intervals(), confint.mfrm_random_ra fit_mfrm_testlet(), predict.mfrm_testlet()

Examples

```
mfrmr_interval_guide()
mfrmr_interval_guide("visual")[, c("Route", "DisplayRoute", "Basis")]
mfrmr_interval_guide("gpcm")[, c("Route", "GPCMStatus", "InterpretationBoundary")]
```

---

mfrmr_linking_and_dff *mfrmr Linking and DFF Guide*

Description

Package-native guide to checking connectedness, building anchor-based links, monitoring drift, and screening differential facet functioning (DFF) in mfrmr.

What does an anchor do?

Suppose a previous calibration placed Rater R01 at 0.35 logits. A direct anchor tells the new fit to keep that value exactly; the new ratings do not re-estimate it. This requires the same rater identity, model orientation and a defensible stability assumption, not just the same text label.

Keep four ideas separate:

- **Common ratings:** overlapping Persons, tasks or raters connect observed ratings. Their parameter values need not be known or fixed.
- **Scale identification:** a constraint such as a zero mean sets an origin for otherwise interchangeable coordinates. It does not establish invariance.
- **Direct anchors:** Facet, Level, Anchor fixes a location. For example, R01 = 0.35 means exactly 0.35 on the internal logit scale.
- **Group anchors:** Facet, Level, Group, GroupValue fixes a mean. For R01 and R02 in a group with target 0.20, their locations must sum to 0.40; neither location is individually fixed unless also directly anchored. This is an unweighted mean across declared levels, including fixed members.

What the current implementation supports

Non-Person location constraints are available in supported ordinary MML/JML fits. Person direct/group constraints require JML; MML integrates ability and rejects them. This is an implementation boundary, not a claim that fixed-Person MML is mathematically impossible. Two-family GPCM has separate restrictions; see gpcm_capability_matrix().

With `centering_scope = "facet"` (the ordinary-fit default), centered facets sum to zero across all levels, including direct anchors and group totals. A single ungrouped remainder therefore balances those totals. Use `centering_scope = "remainder"` to reproduce earlier fits that centered only the levels outside direct/group constraints. Ordinary additive RSM/PCM JML also offers `centering_scope = "anchor_origin"`: direct or group anchors replace their facet's centering, with an unanchored, active noncenter facet. Group targets are repeated arithmetic means, not per-element contributions. The default remains `"facet"`. Read `fit_mfrm()` before linking a cohort with new raters or tasks; population and `noncenter_facet` settings also determine the scale. Contradictory constraints stop before estimation.

`anchor_to_baseline()` transfers selected facet locations, not a whole calibration: thresholds/slopes are re-estimated, and most settings must be supplied again. RSM/PCM MML defaults to a fixed standard-normal population; anchoring does not automatically estimate a changed ability distribution. Use `extract_mfrm_calibration()` and `score_mfrm_calibration()` to score against a complete fixed calibration within that separate workflow's supported scope.

Annual assessments with common recordings

Suppose an English speech contest has new contestants each year, mostly the same judges, and a common three-minute format. Judges also rate the same recorded speeches each year. These recordings can connect annual ratings if their performance meaning and scoring conditions remain comparable. Returning judges or unchanged criterion names alone do not establish that their severity or interpretation stayed constant.

Give the same recording the same performance ID across years. Give a new speech a new performance ID, even if its speaker competed before. Keep judge identity and rating year/session separately: the question is how a judge's severity changes, not whether that judge is the same person. Contestant-group differences are distinct from individual growth.

Two analysis strategies have different assumptions:

- **Joint calibration:** share an unknown performance measure for each common recording across years and estimate judge-year effects. The recording values need not be known or passed to `anchors`. Shared unknown measures still require an identified model and common scale.
- **Fixed-reference calibration:** fix previously calibrated recording measures when fitting a later year, while estimating its judges' effects. When recordings occupy the Person role, those direct anchors require JML in the current implementation. The receiving fit treats their values as exact; their source estimation error is excluded.

Do not fix the returning judges' severities merely because they return. Also, centering the panel to the same mean separately each year can remove the panel-wide change being studied. Annual rater effects, criterion/step invariance and population assumptions must be specified together. Separate fixed-standard-normal MML fits do not estimate changing cohort means.

Current helper output answers narrower questions. In an anchored result, `$drift` compares retained fixed non-Person values with their supplied values; zero differences do not show stability. In `detect_anchor_drift()`, `$drift_table$Drift` subtracts `LinkOffset`. If judges are the linking set, a common shift in all their estimates is removed; the remainder cannot establish that the panel was stable. Nor does `LinkOffset` alone identify a real panel change, since it can also reflect different coordinates.

Preserve individual ratings before discussion or disclosure of reference scores, training phase, and video/live mode. Check coverage across performance levels and criteria, memory of familiar scores,

shared assignments and dependence among repeated ratings. Training agreement alone does not demonstrate stable operational scoring. A complete common-recording workflow and formal year-difference intervals are not yet verified in mfrmr; conditional anchor SEs do not supply cross-year covariance or source-anchor uncertainty. The companion linking vignette develops this design example and explains what the existing returned tables can establish.

Two schools connected by common recordings

Suppose School A has 30 students and five raters, School B has 40 students and three raters, and both use three tasks. If each rater scores only their own school's students, common tasks alone do not separate a school-specific shift in student ability from an equal shift in its panel's severity. Having all eight raters independently score the same three recordings can supply that missing link. In a joint calibration, the common performances share IDs and unknown measures; their scores need not be known anchors. Unique local student IDs and distinct rater IDs prevent accidental overlap. Record the task actually performed in each recording; do not replicate a single recording across tasks that were never observed. Define whether the measured unit is a student across tasks or a particular performance.

This supports a model-based common scale under the linking assumptions, not automatic interchangeability of school raw scores or a causal school effect. Three recordings are three shared performances, not 24 independent linking performances merely because eight raters assess each. Their ability range, category coverage and stability across panels matter. Review whether judgments were independent, reference scores were disclosed, or recording versus live presentation changes severity. A connected graph is necessary evidence to inspect, not proof of identification, precision or invariance.

Specify each rater-by-school comparison separately. A rater's A-minus-B effect relative to the entire panel is different from the difference between two raters' A-minus-B effects. With incomplete school coverage, the second comparison can be identified even when the first is not. State the reference raters and their weights; dropping an unobserved rater changes a whole-panel comparison. Under PCM, also state the step owner: a location contrast alone need not describe every adjacent-category response contrast when the compared raters own different step ladders.

Preserve a recording's school of origin when another school rates it. Shared neutral recordings can connect the scale without supplying the missing A-versus-B observations for each rater. A common latent-population assumption can supply information absent from an unrestricted-Person comparison; allowing school-specific population means can restore that ambiguity. These are different assumptions, not evidence that MML always resolves an unidentified JML comparison. Use `describe_mfrm_data()` with `comparisons` to review explicitly weighted RSM/PCM interaction contrasts before fitting. Its conditional-map results and MML limitations remain separate from marginal identification and from residual/refit DFF effects. Its `IdentificationBasis` column distinguishes a target identified in the unrestricted-Person response model from one whose conditional identification requires the declared population restrictions. Neither makes MML distribution-free or proves global marginal identification. CML instead records identification from conditional support at the observed totals. `analyze_dff()` can retain that review as `comparison_review` after checking the retained responses and origin labels; it does not automatically certify the identification of its own residual/refit contrasts. For a matching joint interaction fit, pass the review to `interaction_effect_table()` or `summary(fit, comparison_review = review)`. These return descriptive weighted interaction estimates after full conditional-rank, model/data alignment and retained-point numerical checks. Unavailable rows retain reasons and separate optimizer traces. Positive MML conditional ranks still do not certify global marginal identification; no partial fit is run

by extraction. For target-only identification, explicitly use `fit_mfrm(fit_scope = "comparisons", comparison_review = review)` with the review's model arguments. Its separate comparison result preserves the observed likelihood and invariant targets, checks JML recession and numerical adequacy, and retains unidentified/failed targets as missing. MML conclusions remain conditional/local; JML/MML comparison fits supply no formal uncertainty. Declaring `method = "CML"` in the review enables conditional-information SEs and pointwise 95 percent Wald intervals for identified RSM/PCM contrasts. `mfrm_facet_intervals(fit)` reuses these saved comparisons for interval plots and APA tables, retaining unidentified rows without estimates. Its `level` argument changes the normal critical value without refitting. Identification uses conditional support at the actual Person totals, and finite-MLE and information checks determine availability. Unidentified targets remain missing. This route needs no ability distribution and estimates no population SD; its asymptotic intervals do not establish finite-sample coverage or a calibrated DFF classification. See `fit_mfrm()` and the linking vignette.

For matching full RSM/PCM MML fits, `mfrm_facet_intervals()` with `comparison_review` adds model-based interaction intervals after joint information and finer-grid checks. `scale = "standardized"` divides by the same fit's estimated intercept-only normal population SD, with the full joint delta method. That common-population assumption differs from `~Group`; the latter supports native intervals but its residual SD is not treated as marginal population SD. Numerical eligibility does not certify global marginal identification or finite-sample coverage.

GPCM comparisons also require the slope owner and ability scale. A positive location difference does not ensure that the expected-score difference has the same sign across ability: unequal slopes can produce crossing response curves. Specify the ability range and compare category probabilities or expected scores with the actual step ladders. One-family slopes have geometric mean one; under default MML the population standard deviation supplies the common discrimination scale. Fixing both that standard deviation and the slope geometric mean to one imposes a narrower model.

With two slope families, distinguish each observed effective slope product from its individual facet components. Disconnected facet crossings can preserve observed products while reversing a component comparison; the current fit route rejects that design. Predicting an unobserved crossing depends on the multiplicative slope model. A full-rank product map alone does not certify the full marginal response model, and a local numerical rank check is not a global identification or precision guarantee. The supported two-family route uses fixed-standard-normal MML with exactly two measurement facets; it does not supply a rater-by-school interaction model.

Prefer the joint route when no externally established reference values are available. Fixing previously estimated recording values in the Person role instead uses the JML-only anchor route and excludes their source uncertainty. Do not force each school's rater panel to have the same mean to manufacture a link, or add School as a free additive response facet alongside unrestricted student effects without checking identification. A population school-mean contrast needs its own population model and handling of common recordings outside those cohorts; it is not an ordinary comparison of EAP means. The existing common-recording workflow has not yet been verified end to end for this school design or its school-difference uncertainty.

Can I obtain intervals when I use anchors?

For ordinary unit-weight RSM/PCM MML with fixed standard-normal ability and fixed-grid integration, `mfrm_facet_intervals()` can return model-information or cluster-sandwich intervals for free facet locations and contrasts, subject to its checks. These intervals condition on the supplied anchors. A fixed location has SE zero and no inferential bounds; its source uncertainty has not been estimated by the receiving fit.

Estimating the population mean and variance with anchors is a different configuration. Its point estimates are supported, but the native population and location interval helpers currently exclude anchors and constraint groups. An `EstimabilityState` of `"not_evaluated"` denotes an incomplete identification check, not failed optimization or proof of nonidentifiability. Inspect `fit$readiness$fit` for numerical status and `fit$data_review$estimability$nonlinear_local_estimability` for the separate local-rank result. Passing that local check alone does not authorize intervals or anchor export for reuse. Do not choose a fixed population solely to obtain intervals when its assumptions are inappropriate.

One-family GPCM MML has a separate slope/curve route: `confint.mfrm_fit()` with `parm = "slopes"` and `mfrm_curve_intervals()` can condition on fixed anchors when their own checks pass. Anchored native-location/profile-slope intervals and portable GPCM extraction remain unavailable. An available interval does not certify the separate integration checks required by `predict_mfrm_units()`.

Reusing an analysis saved before the centering change

- **Read the original result:** `readRDS()` preserves its coefficients and constraints. `summary(fit, compute = "never")$settings_overview` and `specifications_report()` identify the centering convention. A native fit without convention metadata uses the legacy `"remainder"` interpretation.
- **Repeat the original analysis:** `build_mfrm_replay_script()` embeds its anchors and retains that convention. Running the script refits the model; it does not merely read a saved result.
- **Adopt whole-facet centering:** refit from the original responses with `centering_scope = "facet"` and the intended anchor, population, rubric, uncentered-facet and owner settings. Do not edit `fit$config` or recenter an output table to convert the fit. Keep the old and new analyses separately.
- **Refresh derived output:** recompute interval objects from the unchanged fit for revised fixed-contrast derivatives or anchor explanations. Rerun affected `analyze_dff()` subgroup analyses to retain target/Person anchors; this does require subgroup refitting. Loading old interval/DFF objects does not update them.
- **Keep a frozen calibration:** `load_mfrm_calibration()` preserves its original labels and values. Re-extract from the source fit, satisfying the current validation requirements, for explicit centering labels. If only the fit survives without the required RSM/PCM MML quadrature review, obtaining that review needs the original data and additional grid fits. If only the calibration file survives, do not infer a missing convention or relabel it; keep its original scoring interpretation. Changing its calibration requires the source analysis and a new validated, frozen artifact.

The linking vignette provides a runnable migration example. Source-anchor uncertainty remains excluded; a new label or file format adds no uncertainty calculation. Older MML analyses containing Person anchors need substantive review: those values were ignored, whereas current fitting rejects them. Merely selecting the legacy centering mode does not repair such an analysis.

Comparing anchors with TAM and ConQuest

The same anchor number does not necessarily fix the same comparison. Check the response model, parameter coding and remaining constraints before transferring values between programs. A parameter index can change when the model or retained facet levels change; match the actual design and labels, not just the row number in an earlier output.

TAM's `xsi.fixed` fixes indexed coefficients of its response design, which need not be standalone facet locations. ConQuest's `anchor_parameters` (`anchor_xsi`) serves that role. Population regression and covariance have separate controls: TAM's `beta.fixed`/`variance.fixed` and ConQuest's `anchor_beta`/`anchor_sigma`. Slopes also require their own coding; neither an mfrmr location table nor an arbitrary list of external coefficients represents a complete calibration.

For a concrete RSM example, fix one of four raters at 0.70 and leave the others outside group constraints. Whole-facet centering makes the other three raters sum to -0.70: R01 is about 0.933 above their mean. This matches the rater constraint in TAM's `tam.mml.mfr(formulaA = ~ item + rater + step, constraint = "cases")` and ConQuest's `model task+rater+step` with `lconstraints=cases`. Those settings also leave task locations uncentered; use the corresponding `noncenter_facet` and match the population explicitly. mfrmr's legacy `"remainder"` mode instead makes the other raters sum to zero, putting R01 0.70 above their mean. Changing labels cannot reconcile these restrictions. Group means likewise require matching linear constraints, not copying them as individual fixed values.

Match the population mean/variance, scoring, estimator and integration accuracy as well. Equal numbers of integration nodes need not mean equal rules or accuracy. ConQuest documents `anchor_theta` as ignored under MML; mfrmr instead rejects Person anchors there. TAM's `tam.jml()` documents `theta.fixed` for `version = 1`. Common Persons and fixed Person abilities remain distinct concepts in all these comparisons.

Agreement of point estimates does not establish agreement of SEs. Compare uncertainty for the same free parameter or contrast, using the appropriate covariance transformation. In ordinary fixed-anchor estimation, uncertainty is conditional on the supplied values; a zero or missing SE for an anchor does not account for its source calibration error. See the companion vignette for a comparison table and the official manuals below.

Start with the linking question

- "Is the design connected enough to support a common scale?" Use `subset_connectivity_report()` and `plot(..., type = "design_matrix")`.
- "Which elements can I export as anchors from an existing fit?" Use `make_anchor_table()` and `review_mfrm_anchors()`.
- "How do I anchor a new administration to a baseline?" Use `anchor_to_baseline()`.
- "Have common elements drifted across separately fitted waves?" Use `detect_anchor_drift()` and `plot(x)`.
- "Can I synthesize anchor review, drift, and chain evidence into one review?" Use `build_linking_review()`.
- "Do specific facet levels function differently across groups?" Use `analyze_dff()`, `plot(x, type = "heatmap")`, and `plot(x, type = "summary")`.

Recommended linking route

Before fitting, use `describe_mfrm_data()` on the rating table and inspect `design_connectivity`. Supply `expected_design` if the planned assignment is available, so unassigned cells and missing assigned ratings are distinct. A connected graph alone does not establish identification of all model parameters, adequate precision, or absence of informative assignment.

1. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`.
2. Check connectedness with `subset_connectivity_report()`.

3. Build or review anchors with `make_anchor_table()` and `review_mfrm_anchors()`.
4. Use `anchor_to_baseline()` when you need to place raw new data onto a baseline scale.
5. Use `build_equating_chain()` only as a screened linking aid across already fitted waves.
6. Use `detect_anchor_drift()` for stability monitoring on separately fitted waves.
7. Use `build_linking_review()` when you need one operational synthesis object rather than separate anchor/drift/chain tables.
8. Run `analyze_dff()` only after checking connectivity and common-scale evidence.

Keep anchor roles separate

- A common-element or common-rating link is observed overlap in the design.
- `anchors` are direct equality constraints on selected parameter values.
- `group_anchors` constrain declared group means and require an externally defensible target or equal-mean assumption.
- `make_anchor_table()` exports candidate direct constraints mechanically; by default it requires a current inference-ready source fit, but it does not select invariant elements. `readiness_policy = "review"` is for review-only extraction, not anchor reuse.

Constraint-based coordinate transfer does not create empirical overlap. Before reuse, also verify consistent model/score/orientation/population conventions, cross-run element identity, and the relevant assignment connectedness. `review_mfrm_anchors()` checks syntax and receiving-data support, not those substantive assumptions.

Common persons, rotating assignments, and random assignments

These are ways to collect ratings, not instructions to fix a parameter. For example, asking all eight raters to score the same four performances creates common-person links. Their abilities can still be estimated; do not pass them to `anchors` unless known values really are intended.

Existing `build_mfrm_sim_spec()` choices let you inspect these structures:

- Common persons: `assignment = "sparse_linked"` with `sparse_controls = list(link_persons = 4, link_raters_per_person = 8)`. Set `n_rater = 8` and specify the ordinary `raters_per_person` separately. The counts illustrate syntax, not a recommended number of linking persons.
- Overlapping rotation: `assignment = "rotating"` with `raters_per_person = 2` assigns successive subsets of raters.
- Random subsets: `assignment = "sparse_linked"` with `sparse_controls = list(link_persons = 0, assignment_mode = "random")`. This does not guarantee connectedness or equal workloads for every draw.
- Task-specific or externally planned assignments: `assignment = "skeleton"` accepts an explicit Person-by-facet template. The ordinary rotating generator assigns the chosen raters across all generated criteria; it is not a general implementation of task-nested spiral designs.

Generate ratings with `simulate_mfrm_data()` and inspect them with `describe_mfrm_data()`. Use the linking vignette for runnable examples. A generator can represent a design without qualifying an estimator for it. Compare both the amount and placement of overlap, and what happens when a critical common person or rater is unavailable. Keep rating costs equal when the question is which allocation uses a fixed budget better.

Design the link, not only the anchor count

There is no universal adequate anchor count or percentage. Linking quality also depends on where overlap occurs, whether links are distributed or rely on one critical bridge, the rating workload and coverage, and model–data fit within the linking set. Inspect both Rater-centered and Person/task-centered assignment graphs when the design is sparse. The package's five-element `LinkSupportAdequate` screen is therefore a local warning, not a design recommendation or proof of a common scale.

Fixed anchor values also carry uncertainty from their source calibration. Current anchor, drift, and chain summaries do not propagate all source-fit, offset, or cross-fit covariance. Compare substantively defensible anchor sets and report sensitivity when conclusions depend on the selected set. The bounded fixed-SD RSM/PCM anchor-origin route now offers explicit independent-source covariance propagation through `mfrm_facet_intervals()` and `mfrm_population_intervals()`. This does not change drift or chain summaries, correct noninvariant anchors, or handle dependent samples.

Reading the theoretical background

Uto (2021, pp. 1442–1444) explains fixed common-rater/task linking. Its simulation fixes generated common parameters and removes a redundant rater identification constraint; its Bayesian estimation and constraints differ from mfrmr. It does not validate every centering convention above or provide uncertainty propagation for anchors estimated in a previous mfrmr fit.

Robitzsch (2024, pp. 1181–1182, 1187–1188) studies bias and linking error from random uniform DIF in fixed-item 2PL calibration. It separates this issue from sampling Persons and does not provide a variance estimator for the fitted population parameters. It motivates checking anchor invariance, not treating current mfrmr SEs as including every source of linking error. For the distinction between an origin constraint and a fixed external value, see the FACETS centering and element documentation cited below; its constraint convention is not identical to mfrmr's.

Drift and chain inference boundary

`detect_anchor_drift()` and `build_equating_chain()` currently estimate one pooled offset across all selected common elements and facets. Their element-level SE ratios omit estimated-offset uncertainty and cross-fit covariance, and chain output does not propagate uncertainty across adjacent links. `Offset_SD` is residual spread, not an offset SE. Treat flags, support counts, and cumulative offsets as review screens; when a common shift across facet blocks is not defensible, analyze one coherent linking facet at a time. `ScreeningFallback` indicates that preliminary screening retained nothing and the offset instead used all finite differences. `N_Contributing` counts the rows actually entering the final average, which can differ from `N_Retained`. Neither count certifies invariant anchors. Missing offsets/spread are explained by `OffsetStatus`, `OffsetReason` and, for chains, `OffsetSDStatus`. A single retained element has undefined residual sample SD, not zero uncertainty. Drift `FlagBasis` distinguishes the magnitude-only screen from a screen with an available SE ratio.

Which helper answers which task

`subset_connectivity_report()` Summarizes connected subsets, bottleneck facets, and design-matrix coverage.

`make_anchor_table()` Extracts candidate direct-anchor values from a fit without certifying them for reuse.

`anchor_to_baseline()` Anchors new raw data to a baseline fit and returns anchored diagnostics plus a consistency check against the baseline scale.

`detect_anchor_drift()` Compares fitted waves directly to flag unstable anchor elements.

`build_equating_chain()` Accumulates screened pairwise links across a series of administrations or forms.

`build_linking_review()` Synthesizes anchor review, drift, and screened-chain evidence into one operational review surface.

`analyze_dff()` Describes group residual differences or linked subgroup facet estimates. Linked refit point contrasts remain screening-only because their uncertainty is conditional on baseline anchors.

`mantel_haenszel_table()` Compares binary responses between two groups within declared matching strata, with one response per Person and target. Common odds-ratio inference does not certify matching validity or support pooled repeated ratings.

`ordinal_logistic_table()` Compares ordered responses using nested cumulative logistic models for independent Persons. Separates uniform, nonuniform and joint group tests from a proportional-odds diagnostic; retains numerical failures and common-range effect contrasts.

Practical linking rules

- Check connectedness before interpreting subgroup or wave differences.
- Use `mfrm_network_analysis()` for assignment/co-observation connectedness. Agreement, severity-direction, and halo networks describe score relations; they do not establish an empirical link or a common measurement scale.
- Residual differences do not isolate differential functioning and have no tests or binary classifications. Even with an adequate link, refit SEs condition on baseline anchors and omit their uncertainty and cross-refit covariance; formal inference remains unavailable.
- Always name the facet, facet level, and group pair involved in a DFF contrast. A generic "DIF exists" statement is not interpretable in a many-facet design.
- Linked refit contrasts remain descriptive; neither method assigns ETS A/B/C labels or supports a formal differential-functioning decision.
- Read a DFF figure with `plot_data(p, "pairs")`: summary plots retain all source comparisons with a `PlotStatus`, including those omitted by `top_n` or lacking a contrast. Both views retain method settings and, for residual results, the response-matching `cases` table. Positive residual and refit contrasts have different meanings; see plot.mfrm_dif.
- For sparse binary cells, exact MH conditions on the stratum margins; intervals may be wide and p-values discrete. This does not correct invalid matching or dependence. Ordinal-logistic LR tests remain asymptotic: successful numerical convergence does not establish small-sample validity.
- MH or ordinal-logistic comparisons require explicit matching, and residual DFF's `$cases` records the actual requested-response matches. Inspect exclusions before comparing methods. Residual `N` counts rows; `NPersons` counts distinct Persons, without establishing independence. All methods require comparable rating plans and an appropriate unit of independence. A raw total across different plans or multiple rows from the same Person does not establish these conditions. See `analyze_dff()` and the companion vignette for differences in targets and response-model assumptions. Its "Compare an explicitly shared response roster" example

aligns group direction, retains method-specific case counts and exclusions, and saves complete residual/MH/logistic results together. It uses a common binary item roster; it does not support pooling repeated Person-target ratings.

- Treat drift flags as prompts for review, not automatic evidence that an anchor must be removed.
- In drift/link-review results, `LinkSupportAdequate = FALSE` warns that at least one linking facet retained fewer than 5 common elements after screening. Refit DFF uses a different screen: by default at least five anchored levels in total across the non-target linking facets in each subgroup. Inspect `ScaleLinkStatus`, `LinkingThreshold`, `LinkingAnchoredLevels1`/`LinkingAnchoredLevels` and `LinkingDetail1`/`LinkingDetail2`. Neither screen proves invariance or supplies a universally adequate anchor count.
- Rebuild anchors from a defensible baseline rather than chaining unstable links by hand.

Typical workflow

- Cross-sectional linkage review: `fit_mfrm()` -> `diagnose_mfrm()` -> `subset_connectivity_report()` -> `plot(..., type = "design_matrix")`.
- Baseline placement review: `make_anchor_table()` -> `anchor_to_baseline()` -> `diagnose_mfrm()`.
- Multi-wave drift review: fit each wave separately -> `detect_anchor_drift()` -> `build_linking_review()` -> plot(x).
- Group comparison route: `subset_connectivity_report()` -> `analyze_dff()` -> `dif_report()` -> plot(x, type = "heatmap") / plot(x, type = "summary").

Companion guides

- For visual follow-up, see mfrmr_visual_diagnostics.
- For report/table selection, see mfrmr_reports_and_tables.
- For end-to-end routes, see mfrmr_workflow_methods.
- For a longer walkthrough, see `vignette("mfrmr-linking-and-dff", package = "mfrmr")`.

References

TAM documentation: MML estimation and fixed parameters and JML and fixed Person parameters.

ACER ConQuest manual: Command reference, especially `import` (anchor parameter types) and `set` (location and scale constraints). Defaults and supported options depend on the program version.

FACETS documentation: Centering facets and Elements and anchors.

Linacre, J. M. (2012). Facets Tutorial 4: Anchoring, p. 9, discusses rater drift and the assumption involved in fixing a group mean across sessions. It does not establish the common-recording workflow described here or its uncertainty performance.

Eckes, T. (2023). *Introduction to Many-Facet Rasch Measurement*, Section 9.1, pp. 151–155, Table 9.1. Peter Lang. ISBN 978-3-631-90305-6.

Myford, C. M., & Wolfe, E. W. (2000). Strengthening the ties that bind: Improving the linking network in sparsely connected rating designs. *ETS Research Report Series*, 2000(1). doi:10.1002/j.23338504.2000.tb01832.x

Uto, M. (2021). Accuracy of performance-test linking based on a many-facet Rasch model. *Behavior Research Methods*, 53(4), 1440–1454. doi:10.3758/s1342802001498x

Wind, S. A., & Jones, E. (2018). The stabilizing influences of linking set size and model–data fit in sparse rater-mediated assessment networks. *Educational and Psychological Measurement*, 78(4), 679–707. doi:10.1177/0013164417703733

Robitzsch, A. (2024). Bias and linking error in fixed item parameter calibration. *AppliedMath*, 4(3), 1181–1191. doi:10.3390/appliedmath4030063

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrmr_linking_and_dff", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
people <- unique(ratings$Person)
baseline_data <- ratings[ratings$Person %in% people[1:24], , drop = FALSE]
new_data <- ratings[ratings$Person %in% people[25:48], , drop = FALSE]
baseline <- fit_mfrm(
  baseline_data, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", rating_min = 1, rating_max = 4,
  category_policy = "preserve"
)
anchors <- make_anchor_table(baseline, facets = "Criterion")
review <- review_mfrm_anchors(
  new_data, "Person", c("Rater", "Criterion"), "Score", anchors = anchors,
  rating_min = 1, rating_max = 4, category_policy = "preserve"
)
review$issue_counts
linked <- anchor_to_baseline(
  new_data, baseline, "Person", c("Rater", "Criterion"), "Score",
  anchor_facets = "Criterion", anchor_policy = "error",
  rating_min = 1, rating_max = 4, category_policy = "preserve"
)
linked$fit$facets$others
# Fixed-anchor differences are zero by construction, not evidence of stability.
linked$drift[, c("Facet", "Level", "Drift")]


## End(Not run)
```

`mfrmr_methods_reference` *What mfrmr Calculates and Where to Read More*

Description

Find an analysis by its question, understand what its numbers mean, and follow links to the calculation details and original references. For a first runnable analysis, use mfrmr_workflow_methods or `vignette("mfrmr-workflow", package = "mfrmr")` first. This is a help topic, opened with `help("mfrmr_methods_reference")`, not a function to run. For an integrated guide to **when to use each model and estimator**, read `vignette("mfrmr-model-choice", package = "mfrmr")`

or its online version. It compares practical uses and limitations, including why a more complex model is not automatically preferable. The sections below provide the calculation references after that choice.

Start with the question

Suppose several judges score performances on several criteria. A low score might reflect the performance, a stricter judge, a harder criterion, or a combination the chosen model does not explain. MFRM estimates these effects under a stated model. Raw agreement asks a different question: did judges give the same ratings to the same performances?

A **facet** is a modeled role, such as Rater or Task; a **level** is one rater or task. Person is specified separately in `fit_mfrm()`, so `facets = c("Rater", "Criterion")` gives three total facets including Person. More facets do not mean more ability dimensions. The rating assignments must contain information that distinguishes the requested effects. Arbitrary names and flexible facet counts do not make every crossed, nested or sparse design identifiable. Before interpreting a diagram, read the score-direction and diagnostic-term section of mfrmr_visual_diagnostics. It explains reverse-coded comprehensibility, calibrated judges and fit, conditional interaction bias, and adjacent-category Steps. A higher coded score is not inherently better.

Check the rating design before fitting

Start with `describe_mfrm_data()` and the intended comparison. A successful data review is not a certificate of identification or adequate precision.

1. **Count the information, not only Persons.** Inspect ratings per Person, observations per facet level and shared ratings. Adding raters while each Person still receives two ratings spreads the same workload across more rater effects. More Persons do not repair an assignment that cannot separate the effects. There is no package-wide minimum sample size or facet-count rule that guarantees reliable estimates or intervals.
2. **Separate the effects you want to compare.** For example, if Task A always uses Criterion A and Task B always uses Criterion B, their separate difficulties are confounded without additional information or constraints. Each Person-facet graph can be connected while this joint ambiguity remains. Review `design_connectivity`, then the estimator-specific identification and readiness results from `fit_mfrm()`. A local nonlinear rank check is not a proof of global identification.
3. **Declare the assignments and rubric.** Supply `expected_design` to the data review when the planned roster is known. Compare planned cells, observed cells and retained rating rows; repeated events need their own interpretation. An unassigned cell is not a zero score. Set the intended category bounds and inspect category use within the selected step owner. A category observed elsewhere may still be unsupported for that owner. See `rating_scale_table()` and `fit_mfrm()` for the model-specific checks.
4. **State what missingness and anchors mean.** Using available ratings does not correct ability-dependent assignment or score-dependent missingness. Missing population predictors have a separate `population_policy`. Shared Persons provide linking observations; they are not fixed anchors. `review_mfrm_anchors()` checks declared anchors and constraints. Treating an external estimate as an exact anchor does not propagate its uncertainty. Person anchors require JML; MML rejects them. By default, `centering_scope = "facet"` centers the whole declared facet, including fixed values. Use `"remainder"` only to reproduce legacy centering

of remaining ungrouped levels. Read mfrmr_linking_and_dff for a complete example, the distinction from FACETS, and the references supporting each linking claim.

5. **Check the requested output separately.** A fitted point, covariance, interval, residual analysis and G/D projection have different requirements. Flexible facet names/counts in one route do not extend another route. For example, two-family GPCM currently has exactly two non-Person facets; incomplete G-study observations do not authorize arbitrary unbalanced future D-study plans. Follow `gpcm_capability_matrix()`, `mfrmr_interval_guide()` and `mfrm_d_study()` for those limits.

Find the appropriate calculation

Follow the help link for arguments, equations, returned fields, examples and references. These are complementary analyses; running every row is not a requirement for every assessment.

- **Can my ratings support the intended comparison?** Retained ratings, category use, exposure and design/identification checks. Connectedness alone does not establish precision. `describe_mfrm_data()`, `subset_connectivity_report()` and `fit_mfrm()`; Linacre (1989).
- **How do ability, severity and category use explain scores?** Ordered-category response probabilities with specified facet effects. RSM shares steps; PCM assigns steps to one selected facet; GPCM additionally allows positive slopes. `fit_mfrm()` and `gpcm_capability_matrix()`; Andrich (1978), Masters (1982), Muraki (1992).
- **How are the parameters estimated?** MML integrates over a declared ability distribution; JML estimates Person parameters jointly with structural parameters. Native RSM/PCM CML conditions on each Person's total, eliminating ability without estimating a population distribution. It supplies structural conditional-information intervals and no ability population SD. Explicit comparison-only CML also supports identified interaction contrasts when other structural directions are unidentified; unavailable targets stay missing. These routes have different assumptions and uncertainty targets. The estimator and population sections of `fit_mfrm()`; Bock and Aitkin (1981), and the JML references on that page.
- **How uncertain is a measure or difference?** A model-specific SE or interval for a named target, conditional on specified assumptions. A difference may need the covariance between estimates. `mfrmr_interval_guide()`, `mfrm_facet_intervals()` and `mfrm_population_intervals()`; each page describes its calculation and limitations.
- **Which ratings are unexpected?** Infit/Outfit and residual summaries compare ratings with model predictions; ZSTD transforms fit using the selected reference convention. `diagnose_mfrm()` and `fit_measures_table()`; Wright and Masters (1982), Linacre (2002). Extended models use `mfrm_response_diagnostics()`.
- **Are categories functioning as intended?** Usage counts, owner-specific step estimates and probability curves describe category use and transitions. `rating_scale_table()`, `category_structure_report()` and `category_curves_report()`; Andrich (1978), Masters (1982). RSM has a common step set; PCM/GPCM has one declared step-owning facet, which can differ from the single slope-owning facet in GPCM. Review `owner_category_table` before interpreting pooled counts. Stored step SEs describe centered offsets, not full crossing, gap or curve uncertainty. Disorder does not automatically invalidate the model or justify collapsing categories. Two-family curves use their separate fitted-curve route with unavailable curve/step intervals; these ordinary helpers do not extend to it.

- **Is a rater unusually strict in a particular context?** Conditional bias screens describe departures from the additive model for a combination of facet levels. `estimate_bias()` and `estimate_all_bias()`; Linacre (1989) and the FACETS sources in their help. These screens hold calibration fixed. Use `bias_interaction_report()` to keep the chosen review rules with saved tables and plots. Its scatter compares observed-minus-expected score averages with conditional logit shifts; neither its normal/profile bounds nor repeated views provide a calibrated fairness test.
- **Are group differences specific to a facet?** DFF screens and separately matched MH/ordinal-logistic analyses use different adjustment models and sampling assumptions. `analyze_dff()`, `mantel_haenszel_table()` and `ordinal_logistic_table()`; see their References and matching rules.
- **How well are measures separated relative to error?** Separation, reliability and strata summarize estimated spread and measurement error for a facet. The interpretation and References sections of `diagnose_mfrm()` and plot(x, type = "reliability"); Wright and Masters (1982).
- **Do raters or scoring procedures agree?** Matched observed scores supply agreement coefficients, mean differences or ICCs. These do not estimate MFRM severity. `pairwise_kappa_table()`, `gwet_ac2_table()`, `krippendorff_alpha_table()`, `pairwise_icc_table()` and `score_comparison_table()`; each has method-specific References.
- **Do unexplained scores vary together?** Residual association concerns departures from the fitted model, not agreement of the original scores. `q3_statistic()` uses standardized residuals averaged within Person-by-level cells; it differs from Yen's raw-residual Q3. Its help identifies the original definition and the limits of the package's screening cutoffs. For organized residual patterns, `analyze_residual_pca()` distinguishes overall combined-facet columns from separate facet-level matrices. Eigenvalues across these different matrices do not rank facet importance. Its reference comparisons are exploratory; neither component counts nor fixed cutoffs establish a number of substantive abilities. Save the PCA result before repeated plots so the reference is not generated again.
- **Do raters match an expert standard or show centrality?** Agreement with supplied references and concentration of category use are separate rater properties. `fit_rater_accuracy()` and `rater_centrality_table()`; their References identify the accuracy and centrality methods and current restrictions. Centrality's RSM residual correlation, supporting fit and rater-owned PCM threshold SD are indirect indices with separate availability and source status, not an estimated centrality parameter or a severity ranking. `plot(fit, type = "rater_severity", diagnostics = diagnostics)` instead orders facet locations and returns SE sources and interval-use labels. Its whiskers are normal approximations, not likelihood-profile intervals or pairwise tests. See `plot_rater_severity_profile()` for interpretation and mfrmr_plot_migration for the public `plot()` entry point. `rater_halo_network_analysis()` instead compares observed-score profiles across shared contexts. Its graph centrality is a network property; neither its edge filters nor high same-rater correlations establish causal halo or residual independence. See the rater-effects section below for the distinction from centrality and the horn effect.
- **What does a cut score imply?** Posterior ability probabilities concern a Person now; classification accuracy/consistency concerns a specified future rating plan and population. `score_mfrm_persons()`, `predict_mfrm_units()` and `classification_table()`; the latter gives Rudner (2001, 2005) and Lee (2010), with DOI links.
- **Would another assessment plan be more dependable?** G-studies estimate observed-score variance/covariance components; D-studies project specified future plans and score weights.

`mfrm_g_study()` and `mfrm_d_study()`; Brennan (2001) and the cited design-specific methods. These are separate from MFRM separation reliability.

- **How do I retain the analysis and its caveats?** Saved results, figures, notes and exports preserve the calculation and its interpretation. `mfrm_results()`, `mfrm_report()`, `plot_data()` and `export_mfrm_results()`. See mfrm_calibration_workflow for eligible calibration reuse. Saving an analysis supports later review; a frozen calibration supports scoring new Persons at known facet levels; saving a score result supports later tables and plots without rescoring. These are different objects. Freezing makes a checked calibration available for scoring. It does not establish that the model or its prior suits another cohort. The calibration help links Bock and Mislevy (1982) to conditional EAP scoring, separately from calibration-estimation and interval-performance claims.

Use results as evidence about a construct

A construct is the attribute a score is intended to represent, such as speaking proficiency. Begin with a claim about that score and its use: "Content and Organization describe related aspects of performance across the specified tasks and raters." State the population, rubric, score units, intended composite or separate scores, and an expected pattern before inspecting results. A different claim, such as diagnosing an individual's weakness in Organization, needs evidence about that specific score or difference, not just the dependability of a total.

The *Standards* (AERA, APA, & NCME, 2014, Standards 1.13–1.14) require evidence for the proposed internal structure, subscore interpretation and composite rationale. Kane (2013) connects score-use claims to their assumptions and supporting evidence. The following is a guide to applying those principles with mfrmr, not a validity test implemented by the package.

Proposed claim	Inspect with mfrmr
Related criterion scores reflect shared differences between people.	G-study Person covariance, its uncertainty, and the other
The association is not solely a common scoring condition.	Rater, Task, Person-by-Rater and Person-by-Task covaria
The rubric and model represent the intended ordering.	Owner-specific category counts/curves, Wright-map targ
A proposed score is dependable enough for its stated decision.	G/Phi and SEM for the prespecified score, population an

Work through conflicting results. If G is high but Person covariance between the criteria is uncertain, the composite may be dependable while the proposed shared relation remains unresolved. If Person covariance is positive but Person-by-Task variation is substantial, consider whether the intended generalization spans those tasks. A task-specific strength may be substantive; it is error only relative to the chosen across-task target. If fit looks satisfactory but the claimed proficiency is poorly represented by the tasks, fit does not repair content underrepresentation.

Inspect all components rather than choosing favorable ones. Covariances depend on units, and a raw component is not its contribution to a future average: D-studies apply the relevant count divisors. Review negative or indefinite component estimates and interval qualifications. Separate pointwise intervals do not test equality between components. An interval containing zero neither proves independence nor establishes equivalence.

Choose reporting and follow-up from the claim. Prespecify composite weights for substantive reasons; optimizing G/Phi after inspecting the data can change the score being interpreted. Dependable subscores do not by themselves establish dependable or useful differences between them. MFRM and multivariate G-theory provide complementary evidence, not a joint fitted model or a

latent-dimensionality test. Agreement between outputs based on the same ratings is not independent replication.

Report the claim, design, observed pattern with uncertainty, competing explanation, justified conclusion and unresolved evidence. Where the claim requires it, examine task content, how raters reach judgments, or relations with an appropriate external measure. These require substantive evidence beyond the returned tables; the package does not certify validity or prescribe another analysis for every assessment. See the G-theory worked example and the figure reading guide.

Choose an agreement target and keep its denominator

Start with the same rated response, identified by all relevant keys such as Person, Task and Criterion. Matching several responses from one Person does not make them independent sampling units. The runnable `vignette("mfrmr-rating-agreement", package = "mfrmr")` shows matching, exclusions, interpretation and saving without fitting a measurement model.

- **Identical scores and model expectations:** `interrater_agreement_table()` reports exact agreement on matched contexts. Its `ExpectedExact` averages products of fitted category probabilities, with calibration fixed. This is a conditional model expectation, not the chance-agreement term in kappa or AC2. MML uses a plug-in EAP, not posterior integration. Unavailable primary estimates withhold expectations with an `ExpectedReason`; observed agreement is retained. Overall exact agreement counts every matched pair-context opportunity; a context rated by more judges supplies more pairs.
- **Category agreement for two raters:** `pairwise_kappa_table()` uses the same common complete units for both rater margins. QWK penalizes category differences quadratically. Numeric categories use their actual spacing; character categories use their declared positions. Each pair may use a different sample, so these coefficients are not a panel average.
- **Pooled agreement:** `krippendorff_alpha_table()` excludes singleton units from both disagreements and weights pairable units by rating count. Its ordinal distances depend on pairable-category frequencies. `gwet_ac2_table()` weights pairable units equally for observed agreement; singletons enter its expected-agreement margins. Its declared unused categories also matter. Choose the definition before seeing which value is larger; neither numerical missing-data convention corrects selective assignment or missingness.
- **Reliability on an interval score scale:** `pairwise_icc_table()` requires choices about selected versus randomly sampled raters, absolute agreement versus consistency, and one rating versus the mean of two. Consistency excludes systematic rater offsets; agreement includes them. The default concerns a single rating from selected raters. An average-rating pairwise ICC concerns two ratings, not the whole panel; numeric category codes alone do not justify an interval scale.
- **A score source against a supplied reference:** `score_comparison_table()` separates QWK, correlation and standardized mean difference. Adding a second reference uses common complete records for all statistics and can change the sample. A chosen reference is not automatically correct, and the optional Williamson et al. (2012) screen is an application-specific choice rather than a universal quality standard.

The raw-score helpers above return descriptive point estimates with their counts, cases, settings and notes; they do not provide sampling intervals. Save their **complete lists** with `base::saveRDS()` and export chosen tables with `utils::write.csv()`. They are not fitted-model result bundles for

`export_mfrm_results()`, and their custom tutorial figures are not new `plot()` methods. The separate `mfrm_interrater` bundle supports `summary()` and `plot()`; see `interrater_agreement_table()` for its different repeated- cell handling. Low residual association does not establish raw agreement, and high agreement does not establish accuracy against an expert standard.

Distinguish rater effects before choosing a diagnostic

Consider a judge scoring content, organization and language on a four-category rubric. These questions concern different features of the ratings:

- **Severity:** does the judge generally give lower scores after accounting for the modeled Person and criterion effects? Review fitted Rater locations.
- **Centrality:** does the judge compress ratings toward middle categories? `rater_centrality_table()` combines the supplied RSM residual/fit and rater-specific PCM threshold descriptions. A severity near zero does not answer this question; neither does graph-theoretic centrality.
- **Halo or horn:** does a favorable or unfavorable impression spill over into ratings on other criteria? For example, a poor impression of language could lower content and organization ratings too. The latter pattern is called a horn effect; it is not defined by a negative criterion correlation. `rater_halo_network_analysis()` describes observed-score associations and cannot determine their psychological cause or classify halo versus horn. Genuine relationships among performance criteria also produce correlations.
- **Local dependence:** do ratings share variation beyond the chosen model? A supported `fit_mfrm_testlet()` model can account for a declared shared effect, but its variance does not identify halo or horn.

Infit/Outfit summarize squared residuals. Low values can have several causes and do not identify which effect occurred or its positive/negative direction. Inspect shared-rating counts, category use and the actual performances before drawing conclusions about a rater. Agreement among several indices computed from the same ratings is not independent confirmation.

Direct models answer a further question: how would estimated measurements change when a specified rater effect is included in the response model? The FM-SC severity/centrality model and the MRFM-H halo mixture models are **not implemented**. See the original references and model distinctions in `rater_centrality_table()` and `rater_halo_network_analysis()`. These models require their own identifying assumptions and uncertainty evaluation; adding GPCM slopes or a testlet variance does not implement them.

Choose the G-theory target and uncertainty

Start with the rating design and the intended score, not the word "reliability" alone. The multivariate route also accepts **one score**; multiple score columns are fixed score components, not additional random facets or latent dimensions. All methods below require explicit `assumption = "normal"`, prespecified scores/weights and the documented sampling/assignment assumptions. An incomplete source is not permission to use a sparse future plan: current D-studies project complete balanced plans.

First choose the quantity whose uncertainty you need:

Question	Interface
How uncertain is a source variance or covariance?	`confint.mfrm_multivariate_gstudy()`
How uncertain are the original component estimates jointly?	`vcov.mfrm_multivariate_gstudy()`

How uncertain is a future plan's G, Phi or SEM?	`confint.mfrm_multivariate_d_study()`
How uncertain is the change between two plans?	`confint(d, reference = 1, ...)` or `mfrm_multivariate_d_`

Then select a method allowed by the **observed source**. Here `g` is a G-study and `d` its D-study. The table describes available calculations, not a ranking by interval width or a guarantee of nominal coverage.

Target and source	Default or explicit choices
Component variance, balanced ANOVA	`confint(g)`: `auto`/`mls`
Signed covariance between two scores, balanced ANOVA	Explicit `gci` or `regression_region`
One-facet G/Phi, balanced ANOVA	`confint(d)`: `exact_f` for G, `gci` for Phi
Two-crossed-facet G/Phi, balanced ANOVA	Default `mls`; explicit `delta`, `approximate_f`, `mean_`
Nested G/Phi, balanced ANOVA	With unchanged source child count per parent, the o
Relative/absolute SEM, balanced crossed/nested ANOVA	Default `mls`; two-facet sources also permit `mean_squ`
G/Phi, identifiable MINQUE source	Explicit `delta` or `reml_profile`; one-facet G also p
Component variance/covariance or SEM, identifiable MINQUE source	Explicit `reml_profile`; `residual_contrast` additi

`delta` is also an explicit G/Phi option for balanced one-facet sources; `reml_profile` accepts every supported balanced source as well as identifiable incomplete/unequal sources. `residual_contrast` and one-facet `spectral_f` also accept retained complete ratings. These methods do not become an automatic fallback when another calculation fails. MINQUE requests require an explicit interval method even when the source happens to be balanced. For derivations, required inputs, optional dependencies and references, use `confint.mfrm_multivariate_gstudy()` and `confint.mfrm_multivariate_d_study()`. Almehrizi's (2021) coefficient-SE formula and the limits of its extension are in the latter's *Delta coefficient SEs and intervals* section; SE accuracy does not establish Wald coverage.

For **paired plans**, choose the relevant source law separately:

- Balanced one-facet G/Phi map the whole common F/generalized set. The nested G/Phi reduction requires every future child count to equal the source count. One-facet SEM and nested SEM with a common future child count map one error-SD interval; that future child count may differ from the source.
- General balanced two-facet comparisons have explicit `mean_square_region` enclosures from one shared source region. These retain dependence but can be wider than the tight difference image. Two crossed random facets also permit `mfrm_multivariate_d_compare()` for a separately qualified paired delta approximation, including identifiable MINQUE sources.
- Explicit `reml_profile` supplies G/Phi/SEM comparisons for supported complete or incomplete sources. Proportional one-facet/common-child nested plans map a common profile; other plans profile the difference directly. Calibration remains approximate, especially at zero-contrast singularities.
- For retained one-facet ratings, `spectral_f` G and `residual_contrast` RelativeSEM differences map their shared confidence sets. Marginal `method = "delta"` in `confint(d)` does not supply paired differences.

Read and save the result. `summary.mfrm_multivariate_d_study()` with `intervals = TRUE` collects supported methods, original points, any separate `IntervalCenter`, source/future counts

and weights. Select `parm` explicitly when the chosen method supports only some metrics. Ordinary `summary(d)` returns the coefficient table. `confint(d, method = c(...))` compares compatible methods for one target; `confint(g)` selects one component method at a time.

Wide computable intervals remain results. Delta limits outside `[0,1]`, negative original components, empty sets, failed numerical checks and absent methods are different states. Read `Status`, availability and saved notes. A REML or residual-projection center must not be relabeled as MINQUE. General bootstrap intervals are not implemented. Nonnormal robustness, selection-adjusted inference and sparse/unequal future plans are not supplied. These limits do not make an implemented interval calculation nonexistent.

There is no universal acceptable width or G/Phi cutoff. Define the intended score use, a meaningful improvement or threshold, and task/rater costs before choosing a plan. An interval spanning a decision threshold describes uncertainty about that decision; selecting the narrowest method after seeing its result does not resolve it. A method comparison on one dataset is sensitivity analysis, not a coverage study. Neither high G/Phi nor a narrow interval alone establishes construct validity or fairness.

Use `plot(d)` for point curves, `intervals = TRUE` for an explicit interval request, or supply saved intervals for reuse. `plot(ci)` compares D-study intervals on one axis; `plot(g, intervals = TRUE, ...)` shows component intervals and retains them in the returned plot object. Save complete results with `saveRDS()`; `as_ggplot()` renders saved plot data without fitting. Ribbons are pointwise, not simultaneous bands; overlap is not a paired test. The G-theory tutorial supplies a bundled-data reading path, and the function help holds mathematical details and references. The simpler `mfrm_generalizability()` / `mfrm_d_study()` route is a separate main-effects approximation; its residual sensitivity is not a confidence interval.

Read the method before interpreting the number

Model: RSM/PCM use unit discrimination. One-family GPCM assigns slopes to one selected facet; MML allows a different step owner, whereas JML requires the same owner. The experimental two-family MML route has exactly two non-Person facets and additional restrictions. Check `gpcm_capability_matrix()` before transferring an ordinary-model analysis. The original response-model papers do not establish the performance of every mfrmr estimator, extension or interval.

Estimation and scoring: MML uses numerical integration of the marginal likelihood. JML includes Person parameters in the joint likelihood. EAP scoring subsequently averages over a conditional ability distribution; its posterior SD is not a sampling SE for a structural parameter. Ordinary RSM/PCM MML fixes N(0,1) when `population_formula = NULL`; estimating a population model is an explicit choice. A histogram of EAP estimates is not itself an estimate of the latent population density. Numerical integration, population assumptions and the source of uncertainty are documented separately in `fit_mfrm()` and the scoring help pages.

Fit: Infit weights squared standardized residuals by their model variance; Outfit gives each retained residual equal weight when observation weights are all one. Values above one indicate more residual variation than the reference model expects. A flag is a reason to inspect ratings and assumptions, not a verdict on a rater. Read the selected residual basis, weighting and ZSTD convention in `diagnose_mfrm()`. Posterior diagnostics for extended models do not inherit ordinary fit cutoffs.

Reliability and strata: the ordinary summaries adjust the spread of finite measures using their matching squared SEs. Invalid or missing SEs can make the summary unavailable. High rater separation indicates distinguishable severities, not agreement; low separation can also reflect poor precision. Strata is a hypothetical separation index, not a discovered number of groups or a set

of pass/fail boundaries. Model and fit-adjusted summaries are not confidence bounds. See the Facets explanation alongside mfrmr's estimator-specific conventions in `diagnose_mfrm()`. These marginal-SE summaries omit covariance among calibrated measures. Applying variance subtraction to EAP estimates is not posterior-variance EAP reliability, and neither quantity is a G-theory coefficient.

Post-fit shrinkage: `apply_empirical_bayes_shrinkage()` adds a descriptive zero-centered normal-model adjustment while retaining the fitted estimates. Its plug-in SD conditions on the chosen prior variance; it is not a calibrated sampling SE or a general improvement guarantee. Additional Person shrinkage requires ordinary JML sampling SEs. MML EAP estimates and posterior SDs must not be fed through a second such update.

Categories and missingness: use the intended rubric, not only the observed score range. `category_policy = "preserve"` retains that ladder; `"collapse"` changes a gapped ladder and records the mapping. Missing assigned scores, unassigned ratings and unavailable rubric categories are different situations. The current ordinary models use a common category range; owner-specific ranges are not implied by PCM step ownership. See `describe_mfrm_data()` and `fit_mfrm()` for support checks and retained rows.

Uncertainty: distinguish the estimate, its numerical status and the interpretation allowed by its assumptions. A finite experimental interval is not evidence of nominal coverage. A missing endpoint is not infinity. Conditional bias screens and fixed-calibration Person intervals omit calibration uncertainty. Use `mfrmr_interval_guide()` to select the target before choosing a method.

Find and cite the supporting literature

Open the function's **Details**, named method sections and **References**; report its selected settings as well as the paper. For example, `help("diagnose_mfrm")` connects the Infit/Outfit definitions to the ZSTD conventions and reliability formulas; `help("classification_table")` explains how the Lee recursion differs from the Rudner approximation. The foundational sources below identify the model/estimation framework. Specialist help pages provide the references for their own formulas and adaptations. A citation to a general framework is not validation of a new implementation or of your data's assumptions.

Use `citation("mfrmr")` for the software citation and `packageVersion("mfrmr")` for the installed version. Cite the relevant methodological sources as well. A DOI identifies a paper even if access to its full text requires a library subscription.

References

American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (2014). *Standards for educational and psychological testing*. American Educational Research Association. Internal structure and score-combination rationale: Standards 1.13–1.14, pp. 26–27; reliability/precision, Chapter 2. https://www.testingstandards.net/uploads/7/6/6/4/76643089/standards_2014edition.pdf.

Kane, M. T. (2013). Validating the interpretations and uses of test scores. *Journal of Educational Measurement, 50*(1), 1–73. doi:10.1111/jedm.12000. A framework for claims, assumptions and evidence; not validation of a particular mfrmr calculation.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*, 561-573. doi:10.1007/BF02293814. RSM foundation.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*, 149-174. doi:10.1007/BF02296272. PCM foundation.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176. doi:10.1177/014662169201600206. GPCM foundation.

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika, 46*, 443-459. doi:10.1007/BF02293801. Marginal-likelihood estimation framework; this citation does not mean every mfrmr engine uses EM.

Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press. Many-facet formulation and interpretation.

Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press. Fit, separation and reliability conventions.

Linacre, J. M. (2002). What do Infit and Outfit, Mean-square and Standardized mean? *Rasch Measurement Transactions, 16*(2), 878. https://www.rasch.org/rmt/rmt162f.htm. An introductory fit explanation.

Brennan, R. L. (2003). *Coefficients and indices in generalizability theory* (CASMA Research Report No. 1). University of Iowa. Pages 3–4 introduce random-facet G/D error and coefficient targets. https://education.uiowa.edu/sites/education.uiowa.edu/files/2026-04/casma-research-report-1-archived.pdf.

See Also

mfrmr_workflow_methods, `mfrmr_output_guide()`, `mfrmr_interval_guide()`, `gpcm_capability_matrix()`

`mfrmr_model_maps` *Wright and Pathway maps for extended RSMs*

Description

Plot a saved `mfrm_results()` object using `type = "wright"` or `type = "fit_pathway"`. These routes support testlet and shared-rater models; ordinary-model maps keep their existing definitions.

Locations and their meaning

The fitted equation is log(P(k)/P(k-1)) = ability + local effect - sum(facet severities) - step(k). A fixed facet's displayed location is its severity plus the unweighted mean of the fitted steps. It is the mean adjacent-category boundary when all other facet severities and local effects are zero. It is not the ability producing the middle expected score. Shared raters use the fitted conditional rater mode plus that step mean; replacement-rater marginal predictions are a different target. A testlet effect is Person-local and has no separate global facet column.

Person positions are saved conditional EAPs on the fitted mean-zero ability scale. The separate category-boundary column shows fitted steps at zero facet/local effects. These locations are conditional references: integrating latent effects does not generally preserve these category crossings. `SourceEstimate` and `StepCenter` retain each transformation in plot tables. Do not add displayed

facet locations to obtain a combined rating boundary: this would count the mean step repeatedly. Use the fitted equation instead. Whiskers, when requested, show **only Person conditional intervals**. Composite facet/step uncertainty is not obtained by shifting an existing facet interval; use the separate estimate plots for their original targets.

Saved inputs

Supply source-roster scores through `mfrm_results(fit, scores = scores)` for either extension (testlet `predictions = scores` remains supported). Calculate them explicitly using `score_mfrm_persons()`. Wright maps require matching scores; older testlet scores without `scoring_data` need rescoring, not refitting. The source calibration and complete rating-event multiset must match. Shared-rater scores computed from a reduced roster cannot be combined with rater modes from the full fitted roster.

Pathways also require separately saved `mfrm_response_diagnostics()` output. `fit_stat = "Infit"` (default) or `"Outfit"` chooses the horizontal axis; the vertical axis uses the locations defined above. Selected-row counts and full-roster conditioning are retained. A residual index for a subset of ratings describes that subset, not the group's entire workload. Missing or unresolved indices remain in tables but cannot be drawn. No expectation-one line, acceptance band, ZSTD, bias test or rater-quality classification is used. A pathway can display fixed facets/raters without Person scores.

Display controls

`facet = NULL` shows all available location panels in a Wright map and all located diagnostic groups in a pathway; supply fitted column names to select panels. `persons = NULL` retains all saved Person scores; a character vector selects saved IDs without rescoring. `show_steps = TRUE` includes the separate Wright category column; steps do not appear in pathways. `show_intervals = TRUE` shows available Person conditional intervals. `show_labels` defaults to `FALSE` for Wright maps and `TRUE` for pathways; set it to `FALSE` for crowded pathways. `palette` is `"accessible"` or `"mono"`; shapes also distinguish the location types and prior-only results. A prior-only point is not a measured ability. `title`/`caption` replace defaults; `show_title`/`show_notes` hide annotations without removing interpretation metadata. `text_scale` and `point_size` change sizes, and `draw = FALSE` returns saved plot data. `as_ggplot()` preserves these choices and supplies alternative text. Plot tables retain all selected unavailable rows; `locations` also retains undisplayed panels. No fitting, scoring or integration is performed during plotting/export.

Examples

```
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
res <- mfrm_results(example$testlet$fit,
  scores = example$testlet$scores,
  response_diagnostics = example$testlet$diagnostics, compute = "never")
plot(res, type = "wright")
pathway <- plot(res, type = "fit_pathway", facet = "Rater", draw = FALSE)
if (requireNamespace("ggplot2", quietly = TRUE)) as_ggplot(pathway)
```

`mfrmr_output_guide` *Choose an mfrmr output helper by user goal*

Description

Find a function for the next question in your analysis. Start with `mfrmr_output_guide("beginner")` and read its `Question` and `MainFunction` columns. With no argument the function returns the full specialist catalogue, which includes tables, reports, reviews and exports. It does not analyze data.

Usage

```
mfrmr_output_guide(
  scope = c("all", "public", "beginner", "psychometric", "entry", "viewer", "binary",
    "tables", "reports", "reviews", "bundles", "exports", "compatibility", "gpcm",
    "calibration", "simulation", "linking", "network", "response_time", "facets",
    "conquest", "r", "models", "features", "imputation", "gtheory", "feedback", "plots")
)
```

Arguments

`scope`	Which rows to return. `"all"` returns the full route catalogue. `"plots"` returns a separate purpose-based plot capability table (see below). `"public"` returns the canonical six-step route for most users; `"beginner"` returns the same compact route rather than combining every beginner-labelled specialist row. `"entry"` returns the recommended first-screen routes. `"viewer"` returns local-viewer routes built around `mfrm_results(include = ...)`. `"binary"` returns the two-category person-item Rasch route and checks. Other values filter to one output family or to `GPCM`-relevant routes. `"linking"` returns anchor, drift, equating and DIF sensitivity routes. `"calibration"` returns the portable fixed-calibration lifecycle and artifact-only scoring route, conditional decisions and fixed-plan classification accuracy/consistency. `"simulation"` and `"network"` return advanced design-review rows. `"response_time"` returns descriptive response-time QC rows. `"models"` compares fixed-facet, shared-rater and Person-specific testlet workflows, including their distinct prediction and reporting boundaries. `"feedback"` distinguishes fixed-rater uncertainty, unexpected rating patterns, shared-rater uncertainty, observed agreement, centrality, criterion-referenced accuracy and screening accuracy with known truth. `"features"`, `"imputation"` and `"gtheory"` show exploratory attributes, assigned-score multiple imputation and observed-score G/D-study workflows, including their own table, plot and saving routes. `"facets"`, `"conquest"`, and `"r"` return user-pathway rows for people arriving from those workflows.

Details

Naming convention used by the guide:

- `*_table`: focused table or table-like result for one evidence source
- `*_report`: multi-table evidence bundle for a reporting question

- *_review: status, interpretation, or decision-support object
- *_bundle: reusable collection of tables/metadata for handoff
- export_*: writes files or appendix artifacts

Value

For scope = "plots", a data.frame with Question, InputClass, PlotCall, PlotName (the name in the saved plot data), DataComponent, GGPlot, GGPlotCall (NA when unavailable), Notes, ResultFunction and NextStep. All other scopes return a data.frame with one row per route and columns:

- Scope
- Question
- OutputFamily
- Lifecycle
- UserLevel
- APILayer
- ObjectRole
- DecisionBoundary
- RecommendedEntry
- MainFunction
- UseWhen
- TypicalInput
- NextStep
- GPCMStatus
- Notes

First-screen route

Use mfrmr_output_guide("public") or mfrmr_output_guide("beginner") for the shortest top-level API map: an explicit describe_mfrm_data() check and fit_mfrm() MML fit, the lightweight fit summary, res <- mfrm_results(fit) and its triage, optional figures, and report/export or RDS saving. Plotting is not required before saving. Earlier scripts may obtain the same result bundle from summary(fit, profile = "facets")$results; no FACETS software is required. Use mfrmr_output_guide("entry") when you specifically need alternative first-screen creation routes, including existing result objects, the optional viewer, or interactive console work. After creating res, use summary(res)$next_actions to choose a purpose-specific specialist helper. Use mfrmr_output_guide("viewer") when the next step is the optional local Shiny reader; it shows which include preset to use before calling launch_mfrmr_viewer(). Use mfrmr_output_guide("feedback") when preparing rater feedback. Start with the question and the fitted model: severity, response misfit and the accuracy of a warning rule are different quantities. A severe rater need not misfit, and an observed flag does not establish poor rater quality. For observed agreement, choose pairwise_kappa_table(), krippendorff_alpha_table(), gwet_ac2_table() or pairwise_icc_table() according to the score scale, rater target and missingness assumptions. score_comparison_table() compares a score source with reference ratings; rater_centrality_table() describes residual

and threshold-spread indices; `fit_rater_accuracy()` models exact matches to fixed reference ratings. High rater separation reliability indicates distinguishable severities, not high agreement. These quantities do not replace one another. For a fixed future plan, use `classification_table()` under `scope = "calibration"`; its model-based accuracy/consistency differs from observed reference agreement and posterior ability-band probabilities. For DIF sensitivity, `scope = "linking"` points to `mantel_haenszel_table()`, `ordinal_logistic_table()` and the shared-roster example in `vignette("mfrmr-linking-and-dff")`. For an individual native additive RSM/PCM or experimental two-family GPCM sheet, use `mfrm_report(res, style = "rater", facet = "Rater", rater = "R01")`. The selected sheet omits source identifiers; the comprehensive result and export bundle retain the original analysis. Use `mfrmr_output_guide("psychometric")` for the technical table, review, and reporting routes whose interpretation boundaries should be checked before manuscript use.

How to use this guide

For route scopes, read `Question` first, then open the help for a function in `MainFunction`. For `"plots"`, use the figure instructions below. `UseWhen` describes its inputs and purpose; `NextStep` explains what to inspect. Cells containing `...` are outlines, not complete scripts to paste and run. Use mfrmr_workflow_methods for a runnable introduction and an explanation of function names. Inspect `DecisionBoundary` before interpreting a result. For construct-focused research, the *Use results as evidence about a construct* section of mfrmr_methods_reference connects a claim to observable patterns, competing explanations and the limits of a conclusion. For `GPCM`, use `scope = "gpcm"` to find both the support matrix and the table that explains how out-of-scope routes are handled. Use `scope = "exports"` to distinguish saved-analysis archives from `export_mfrm_model()`, which generates a Stan/brms model for a separate Bayesian analysis with explicit priors. That route returns to mfrmr through `read_mfrm_posterior()`: inspect `summary(x)` before using `plot(x)`, `confint(x)` or `mfrm_report()`. Other interval levels, contrasts and predictions require the backend's joint draws.

Check the model before following a route

The six-step beginner route starts with ordinary RSM/PCM. The result class `mfrm_fit` alone does not establish which diagnostics or figures are valid: one-family GPCM, two-family GPCM and corrected JML share that class. Read the fitted model and estimation method in `summary(fit)` first. For GPCM, consult `gpcm_capability_matrix()` and `vignette("mfrmr-gpcm-scope", package = "mfrmr")`. Two-family GPCM and corrected JML have separate result/report and descriptive response-diagnostic routes; ordinary fit tests and Wright/Pathway maps do not apply. Corrected JML supplies point plots for slopes, locations and steps, without structural confidence intervals.

Use `scope = "models"` for fixed-facet, shared-rater and testlet roles, `scope = "feedback"` for the distinct rater-review targets, and `scope = "calibration"` for scoring or fixed-plan decisions. For exploratory bias review across eligible facet pairs, use `estimate_all_bias()`, followed by `summary(screens)` and `plot(screens)`. Use `estimate_bias()` for a prespecified pair. Retain failed pairs and unavailable cells; the collection does not supply multiplicity-controlled inference or evidence that no bias exists. Save the complete collection.

Use `scope = "gtheory"` for observed-score assessment planning. Start with the G-study component or covariance view, then compare D-study curves while keeping other facet counts fixed or in separate panels. Use `plot(d, type = "error_components")` to inspect each plan's error sources, including signed covariance contributions, or `type = "sem"` for error in score units. Error-source con-

tributions are point estimates; SEM uncertainty requires the separately supported interval calculation. These projections and paired plan-difference intervals answer different questions. Component SEs/intervals are available for the designs and methods documented in `confint.mfrm_multivariate_gstudy()` and `vcov.mfrm_multivariate_gstudy()`. For coefficient sampling SEs, use `confint.mfrm_multivariate_d_study()` with `method = "delta"` and `assumption = "normal"`. These differ from measurement SEM. Its normal-reference intervals and plots retain bounds outside `[0,1]` with approximation qualifications; defaults are unchanged. General distribution-robust comparison intervals are not supplied. Save each complete result with `base::saveRDS()` and reopen it with `base::readRDS()`; a CSV alone does not retain its design or qualifications. Fixed-facet output also depends on the estimator: native CML uses dedicated summaries and `mfrm_facet_intervals()` plots/tables, without ordinary Person scoring or consolidated reports. `mfrmr_interval_guide()` separates that route from MML interaction contrasts and their population-SD standardization.

Choosing a figure

Use `mfrmr_output_guide("plots")` to find selected plots by purpose, input class and explicit plot call. This is a curated map, not an exhaustive list of every plot method, view or component. Unlisted routes are not necessarily unsupported. Existing scopes, including `"all"`, keep their route-table format.

For linked figure previews and runnable examples, open `vignette("mfrmr-visual-diagnostics")`. `ResultFunction` names a help page for creating the required result. In `PlotCall`, replace `x` with the indicated result object and save the returned object as `p`. Calls use `draw = FALSE`; change it to `TRUE` to display the original figure. Then follow `GGPlotCall`, if available. `DataComponent` is for `plot_data()` extraction; it is not automatically a valid or equivalent `component` argument to `as_ggplot()`.

`GGPlot` distinguishes `"dedicated"` (a converter for this plot), `"native"` (the plot method already returns ggplot), `"generic"` (a table-based view that need not reproduce the original figure), and `"unavailable"` (use the original plot or build a custom graphic from its data). ggplot routes need the optional ggplot2 package. Display support does not broaden the fitted model's statistical scope or establish interval coverage. Read `Notes` and the source function's help before interpretation.

Examples

```
beginner <- mfrmr_output_guide("beginner")
beginner[, c("Question", "MainFunction", "NextStep")]

# Ask for a specialist map only when that question arises.
linking <- mfrmr_output_guide("linking")
linking[, c("Question", "MainFunction", "UseWhen")]

feedback <- mfrmr_output_guide("feedback")
feedback[, c("Question", "MainFunction", "DecisionBoundary")]

figures <- mfrmr_output_guide("plots")
figures[, c("Question", "ResultFunction", "GGPlot")]
```

`mfrmr_plot_migration` *Plotting after the 0.2.4 API consolidation*

Description

Use `plot()` on a fitted model or an analysis result. The 33 former drawing helpers are no longer exported in 0.2.4. This is a breaking API change: replace old `plot_*()` calls using the table below. Their numerical/drawing implementations remain internal; do not depend on `mfrmr:::` access. The public data accessors `plot_data()` and `plot_data_components()` remain.

Choosing an input

Use a fitted ordinary model for fit-level views. Use saved G/D studies, information, DIF/DFF, bias, QC or PCA results for their own `plot()` methods. Their public help pages explain the view controls, returned tables and interpretation: `plot.mfrm_information()`, `plot.mfrm_dif()`, `plot.mfrm_dif_interaction()`, `plot.mfrm_anchor_drift()`, `plot.mfrm_qc_pipeline()`, `plot.mfrm_bias()`, `plot.mfrm_residual_pca()` and `plot.mfrm_diagnostics()`. `plot.mfrm_population_intervals()` adds saved latent-regression coefficients and contrasts. Start with these pages for saved results. A common function name does not make every view available for every model. Two-family GPCM and corrected JML retain their separate supported result routes. Plotting does not add intervals, repair unavailable values or equate scales. Use original result objects rather than unclassed lists or CSV tables.

Single-fit views

`plot(fit)` still draws the ordinary Wright map. New `type` choices include `reliability`, `agreement`, `person_fit`, `rater_severity`, `thresholds`, `residual_qq`, `residual_matrix`, `residual_pca`, `local_dependence`, `marginal_fit`, `marginal_pairwise`, `fair_average`, `displacement`, `facets_chisq`, `bubble`, `guttman`, `unexpected`, `qc`, `apa_summary`, `bias`, `facet_equivalence` and `facet_dashboard`.

Use `metric` for the quantity and `view` for a display variant. Reliability accepts `reliability`, `separation` or `strata`. Agreement accepts `exact` or `correlation`, with `view = "pairs"` (default) or `"heatmap"`; `metric = "difference"` applies only to pairs. Severity separation is not agreement. The former `plot_type` argument becomes `view` for marginal fit, fair averages, displacement, facet chi-square, unexpected ratings and facet dashboards. Keep other named controls from the legacy parameter help.

Explicit controls are passed only to the chosen renderer; omitted controls retain that renderer's defaults. Inapplicable and conflicting controls are rejected. Supply matching saved `diagnostics` for repeated fit-level views. Some legacy fit-level convenience routes compute diagnostics when omitted; this does not imply model refitting. Saved-result plots reuse their results.

Replacement table

In the table, `x` is the former first argument, including its original result class. For paired models use `reference` and `comparison`. For an ordered trajectory, `fits` is the existing named list of fitted models.

Former exported helper	Replacement
`plot_anchor_drift()`	`plot(x)`
`plot_apa_figure_one()`	`plot(x, type = "apa_summary")`
`plot_bias_interaction()`	`plot(x, type = "bias")`
`plot_bubble()`	`plot(x, type = "bubble")`
`plot_compare_mfrm()`	`plot(reference, comparison = comparison)`
`plot_dif_heatmap()`	`plot(x, type = "heatmap")`
`plot_dif_summary()`	`plot(x, type = "summary")`
`plot_displacement()`	`plot(x, type = "displacement")`
`plot_facet_equivalence()`	`plot(x, type = "facet_equivalence")`
`plot_facet_quality_dashboard()`	`plot(x, type = "facet_dashboard")`
`plot_facets_chisq()`	`plot(x, type = "facets_chisq")`
`plot_fair_average()`	`plot(x, type = "fair_average")`
`plot_guttman_scalogram()`	`plot(x, type = "guttman")`
`plot_information()`	`plot(x)`
`plot_interrater_agreement()`	`plot(x, type = "agreement")`
`plot_local_dependence_heatmap()`	`plot(x, type = "local_dependence")`
`plot_marginal_fit()`	`plot(x, type = "marginal_fit")`
`plot_marginal_pairwise()`	`plot(x, type = "marginal_pairwise")`
`plot_person_fit()`	`plot(x, type = "person_fit")`
`plot_qc_dashboard()`	`plot(x, type = "qc")`
`plot_qc_pipeline()`	`plot(x)`
`plot_rater_agreement_heatmap()`	`plot(x, type = "agreement", view = "heatmap")`
`plot_rater_severity_profile()`	`plot(x, type = "rater_severity")`
`plot_rater_trajectory()`	`plot((fits)[[1L]], type = "trajectory", comparison = (fits)[-1L], label`
`plot_reliability_snapshot()`	`plot(x, type = "reliability")`
`plot_residual_matrix()`	`plot(x, type = "residual_matrix")`
`plot_residual_pca()`	`plot(x, type = "residual_pca")`
`plot_residual_qq()`	`plot(x, type = "residual_qq")`
`plot_response_time_review()`	`plot(x)`
`plot_shrinkage_funnel()`	`plot(x, type = "shrinkage", view = "funnel")`
`plot_threshold_ladder()`	`plot(x, type = "thresholds")`
`plot_unexpected()`	`plot(x, type = "unexpected")`
`plot_wright_unified()`	`plot(x, type = "wright", view = "unified")`

For result objects the object determines the target: for example `plot(info, type = "iif", facet = "Rater")` or `plot(dif, type = "heatmap", metric = "contrast")`. For the fit convenience routes, rename the old bias `plot` argument to `view`; rename the old equivalence `type` to `view`. For residual PCA rename `plot_type` to `view` with `type = "residual_pca"`, or use `plot(pca, type = "loadings")` directly on a saved PCA result.

Multiple models and occasions

Use `plot(reference, comparison = other, type = "ccc")` or `type = "wright"` for paired displays; `view = "difference"` shows differences without adding difference SEs or model-selection tests. For trajectories use `plot(fits[[1]], type = "trajectory", comparison = fits[-1], labels = names(fits))`. Labels include the first model, and the supplied order is retained. Models must

already be placed on an interpretable common scale; plotting performs no linking.

Returned data and appearance

Use `draw = FALSE` to obtain the existing payload without drawing. Most routes return `mfrm_plot_data`; the QC pipeline retains its verdict-table return, and native interval routes retain their existing ggplot results. Return classes have not been homogenized. Use `plot_data()` for supported payloads and `as_ggplot()` only for supported conversions. Unsupported conversions do not become available through this API change. Titles and cautions stay outside figures by default; keep returned notes when exporting. G/D plots retain their existing color, monochrome, score and facet-panel controls. Save full result objects for later plotting.

See Also

`plot.mfrm_fit()`, mfrmr_visual_diagnostics, `mfrmr_output_guide()`

---

mfrmr_reporting_and_apa

mfrmr Reporting and APA Guide

Description

Package-native guide to moving from fitted model objects to manuscript-draft text, tables, notes, and revision checklists in `mfrmr`.

This guide applies to ordinary MML/JML `RSM` / `PCM` reporting, with the estimator-specific limits on structural inference. Native CML and comparison-only fits do not supply this consolidated reporting workflow. For CML, use `summary(fit)` and `mfrm_facet_intervals()`, then `summary(ci)`, `plot(ci)`, `apa_table()` and `saveRDS(ci, file)`; see `vignette("mfrmr-facet-intervals", package = "mfrmr")`. Ordinary one-family `GPCM` fits support `reporting_checklist()`, `precision_review_report()`, direct curve/graph and residual table helpers, and caveated APA/QC/export bundles. Use `gpcm_capability_matrix()` when you need the formal boundary for the current `GPCM` reporting path.

In particular, `GPCM` `build_apa_outputs()`, `build_visual_summaries()`, `run_qc_pipeline()`, `build_mfrm_manifest()`, `build_mfrm_replay_script()`, and `export_mfrm_bundle()` outputs include explicit `gpcm_boundary` caveats. Full FACETS-style score-side contract review remains blocked. Scorefile export, design forecasting, diagnostic/signal-detection screening, and linking synthesis use their own caveated `GPCM` routes and should not be treated as automatic operational-scoring evidence.

See `vignette("mfrmr-networks-and-reporting")` for Lamprianou-inspired network views, the FACETS/ConQuest reporting crosswalk and speaking/writing examples combining MFRM with G/D studies. `plot_data()` supports long-format extraction and original-value cluster profiles for custom distribution plots.

Start with the reporting question

- "Where should I start with an existing fit?" Use `mfrm_results()` to retain the fit and matching outputs, inspect `summary(res)`, then use `mfrm_report()` for a report or `export_mfrm_results()` for an analyst archive.
- "What can I give to an individual rater?" Use `mfrm_report(res, style = "rater", facet = "Rater", rater = "R01", output = "html")` for native additive RSM/PCM or experimental two-family GPCM MML results. Review and retain the standalone HTML file. The complete analysis archive serves the analyst and retains source data and identifiers.
- "Which parts of this run are ready to draft, and with what caveats?" Use `reporting_checklist()`.
- "How should I phrase the model, fit, and precision sections?" For `RSM` / `PCM`, use `build_apa_outputs()`.
- "Which tables should I hand off to a manuscript or appendix?" Use `build_summary_table_bundle()`, `export_summary_appendix()`, `apa_table()`, and `facet_statistics_report()`.
- "How do I explain model-based vs exploratory precision?" Use `precision_review_report()` and `summary(diagnose_mfrm(...))`.
- "Which caveats need to appear in the write-up?" Use `reporting_checklist()` first, then `build_apa_outputs()`.
- "How should I report candidate-model comparisons?" Use `compare_mfrm()` for the same-data comparison table, then `build_model_choice_review()` and `build_summary_table_bundle()` for cautious model-role, route-boundary, and wording tables.
- "How should I start figure captions or visual-results wording?" Use `visual_reporting_template()` for conservative caption and results sentence starters, then verify availability with `reporting_checklist()$visual_sc`

Editable one-family GPCM report

Copy the installed `templates/gpcm-report.Rmd` using `system.file("templates", "gpcm-report.Rmd", package = "mfrmr")`, then render it with `rmarkdown::render()`. The default synthetic example produces an HTML report, separation/Strata/reliability and fit tables, a Wright map, category-probability curves, ordinal Krippendorff's alpha, CSVs and an RDS analysis. Set `ratings_csv`, `rating_min`, `rating_max` and `output_dir` through the template's parameters for your data. It expects columns Person, Rater, Criterion and Score, with one rating per cell; repeated occasions require an explicit measurement unit and agreement keys. The model is one-family GPCM MML with criterion-owned steps and slopes. Alpha is calculated separately from observed ratings; `mfrm_report()` does not automatically include it. Fit review states and precision limitations remain visible. Complete the study-context and interpretation prompts before treating the rendered document as a manuscript. See `vignette("mfrmr-reporting-and-apa")` for the output map and full RSM lesson.

Recommended reporting route

1. Fit with `fit_mfrm()` and inspect `summary(fit)` before interpreting it.
2. Calculate the diagnostics or intervals needed for the question. For a fixed-rater RSM/PCM feedback sheet, use `diagnose_mfrm()` and, for eligible MML fits, `mfrm_facet_intervals()`. A rater coefficient and a difference between raters are distinct interval targets.
3. Retain them with `res <- mfrm_results(fit, diagnostics = diagnostics, intervals = list(raters = ci), compute = "never")` when these objects have been calculated. Omit

attachments you do not need. Here `compute = "never"` avoids filling absent diagnostics automatically; missing sections remain explicit.

4. Inspect `summary(res)$triage` and `summary(res)$next_actions`. Choose a figure through `summary(res)$plot_map` or `mfrmr_output_guide("plots")`. Use `plot(res, ...)` or `as_ggplot()` for supported views. An attached rater interval is displayed by `type = "facet_raters"`; an ordinary Wright map does not acquire that interval method automatically.
5. Use `mfrm_report(res)` for the analyst's quality-control report, or `style = "rater"` with explicit `facet` and `rater` for one recipient. `output = "html"` creates a temporary file; copy it to keep it.
6. Save `res` with `base::saveRDS()` for later use, or use `export_mfrm_results()` for CSVs, HTML, RDS and replay files. `preset = "starter"` also requests reports and available figures. Review `written_files` and `plot_errors` before treating export as complete.

These functions serve different purposes; figures and reports are optional branches from the saved results, not compulsory steps before saving them. The worked example in `vignette("mfrmr-facet-intervals")` follows this route through individual feedback, interval figures and save/reopen operations.

Specialist reporting tools

Use `reporting_checklist()` to inspect manuscript sections and `precision_review_report()` when the strength of precision claims matters. `build_apa_outputs()` supplies manuscript-draft prose and metadata; `build_summary_table_bundle()` collects selected tables for `apa_table()` or `export_summary_appendix()`. These narrower tools remain useful and are not replaced by the general report. For `GPCM`, keep the `gpcm_boundary` caveats with APA/QC/export output. When strict marginal rows are available, `plot(x, type = "marginal_fit")` and `plot(x, type = "marginal_pairwise")` support local-misfit follow-up. When candidate models are compared, keep the comparison as a reporting review: `compare_mfrm()` -> `build_model_choice_review()` -> `build_summary_table_bundle()`. Treat `GPCM` as a slope-aware sensitivity route unless the study design explicitly justifies discrimination-based operational scoring.

Keep each fit with its own diagnostics

After changing the data, model, estimator, or fitting settings, compute `diagnostics <- diagnose_mfrm(fit)` again for the new fit. Supplying the previous model's diagnostics can mix its standard errors and precision status with the new model's estimates. Fit summaries, precision reviews, APA and visual reporting, fit plots, and fit-level export helpers reject mismatched or outdated diagnostic readiness records. Matching saved diagnostics remain reusable; changing a table caption or note does not bypass this check. For saved analyses affected by the 0.2.4 calculation changes, follow "Updating saved analyses for 0.2.4" in mfrmr_workflow_methods before rebuilding reports. Matching the source fit alone does not update old diagnostic calculations.

Model-comparison reporting route

Use `compare_mfrm()` to build the candidate-model table and inspect `ICComparable`, `ComparisonBasis`, and any nesting warnings before reading information criteria. Automatic ranking requires the current contract and a shared q>=31 grid; q<31 retains raw screening/review criteria only, and a close decision still needs a prespecified denser common-grid check. Then use `build_model_choice_review()` to attach the comparison to explicit model roles, downstream-route boundaries, wording templates,

and optional build_weighting_review() output. Convert that review with build_summary_table_bundle() when a manuscript appendix, coauthor handoff, or HTML export needs stable table names.

A conservative GPCM reporting sequence is: fit_mfrm() for the equal-weighting RSM / PCM reference, fit_mfrm() for the GPCM sensitivity fit, compare_mfrm(), build_model_choice_review(), build_summary_table_bundle(), then export_summary_appendix() or export_mfrm_bundle(). Do not use AIC, BIC, or log-likelihood alone as an automatic operational-scoring decision.

Latent-regression reporting route

Active latent-regression fits expose their reporting surface through summary(fit)$population_overview, summary(fit)$population_coefficients, summary(fit)$population_coding, and fit-level caveats. Report those coefficients as conditional-normal population-model parameters, not as a post-hoc regression on EAP or MLE scores. Also report the population_formula, coding/contrast information, population_policy, and omitted-person or omitted-row counts when complete-case handling was used.

Prediction-side helpers predict_mfrm_units() and sample_mfrm_plausible_values() can carry the fitted population model into future-unit scoring and plausible-value draws. The supported route is one-dimensional MML for RSM / PCM; avoid stronger claims about multidimensional latent regression, Wald tests, posterior predictive checking, or full external-engine equivalence unless those checks were performed outside this helper family.

Publication-readiness boundary

mfrmr can provide a defensible measurement-output trail for a manuscript: fitted model summaries, diagnostic tables, precision review, report templates, APA table metadata, figure-routing guidance, and reproducible exports. It does not decide whether a specific journal claim is warranted. For high-stakes or selective journals, use the package outputs together with the study design, measurement rationale, primary citations, sensitivity checks, and substantive argument for the target field.

Treat DraftReady, ReadyForAPA, ClaimStrength, and report-template rows as drafting and caveat-routing aids. They are not formal acceptance rules, proof of validity, or a substitute for peer-review judgment. Before copying text, inspect mfrm_report(res, style = "apa")$first_screen, $claim_readiness, $report_gaps, and $template_index.

Standards basis and boundary

The manuscript helpers are APA-oriented drafting aids informed by the *Publication Manual of the American Psychological Association* (7th ed.) and the quantitative Journal Article Reporting Standards (JARS-Quant; Appelbaum et al., 2018). The MFRM-specific prompts also draw on the model and diagnostic sources returned by reporting_checklist(..., include_references = TRUE), including Eckes, Myford and Wolfe, Linacre, Wright and Masters, and Muraki for GPCM.

The package only knows the fitted measurement objects and context supplied by the analyst. It therefore cannot certify research-level JARS completeness for hypotheses and their confirmatory/exploratory status, recruitment and participant characteristics, ethics, sample-size rationale, missing-data mechanism and exclusions, multiplicity or deviations from plan, or complete data/code availability statements. Those study-level fields, effect-size and uncertainty choices, statistic-specific rounding, and journal typography must be reviewed and completed outside the generated template. The "Manuscript coverage map" in vignette("mfrmr-reporting-and-apa", package = "mfrmr") connects each reporting topic to the relevant output and the information the author must supply. It

includes the rating assignment and training, missingness, estimation settings, uncertainty, category functioning, and the distinct meanings of separation reliability and observed agreement. The vignette also demonstrates a question-to-result explanation with actual estimates. Its short Methods and Results example includes rating-row denominators, residual counts, step uncertainty, and the difference between separation-based reliability and posterior-variance EAP reliability. The observed-versus-fair-score example states its reference environment and distinguishes logit-measure uncertainty from fair-score uncertainty. Verify author-provided context labels, such as `scale_desc`, against the study record; generated-content checks do not establish their accuracy.

Appelbaum, M., Cooper, H., Kline, R. B., Mayo-Wilson, E., Nezu, A. M., and Rao, S. M. (2018). Journal article reporting standards for quantitative research in psychology: The APA Publications and Communications Board task force report. *American Psychologist, 73*(1), 3-25. doi:10.1037/amp0000191

Which helper answers which task

`reporting_checklist()` Turns current analysis objects into a prioritized revision guide with `DraftReady`, `Priority`, and `NextAction`. `DraftReady` means "ready to draft with the documented caveats"; `ReadyForAPA` is retained as a backward-compatible alias, and neither field means "formal inference is automatically justified". The `"Visual Displays"` rows also mirror the public plot family, so the checklist doubles as a figure-routing surface.

`build_apa_outputs()` Builds shared-contract prose, table notes, captions, and a section map from the current fit and diagnostics.

`build_summary_table_bundle()` Turns supported `summary()` outputs into named `data.frame` tables plus an index for manuscript or appendix handoff, and now also supports bundle-level `summary()` / `plot()` for role coverage and numeric QC.

`export_summary_appendix()` Writes those documented summary-table bundles to CSV and optional HTML appendix artifacts without requiring a full fit-based export bundle.

`apa_table()` Produces reproducible base-R tables with APA-oriented labels, notes, and captions.

`precision_review_report()` Summarizes whether precision claims are model-based, hybrid, or exploratory.

`facet_statistics_report()` Provides facet-level summaries that often feed result tables and appendix material.

`build_visual_summaries()` Prepares publication-oriented figure data that can be cited from the report text.

`visual_reporting_template()` Provides conservative figure placement, caption-starter, results-wording, and overclaim-avoidance guidance for public visual helpers.

Practical reporting rules

- Treat `reporting_checklist()` as the gap finder and `build_apa_outputs()` as the writing engine.
- Use the checklist's `"Visual Displays"` rows to decide whether the next follow-up should be plot(x, type = "qc"), plot(x, type = "marginal_fit"), plot(x, type = "residual_pca"), plot(x, type = "bias"), or another public plot.
- Use `visual_reporting_template()` to draft visual captions and results-sentence starters, but do not paste the skeletons without checking the actual fit, diagnostics, and study context.

- Before formal inferential claims, review the fit decision, precision checks, and restrictions for the particular statistic. A `model_based` tier alone does not establish that the claim is supported.
- Keep bias and differential-functioning outputs in screening language unless the current precision layer and linking evidence justify stronger claims.
- Treat `DraftReady` (and the legacy alias `ReadyForAPA`) as a drafting-readiness flag, not as a substitute for methodological review.
- Rebuild APA outputs after major model changes instead of editing old text by hand.
- For `GPCM`, use APA/QC/export helpers only as caveated sensitivity-reporting surfaces and keep full FACETS-style score-side review outside this route.

Fit-to-HTML reporting bundle

When the user already has a fitted object and wants a local report folder in one call, use `export_mfrm_bundle()` directly: `export_mfrm_bundle(fit, include = c("core_tables", "checklist", "dashboard", "apa", "summary_tables", "manifest", "script", "html"))`. This route computes missing diagnostics, writes CSV/text/replay artifacts, and creates a lightweight HTML summary without requiring a prior `mfrm_results()` object. Use `mfrm_results()` and `mfrm_report()` first when the goal is interactive triage or report-readiness review; use `export_mfrm_bundle()` when the goal is a file bundle for a project folder, coauthor handoff, or supplementary-methods archive. The bundle is not deidentified; review every file under the study's data-handling policy before any handoff.

Additional task-specific workflows

Use these when the reporting question calls for extra manuscript or appendix assembly, after choosing the supported route above.

- Manuscript-first route: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()` -> `build_apa_outputs()` -> `build_summary_table_bundle()` -> `summary()` / `plot()` -> `apa_table()`, `export_summary_appendix()`, or `export_mfrm_bundle()`(include = c("summary_tables", "html")). For `RSM` / `PCM` final reports, prefer `method = "MML"` and `diagnostic_mode = "both"` in the diagnostics step. For `GPCM`, use the same fit-based reporting/export family only as caveated sensitivity-reporting output and inspect its `gpcm_boundary` rows before writing claims.
- Appendix-first route: `facet_statistics_report()` -> `apa_table()` -> `build_visual_summaries()` -> `build_apa_outputs()`.
- Precision-sensitive route: `diagnose_mfrm()` -> `precision_review_report()` -> `reporting_checklist()` -> `build_apa_outputs()`.
- `GPCM` route: `diagnose_mfrm()` -> `precision_review_report()` -> `reporting_checklist()` -> direct residual/category/information helpers -> caveated `build_apa_outputs()`, `build_visual_summaries()`, `run_qc_pipeline()`, or `export_mfrm_bundle()` as needed.
- Model-comparison route: `compare_mfrm()` -> `build_model_choice_review()` -> `build_summary_table_bundle()` -> `export_summary_appendix()` or `export_mfrm_bundle()`(include = c("summary_tables", "html")).

Companion guides

- For report/table selection, see mfrmr_reports_and_tables.
- For end-to-end analysis routes, see mfrmr_workflow_methods.

- For visual follow-up, see mfrmr_visual_diagnostics.
- For the GPCM support statement, see gpcm_capability_matrix.
- For a longer walkthrough, see `vignette("mfrmr-reporting-and-apa", package = "mfrmr")`.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrmr_reporting_and_apa", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
toy <- load_mfrmr_data("example_operational")

# Fit the model
fit <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  model = "RSM"
)

# Check which report sections have supporting evidence
diagnostics <- diagnose_mfrm(fit)
checklist <- reporting_checklist(fit, diagnostics = diagnostics)
checklist$section_summary
subset(checklist$checklist, !DraftReady,
       c("Section", "Item", "NextAction"))

# Format the per-facet distribution summary as a table
results <- summary(fit, diagnostics = diagnostics)
tbl <- apa_table(results, which = "facet_overview")
tbl # Prints the table, caption, and note; review them before using in a paper

# For individual estimates, see as.data.frame(fit)

## End(Not run)
```

`mfrmr_reports_and_tables` *mfrmr Reports and Tables Map*

Description

Quick guide to choosing the right report or table helper in `mfrmr`. Use this page when you know the reporting question but have not yet decided which bundle, table, or reporting helper to call.

Recommended report route

For ordinary MML/JML `mfrm_fit` results, start here. The Examples section follows the same route with bundled data.

1. Inspect `summary(fit)$decision` before interpreting the estimates.
2. Collect `res <- mfrm_results(fit)`. By default it computes missing requested diagnostics. Pass existing matching checks through `diagnostics = diagnostics` to reuse them; `compute = "never"` leaves missing diagnostics unavailable. Saved intervals are explicit attachments.
3. Read `summary(res)$triage` and `$next_actions`; use `$plot_map` to find supported figures. A failed or unavailable component needs its own explanation, not an automatically substituted analysis.
4. Choose `mfrm_report(res)` for an analyst report, `plot(res, ...)` for a supported figure, or `export_mfrm_results()` for tables and an archive. Save the complete object with `saveRDS(res, file)` for later R use. Reopen with `readRDS(file)`; reading or displaying saved outputs does not refit the model. Reports and figures are optional, not required to save.

Native CML and comparison-only fits do not supply ordinary consolidated reports. Read `summary(fit)`; for CML use `mfrm_facet_intervals()`, then `summary(ci)`, `plot(ci)`, `apa_table()` and `saveRDS(ci, file)`. G/D-study and observed-agreement results also have their own summary and saving routes. For GPCM or extended measurement models, `mfrm_results()` states the supported sections and attachments; model-specific uncertainty limits remain in the collected results.

Individual helpers below answer additional questions. `build_apa_outputs()` drafts manuscript text; `build_summary_table_bundle()` collects selected tables for `export_summary_appendix()`. Neither is a prerequisite for `mfrm_report()`. Use `mfrmr_output_guide("beginner")` for a compact question-to-function map, or the full guide for specialist routes.

Choose what to save

- `export_mfrm_tables()`: core CSV tables from a fitted model. Supplied matching diagnostics enrich the tables; this call does not compute them.
- `export_mfrm_results()`: selected tables/reports and, by default, the complete results RDS plus a script that reloads it without refitting.
- `export_mfrm_bundle()`: fit-level tables, optional diagnostics and a refitting recipe. Its `"script"` component reruns the analysis; it does not preserve a fitted R object. Supply the original data for a colocated replay CSV, or replace the script's data-path placeholder yourself.
- `export_summary_appendix()`: selected summary tables for a manuscript. Display rounding follows `digits`; retain the source R object for reuse.

No export certifies estimates as usable or changes their uncertainty target. Inspect availability and caveat fields along with estimates. For G/D-study or native CML objects, use `saveRDS()` for the complete object and `write.csv()` for a selected supported summary table; the MFRM exporters do not widen their accepted model classes.

Start with the question

- "How can I read assessment-plan estimates and uncertainty together?" Use `summary.mfrm_multivariate_d_study()` with `intervals = TRUE` and `assumption = "normal"`, or supply saved intervals. Its `table`, `intervals` and `notes` retain methods, design and interpretation limits.

- "How should I document the model setup and run settings?" Use specifications_report().
- "Was data filtered, dropped, or mapped in unexpected ways?" Use data_quality_report() and describe_mfrm_data().
- "Did estimation converge cleanly and how formal is the precision layer?" Use estimation_iteration_report() and precision_review_report().
- "Which facets are measurable, variable, or weakly separated?" Use facet_statistics_report(), measurable_summary_table(), and facets_chisq_table().
- "Are score categories functioning in a usable sequence?" Use rating_scale_table(), category_structure_report() and category_curves_report().
- "Is the design linked well enough across subsets, forms, or waves?" Use subset_connectivity_report() and plot(x).
- "What should go into the manuscript text and tables?" For ordinary MML/JML RSM / PCM, use reporting_checklist(), build_apa_outputs(), and build_summary_table_bundle() or export_summary_appendix(). For GPCM, use the same route only where gpcm_capability_matrix() marks it as supported_with_caveat: direct table/plot helpers, summary-table appendix export, caveated build_apa_outputs(), and caveated export_mfrm_bundle() are available with a gpcm_boundary; score-side exports and design-forecasting evidence use their own caveated or blocked GPCM routes.
- "How can I estimate my model in Stan or brms?" export_mfrm_model() writes code, data, mappings and explicit Bayesian priors for supported ordinary MML specifications. Export does not run a sampler; posterior credible intervals are distinct from MML intervals. After sampling, use read_mfrm_posterior() to read the saved summary, inspect its sampling review, and continue with plot(), confint() and mfrm_report(). The summary does not contain joint posterior draws.
- "Did a simulation recover the known generating parameters well enough?" Use simulate_mfrm_recovery() for the recovery study, assess_mfrm_recovery() for the adequacy checklist, and then build_summary_table_bundle() or export_summary_appendix() for the appendix handoff.

Which output answers which question

specifications_report() Documents model type, estimation method, anchors, and core run settings. Best for method sections and reproducibility records.

data_quality_report() Summarizes retained and dropped rows, missingness, and unknown elements. Best for data cleaning narratives.

estimation_iteration_report() Shows replayed convergence trajectories. Best for diagnosing slow or unstable estimation.

precision_review_report() Summarizes whether SE, CI, and reliability indices are model-based, hybrid, or exploratory. Best for deciding how strongly to phrase inferential claims.

facet_statistics_report() Bundles facet summaries, precision summaries, and variability tests. Best for facet-level reporting.

subset_connectivity_report() Summarizes disconnected subsets and coverage bottlenecks. Best for linking and anchor strategy review.

rating_scale_table() Gives category counts, average measures, and threshold diagnostics. Best for first-pass category evaluation.

`category_structure_report()` Adds transition points and compact category warnings. Best for category-order interpretation.

`category_curves_report()` Returns category-probability, cumulative-probability, expected-ogive, total-information, and category-specific information coordinates. Best for downstream graphics and report drafts.

`write_mfrm_residual_file()` Writes an observation-level residual file, optionally with modeled category probabilities. Best for external case review or reproducible handoff.

`write_mfrm_subset_file()` Writes connected-subset summary and node-membership files. Best for scale-linking review outside R.

`reporting_checklist()` Turns analysis status into an action list with priorities and next steps. Best for closing reporting gaps.

`build_apa_outputs()` Creates manuscript-draft text, notes, captions, and section maps from a shared reporting contract.

`build_summary_table_bundle()` Converts supported `summary()` outputs into named `data.frame` tables with a compact index for appendix or manuscript handoff, including recovery simulation and recovery assessment outputs. It also supports bundle-level `summary()` / `plot()` for QC before export.

`export_summary_appendix()` Exports those documented summary-table bundles as CSV and optional HTML appendix artifacts without requiring the broader fit-based export bundle. This is the preferred export route for recovery simulation evidence.

`apa_table()` Can now take those summary-table bundles directly, so a selected component can move from `summary()` to a formatted handoff table without rebuilding the analysis object path.

Read availability and method limits separately

An available table or a passed numerical check does not establish that its intervals have accurate repeated-sample coverage. For implemented experimental methods, incomplete qualification means that a stated statistical property remains unestablished; inspect the specific limitation before interpreting it. Corrected JML retains this explanation in `interpretation_notes`, two-family GPCM in `gpcm_interpretation_notes`, and attached population intervals in `population_<name>_interpretation_notes`. These tables accompany the other result tables in reports and CSV exports; keep them with the estimates.

An `IntervalStatus` of `Incomplete` has a different meaning: only one endpoint was computed. It is neither an infinite bound nor a qualified one-sided interval. `Unavailable` means the requested result was not supplied; inspect its reason. `CIEligible`, where present, concerns that interval procedure's checks, not a universal coverage guarantee. Keep numerical status, method limitations and unavailable results when saving or sharing an analysis.

Practical interpretation rules

- Use bundle summaries first, then drill down into component tables.
- Use `precision_review_report()` to determine whether formal inference is supported for the fitted result.
- Treat category and bias outputs as complementary layers rather than substitutes for overall fit review.

- Treat zero-count score categories as scale-functioning caveats. Boundary zero-count categories can be retained with explicit `rating_min` / `rating_max`; retaining an intermediate zero-count category with `keep_original = TRUE` creates an unsupported adjacent-step contrast in a polytomous fitted ladder, so new fits stop before optimization. `summary(describe_mfrm_data(...))` exposes these in `Notes`, printed `Caveats`, and `$caveats`; `summary(fit)` carries full structured caveats into printed `Caveats` and `$caveats`, with `Key warnings` as a short triage subset. Summary-table exports use `score_category_caveats` and `analysis_caveats`.
- Use `reporting_checklist()` before `build_apa_outputs()` when a report still needs missing diagnostics or clearer caveats.

Additional task-specific workflows

These extend the basic route for eligible ordinary fits; they are not extra steps required for every report and do not apply to native CML.

- Run documentation: `fit_mfrm()` -> `specifications_report()` -> `data_quality_report()`.
- Precision and facet review: `diagnose_mfrm()` -> `precision_review_report()` -> `facet_statistics_report()`.
- Scale review: `rating_scale_table()` -> `category_structure_report()` -> `category_curves_report()`.
- Manuscript handoff (`RSM` / `PCM`): `reporting_checklist()` -> `build_apa_outputs()` -> `build_summary_table_bund` -> `summary()` / `plot()` -> `apa_table()` or `export_summary_appendix()` / `export_mfrm_bundle()`(include = "summary_tables").
- `GPCM` handoff: `reporting_checklist()` -> direct summaries/plots -> `build_apa_outputs()` or `build_summary_table_bundle()` -> `export_summary_appendix()` or caveated `export_mfrm_bundle()`, with `gpcm_boundary` retained in report/export objects.
- Recovery simulation handoff: `simulate_mfrm_recovery()` -> `plot()` / `assess_mfrm_recovery()` -> `build_summary_table_bundle()` -> `export_summary_appendix()`.

Companion guides

- For visual follow-up, see mfrmr_visual_diagnostics.
- For one-shot analysis routes, see mfrmr_workflow_methods.
- For manuscript assembly, see mfrmr_reporting_and_apa.
- For linking and DFF review, see mfrmr_linking_and_dff.
- For legacy-compatible wrappers and exports, see mfrmr_compatibility_layer.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrmr_reports_and_tables", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Fit the model
fit <- fit_mfrm(
  data = ratings,
```

```
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

summary(fit)$decision # Review the fit before interpreting it

# Build the comprehensive results and report
res <- mfrm_results(fit)
report <- mfrm_report(res)
summary(report, view = "reader")

# Find report sections that need attention
report$first_screen[, c("Area", "Status", "MainIssue", "NextAction")]

# Extract individual person, rater, and criterion estimates
estimates <- as.data.frame(fit)
head(subset(estimates, Facet == "Person")) # First six persons
subset(estimates, Facet == "Rater")       # All raters
subset(estimates, Facet == "Criterion")   # All criteria

# Keep the full result, including methods and unavailable-output reasons
file <- tempfile(fileext = ".rds") # Choose a lasting path for your analysis
saveRDS(res, file)
saved <- readRDS(file)
summary(saved)$triage # No refitting or repeated diagnostic calculation

## End(Not run)
```

`mfrmr_visual_diagnostics` *mfrmr Visual Diagnostics Map*

Description

Quick guide to choosing the right base-R diagnostic plot in `mfrmr`. Use this page when you know the analysis question but do not yet know which plotting helper or `plot()` method to call.

If you are preparing figures for a report, start with `reporting_checklist()` and inspect the `"Visual Displays"` rows first. Those rows now map directly onto the public plotting family covered on this page, so the checklist can act as a plot-readiness router rather than just a manuscript checklist.

This guide is primarily written for diagnostics-based `RSM` / `PCM` workflows. Ordinary one-family `GPCM` fits also use the residual-based diagnostics stack through `diagnose_mfrm()`, plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "agreement"), plot(x, type = "facets_chisq"), plot(x, type = "residual_pca"), and plot(x, type = "qc"), plus the posterior-scoring, design-weighted-information path via `compute_information()` / plot(x), and the Wright / pathway / CCC fit plots. Two `GPCM`-specific caveats apply when interpreting these residual-based screens:

- The free discrimination parameter means MnSq mean-square screens carry weaker invariance evidence than they do under RSM / PCM. Treat MnSq flags from GPCM as exploratory pointers to cells that merit closer inspection rather than as Rasch-style violations of strict invariance.
- FACETS-style fair averages are a Rasch-family measure-to-score transformation. Under GPCM the fair-average panel of plot(x, type = "qc") therefore renders with an explicit "unavailable" status, and the broader compatibility-export helpers stay outside the documented GPCM boundary.

Use gpcm_capability_matrix() for the formal per-helper boundary before choosing a GPCM follow-up plot route. Two-family GPCM and corrected JML use their separate mfrm_response_diagnostics() routes; the shared mfrm_fit class does not authorize the ordinary diagnostics or Wright/Pathway displays above.

Portable fixed-calibration score batches have a separate review route. Use summary(scores) and plot(scores) as described in mfrm_calibration_score_methods. Those displays review returned scores; they do not replace source-fit diagnostics or establish calibration fit.

Session plot defaults

Set options(mfrmr.plot_preset = "publication") to choose a session default for plotting functions that expose the common preset argument. The supported values are "standard", "publication", "compact" and "monochrome". Precedence is an explicit call argument, then the session option, then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of plot() for report bundles use the same option through .... Plots without a common preset argument, including extended-model plots with their own palette controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with as_ggplot() uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply preset explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with old <- options(mfrmr.plot_preset = "monochrome") followed by options(old). Use options(mfrmr.plot_preset = NULL) to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

Titles and compatibility

The development version adds title to plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise"), plot(x, type = "unexpected"), plot(x, type = "agreement"), plot(x, type = "facets_chisq"), plot(x, type = "bubble"), plot(x, type = "bias") and plot(x, type = "facet_dashboard"). Omission retains default title metadata; title = "Scoring patterns" replaces it; title = NULL or title = "" suppresses that metadata. Titles and explanatory notes are returned outside the figure by default; display controls below govern explicit drawing. The dashboard's S3 plot() method forwards these controls. Use exact argument names.

Legacy `main` remains supported without warnings, and `main = NULL` keeps its original meaning: use the default heading. Supplying both `main` and `title` is an error, even when they are equal or both `NULL`. Existing positional calls retain their argument order. This is a compatible new spelling, not removal or deprecation of `main`.

A title change does not alter the data, interval level, screening settings, reference lines, subtitles or interpretation notes. In stored plot payloads, an omitted heading is represented by an empty string. When hiding the heading of a review-only dashboard, retain its limitations in the figure caption or accompanying report. Use the plot guide to check ggplot support; accepting `title` does not add a converter for an unsupported plot.

Fitted plots and saved intervals

For ordinary fitted-model plots, `plot(fit, level = 0.90)` is an alternative to `ci_level = 0.90`; use only one spelling. The same control is forwarded by ordinary fit-based routes of `plot(res, ...)` and `as_ggplot(fit, ...)`. It changes the normal intervals where `show_ci = TRUE`, without refitting the model. The default remains 0.95. Specialist helpers that document only `ci_level` retain that argument; this is not a package-wide rename.

A saved inferential result has already chosen its method and level. For example, create `ci <- mfrm_facet_intervals(fit, "Rater", level = 0.90)` before `plot(ci)`; passing `level` to that plotting method is an error. The attached route `plot(res, type = "facet_raters")` uses those saved endpoints and does not inherit the Wright map's interval settings.

A saved plot payload also retains its rendering choices. With `p <- plot(fit, level = 0.90, show_title = FALSE, draw = FALSE)`, `as_ggplot(p)` keeps that selection. To change it, make a new plot from `fit`; extra settings such as `as_ggplot(p, level = 0.80)` are rejected. To change labels on the converted ggplot, use `as_ggplot(p) + ggplot2::labs(title = NULL)`.

Figures omit titles, subtitles and explanatory footnotes by default. Keep the returned object: `p <- plot(fit)`, then use `plot_data(p, "title")` and `plot_data(p, "notes")` to prepare the title and notes separately in a manuscript. Printing a returned `mfrm_plot_data` object also displays notes in the console. The same notes accessor works for mfrmr ggplots and after saving and reloading the returned object. Axis labels, legends, panel identifiers and data/status symbols remain. `title`, `main`, `subtitle` and `caption` now supply metadata by default. Entry points with `show_title` and `show_notes` allow explicit opt-in display; both switches default to `FALSE`. For ggplot customization, use `ggplot2::labs(title = "...")`. Numerical warnings and saved diagnostics remain available regardless of these presentation choices.

Read score direction and diagnostic terms first

A logit is a model coordinate, not a label for good or bad performance. Record the wording of both ends of the rubric before interpreting a plot. In the ordinary positive-discrimination models, a higher Person location predicts higher **coded scores**, holding other effects fixed:

Comprehensibility coding	Meaning of a higher Person location
1 = difficult to understand; 4 = easy to understand	Easier to understand
1 = easy to understand; 4 = difficult to understand	More difficult to understand

For a default negatively oriented Rater facet, a higher Rater location instead predicts **lower** coded scores. This is more severe under the first rubric, but more favorable under the second. A facet

declared in `positive_facets` has the opposite parameter orientation. That argument does not reverse the outcome categories or make a Person score mean "better". Relabeling categories or flipping a graphical axis also does not recode the fitted model. To reverse a numeric 1–4 rubric, recode as `5 - Score` before fitting, retaining the original data and consistent category meanings across criteria. Existing anchors must use the same coding and scale; changing coding is not just editing a figure.

The fitted-model plotting notes state recorded facet directions without guessing the rubric meaning. Use `p <- plot(fit, type = "wright")`, `print(p)` and `plot_data(p, "notes")`; keep the rubric wording with the analysis when saving it. The package does not infer that a column called Comprehensibility measures ease rather than difficulty. An older saved plot retains its original notes; regenerate the plot from its saved fit to obtain new guidance. An imported model requires its source orientation.

Overfit and underfit. Infit and Outfit mean squares compare squared residuals with the variation expected under the selected model. Their reference is 1: smaller values describe less residual variation (overfit), larger values more variation (underfit). Infit weights observations by information; Outfit is more sensitive to unexpected responses far from the person's fitted location. ZSTD uses a different, zero-centered reference. Displayed bands are review limits, not universal acceptability rules, and a tiny deviation from 1 is not automatically flagged.

In a speech contest, trained judges can have similar severity estimates or high agreement while their fit is near 1. Those features alone do not imply overfit. If low mean squares occur, review actual residuals, category use, shared performances and whether judgments were made independently. Training is one contextual explanation to investigate, not a demonstrated cause or a reason to dismiss the result. Joint discussion/copying can introduce dependence; consistent independent application of a rubric is a different process. Overfit is not the machine-learning meaning of fitting too many parameters. Neither overfit nor underfit alone identifies halo, unfairness, judge incompetence or a rule for deleting ratings.

Bias in a bias plot. `estimate_bias()` estimates an additional shift for a combination, such as Judge A scoring Content, after holding the baseline Person/facet/step/slope estimates fixed. It concerns a conditional interaction departure, not the judge's overall severity, statistical bias of an estimator, or proof of prejudice or unfairness. In mfrmr, positive `Bias Size` increases expected coded scores and negative values decrease them. Under a high-is-good rubric, positive bias thus favors that cell, whereas positive default Rater severity reduces its score. FACETS output can use different configured bias directions; compare definitions before comparing signs. The facet-profile plot averages **absolute** bias and cannot show which cells are favored. Read signed cell rows, exposure, uncertainty and the scoring context. Conditional intervals omit calibration uncertainty and the flags remain screening evidence.

Step. A Step is an adjacent-category transition parameter: a 1–4 rubric has transitions 1 to 2, 2 to 3 and 3 to 4. In the default unit-slope RSM, $\log\{P(Y = k)/P(Y = k - 1)\} = \theta - \delta - \beta - \tau_k$. At $\theta = \delta + \beta + \tau_k$, the two adjacent categories have equal probability. Other categories may also occur, so neither probability need be 0.5. A Step is not a learning stage, the mean Person location observed in that category, a cumulative 50-percent threshold, or a pass/fail cut. Stored τ_k offsets differ from full ability-axis crossings when facet effects are nonzero. The ordinary CCC/Wright step reference fixes other additive effects at zero; plot.mfrm_fit documents that profile. RSM shares steps; PCM/GPCM assigns them to a selected owner. Check the owner's counts and curves before interpreting disorder or merging categories. See `rating_scale_table()` and mfrmr_methods_reference for calculations and the Andrich/Masters references.

These distinctions follow the official FACETS facet orientation, fit report and bias report descrip-

tions, and the Winsteps adjacent-category threshold explanation. mfrmr's specific sign, reference-profile and uncertainty rules remain those documented in its own function help.

Start with the question

- "Do persons and facet levels overlap on the same logit scale?" Use `plot(fit, type = "wright")` or plot(x, type = "wright", view = "unified").
- "Where do score categories transition across theta?" Use `plot(fit, type = "pathway")` and `plot(fit, type = "ccc")`.
- "How do measures and observed scores relate to Fair Scores?" Use plot(x, type = "fair_average") with `view = "measure"`, `"scatter"`, or `"difference"`. FairZ is a zero-reference expected score, not a z-score.
- "Is the design linked well enough across subsets or administrations?" Use `plot(subset_connectivity_report(...), type = "design_matrix")`, `mfrm_network_analysis()`, `build_mfrm_network_review()`, `plot(..., type = "network")`, and plot(x).
- "Which responses or levels look locally problematic?" Use plot(x, type = "unexpected") and plot(x, type = "displacement").
- "Which facet/category cells drive strict marginal misfit?" Use plot(x, type = "marginal_fit").
- "Which level pairs drive strict local-dependence follow-up?" Use plot(x, type = "marginal_pairwise").
- "Do raters agree within observed shared scoring contexts?" Use plot(x, type = "agreement") and `rater_network_analysis()`.
- "Do facet levels separate meaningfully?" Use plot(x, type = "facets_chisq").
- "Do criteria within the same rater move together in a halo-like way?" Use `rater_halo_network_analysis()` and `plot(..., type = "edge_distribution")`.
- "Is there notable residual structure after the main Rasch dimension?" Use plot(x, type = "residual_pca").
- "Which interaction cells or facet levels drive bias screening results?" Use plot(x, type = "bias").
- "Which group-by-facet contrasts drive DFF / DIF screening results?" Use plot(x, type = "heatmap") and plot(x, type = "summary") after `analyze_dff()`.
- "Do person response rows follow the expected Guttman-style ordering once persons and items are sorted on the logit scale?" Use plot(x, type = "guttman") as a teaching-oriented screen.
- "Do person-level standardized residuals look Gaussian, or are there heavy tails that warrant follow-up?" Use plot(x, type = "residual_qq").
- "Is rater severity drifting across waves or training sessions (assuming the waves are already on a common anchored scale)?" Use `plot(fits[[1L]], type = "trajectory", comparison = fits[-1L], labels = names(fits))` (see `plot.mfrm_fit()`) together with plot(x) for the linking-scale review.
- "I have many raters and want a compact pairwise agreement / correlation overview instead of the bar chart?" Use plot(x, type = "agreement", view = "heatmap").
- "Do response times suggest rapid responding, slow responding, or timing patterns by person, facet, or score category?" Use `response_time_review()` and plot(x) as a descriptive QC layer outside the MFRM likelihood.

- "Are there pairs of facet levels whose residuals co-move beyond the main-effects MFRM? (Q3-style local-dependence screen)" Use plot(x, type = "local_dependence").
- "How distinguishable is each facet on a single page (separation, strata, reliability)?" Use plot(x, type = "reliability").
- "Where do persons with the largest residual aggregates accumulate across facet levels?" Use plot(x, type = "residual_matrix").
- "How much did empirical-Bayes shrinkage move each facet level?" Use plot(x, type = "shrinkage", view = "funnel") on a fit augmented via apply_empirical_bayes_shrinkage().
- "I need one compact triage screen first." Use plot(x, type = "qc") for RSM / PCM. The GPCM branch can also call plot(x, type = "qc"), but its fair-average panel reports an explicit unavailability indicator because that panel's score-metric semantics are limited to the Rasch-family branch.
- "I scored new Persons from a frozen portable calibration; which scores require review and how wide are their conditional intervals?" Use summary(scores) followed by plot(scores, type = "interval"). Continue with type = "precision" or type = "edge_mass" when response-count precision or quadrature-edge behavior needs follow-up.
- "Which figures are already supported by my current run?" Use reporting_checklist() and review the "Visual Displays" rows before choosing the next plot.
- "Where should this figure go in a paper or appendix?" Use visual_reporting_template() for a static reporting-use table, then cross-check run-specific availability with reporting_checklist()$visual_scope
- "Do I need a 3D-style category probability surface?" Use plot(fit, type = "ccc_surface", draw = FALSE) to get theta-by-category-by-probability plot data for exploratory teaching or downstream interactive rendering. Keep 2D pathway/CCC plots as the default reporting figures.

Keep the network questions separate

mfrm_network_analysis() describes the assignment/co-observation graph and is the relevant network route for disconnected measurement subsets. rater_network_analysis() describes pairwise score agreement, disagreement, or severity direction, while rater_halo_network_analysis() describes rater-by-criterion score-profile correlations. The latter two do not establish assignment connectedness or a common scale. In all three, degree, strength, betweenness, and closeness are graph-theoretic quantities; they are not rating-scale central tendency, MFRM severity logits, or causal halo evidence.

Recommended visual route

1. If you are drafting a report, run reporting_checklist() first and read the "Visual Displays" rows as the plot-readiness layer.
2. Start with plot(x, type = "qc") for one-page triage.
3. Move to plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise"), and plot(x, type = "agreement") for flagged local issues.
4. Use plot(fit, type = "wright"), plot(fit, type = "pathway"), and plot(fit, type = "wright", view = "unified") for structural interpretation.

5. Use plot(x, type = "bias"), plot(x, type = "heatmap"), plot(x, type = "summary"), plot(x), and plot(x) when the checklist or dashboard points to interaction, differential-functioning, linking, or precision follow-up.
6. Use `plot(..., draw = FALSE)` when you want reusable plot data instead of immediate graphics.
7. For an artifact-scored batch, use `summary(scores)` and `plot(scores)`; keep its conditional-uncertainty note with the figure and inspect every `scored_review` or `not_scored` disposition.
8. Use `plot(fit, type = "ccc_surface", draw = FALSE)` only when you need 3D-ready category-probability data; `mfrmr` intentionally does not add a package-native plotly/rgl renderer for this route.
9. Use `preset = "publication"` when you want the package's cleaner manuscript-oriented styling, or `preset = "monochrome"` when journals, accessibility requirements, or print workflows require grayscale output.

Customizing figures

The package's plotting defaults are intended to be safe starting points, not a closed graphics system. Use `preset = "publication"` for clean manuscript defaults, or `preset = "monochrome"` for grayscale output that relies more on line type, point shape, and reference lines than on color. Use `plot(..., draw = FALSE)` when you want the reusable `mfrm_plot_data` object instead of immediate base graphics. Then call `plot_data_components()` to see available components and `plot_data()` to extract the long tables used by the plot.

Custom renderers should keep the returned metadata close to the figure: `reference_lines`, `legend`, `guidance`, `category_support`, `interpretation_guide`, and any reporting-template rows are part of the interpretation contract. They let users change colors, labels, panels, or rendering technology without losing the measurement scale, caveats, and caption boundary attached to the package-native plot.

Visual coverage

The plotting layer supports the following public follow-up roles when the fitted object contains the required data:

- First-pass triage: plot(x, type = "qc") or the `"Visual Displays"` rows from `reporting_checklist()`.
- Structural interpretation: `plot(fit, type = "wright")`, `plot(fit, type = "pathway")`, `plot(fit, type = "ccc")`, and plot(x, type = "residual_pca").
- Local issue follow-up: plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "agreement"), plot(x, type = "bias"), plot(x, type = "heatmap"), and plot(x, type = "summary").
- Strict marginal follow-up: plot(x, type = "marginal_fit") and plot(x, type = "marginal_pairwise") for `diagnostic_mode = "both"`.
- Reporting/export handoff: `build_visual_summaries()` and `draw = FALSE` routes that return reusable `mfrm_plot_data` objects for downstream review and export. When step estimates are available, `build_visual_summaries()` also exposes `$plot_payloads$category_probability_surface`.
- Portable score-batch review: `plot(scores, type = "interval")`, `"precision"`, or `"edge_mass"` after `score_mfrm_calibration()`. These views retain selection accounting, review dispositions, and Persons that could not be scored.

- 3D-ready exploratory handoff: `plot(fit, type = "ccc_surface", draw = FALSE)` returns a theta-by-category-by-probability `mfrm_plot_data` object. This is not a default APA/reporting figure and does not load plotly/rgl.

3D and surface data

The package currently treats 3D as an exploratory data handoff, not as a default plotting layer. The supported route is `plot(fit, type = "ccc_surface", draw = FALSE)`, which returns `surface`, `categories`, `category_support`, `groups`, `axis_contract`, `renderer_contract`, `interpretation_guide`, and `reporting_policy` tables inside an `mfrm_plot_data` object. These columns can be passed to an external renderer if needed, while `category_support` and `interpretation_guide` should be checked before interpreting retained zero-frequency categories or adjacent threshold ridges.

Do not replace the standard 2D Wright map, pathway map, CCC plot, heatmap/profile diagnostics, or information curves with 3D figures in routine reports. In particular, 3D Wright maps are discouraged because perspective and occlusion obscure the shared-scale comparison that the Wright map is meant to support.

Which plot answers which question

`plot(fit, type = "wright")` Shared logit map of persons, facet levels, and step thresholds. Best for targeting and spread.

`plot(fit, type = "pathway")` Expected score by theta, with dominant-category strips. Best for scale progression.

`plot(fit, type = "ccc")` Category probability curves. Best for checking whether categories peak in sequence.

`plot()` Observation-level surprises. Best for case review and local misfit triage.

`plot()` Level-wise anchor movement. Best for anchor robustness and residual calibration tension.

`plot()` Posterior-integrated first-order category residuals. Best for seeing which facet/category cells drive strict marginal flags.

`plot()` Posterior-integrated exact/adjacent agreement residuals. Best for exploratory local-dependence follow-up after strict marginal flags.

`plot()` Exact agreement, expected agreement, pairwise correlation, and agreement gaps. Best for rater consistency.

`plot()` Facet variability and chi-square summaries. Best for checking whether a facet contributes meaningful spread.

`plot()` Residual structure after the Rasch dimension is removed. Best for exploratory residual-structure review, not as a standalone unidimensionality test.

`plot()` Interaction-bias screening views for cells and facet profiles. Best for systematic departure from the additive main-effects model.

`plot()` / `plot()` DFF / DIF screening views for facet-level x group contrasts. Best for showing which facet and group pair is involved before writing substantive interpretations.

`plot()` Anchor drift and screened linking-chain visuals. Best for multi-form or multi-wave linking review after checking retained common-element support.

`plot()` Person x facet-level response matrix with unexpected-response overlay. Best for teaching-oriented scalogram intuition and visual triage of where the data depart from the expected ordering.

`plot()` Normal Q-Q plot of person-level standardized residual aggregates. Best for checking the tail behavior of residuals as exploratory follow-up after a fit screen.

`plot()` Per-rater severity trajectory across named waves / occasions. Best for rater-training or drift feedback when the supplied fits have already been placed on a common anchored scale; the helper itself does not perform linking.

`plot()` Compact pairwise rater x rater heatmap of exact agreement (default) or Pearson-style correlation. Best when the rater count makes the bar-chart form of plot(x, type = "agreement") too busy.

`response_time_review()` / `plot()` Descriptive response-time screening by person, facet, and score category. Best for reviewing rapid/slow response patterns alongside MFRM diagnostics; it is not a joint speed-accuracy model and does not change fitted measures.

`plot()` Standardized, aggregated-residual Q3-style heatmap of pairwise residual correlations between facet levels. Best for exploratory local-dependence screening; pairs with very strong off-diagonal residual correlation merit content-level review.

`plot()` One-figure facet x reliability / separation / strata bar overview built from `diagnostics$reliability`. Best as a single small figure for "which facets are statistically distinguishable?".

`plot()` Person x facet-level standardized residual heatmap. Best as a follow-up to plot(x, type = "guttman") when the residual sign and magnitude matter, not just the response code.

`plot()` Empirical-Bayes shrinkage caterpillar / funnel showing raw versus shrunken facet estimates. Best on fits produced via `apply_empirical_bayes_shrinkage()` for reviewing how much each level moved under the prior.

`plot(scores, type = "interval")` Posterior EAP intervals for a portable fixed-calibration score batch, with review dispositions highlighted. The intervals are conditional on the frozen point calibration and exclude calibration-parameter uncertainty.

Cross-reference to FACETS / Winsteps tables

For users coming from the standard Rasch-measurement software packages, the closest mfrmr helper for each table or figure family is summarised below. The mapping is approximate; mfrmr is designed for many-facet workflows, so column subsets and column names differ.

Wright (variable) map `plot(fit, type = "wright")` and plot(x, type = "wright", view = "unified") correspond to FACETS Table 6 / Winsteps "Person-Item map".

Pathway / probability curves `plot(fit, type = "pathway")` and `plot(fit, type = "ccc")` correspond to Winsteps Table 21 ("Probability category curves") and FACETS category-probability curves.

Test / item information `compute_information()` + plot(x) correspond to Winsteps Table 17 ("Test characteristic curve, test information function").

Misfit / Infit / Outfit `diagnose_mfrm()` and the Largest |ZSTD| / MnSq misfit blocks of `summary(diag)` correspond to Winsteps Table 10/13/14 (Misfit order) and FACETS Tables 7/8.

Bias / interaction `estimate_bias()` + plot(x, type = "bias") correspond to FACETS Table 14 ("Bias / Interaction calibration report").

Differential rater / item functioning `analyze_dff()` / `analyze_dif()` + plot(x, type = "heatmap") / plot(x, type = "summary") cover the FACETS DIF / bias-by-group route and the Winsteps DIF (Table 30 group differences) report.

Inter-rater agreement interrater_agreement_table() + plot(x, type = "agreement") / plot(x, type = "agreement", view = "heatmap") correspond to FACETS Table 7-style observed-vs-expected agreement reports.

Anchoring / linking plot(x) and plot(x) cover the FACETS / Winsteps anchored-run review route; full equating-chain helpers are exposed via build_equating_chain().

Practical interpretation rules

- Wright map: look for gaps between person density and facet/step locations; large gaps indicate weaker targeting.
- Pathway / CCC: look for monotone progression and clear category dominance bands; flat or overlapping curves suggest weak category separation.
- 3D-ready category surface: use as an exploratory view of the same category-probability information, not as a replacement for the 2D pathway/CCC figures in reports. Read category_support first when a retained category has zero observed responses.
- Unexpected / displacement: use as screening tools, not final evidence by themselves.
- Strict marginal and pairwise local-dependence plots are exploratory follow-up layers for diagnostic_mode = "both", not standalone inferential tests.
- Inter-rater agreement and facet variability address different questions: agreement concerns scoring consistency, whereas variability concerns whether facet elements are statistically distinguishable.
- Residual PCA and bias plots should be interpreted as follow-up layers after the main fit screen, not as first-pass diagnostics.
- Portable score plots review returned score batches only. Inspect the source calibration separately, retain not_scored Persons in the audit trail, and do not interpret conditional intervals as including calibration-parameter uncertainty.
- DFF residual- and refit-method plots are screening visuals. Current refit rows do not receive ETS A/B/C labels because their plug-in uncertainty omits baseline-anchor uncertainty and cross-refit covariance.

Typical workflow

- Figure-readiness route: fit_mfrm() -> diagnose_mfrm() -> reporting_checklist() -> inspect "Visual Displays" rows -> chosen public plot helper.
- Quick screening: fit_mfrm() -> diagnose_mfrm() -> plot(x, type = "qc").
- Strict marginal follow-up: diagnose_mfrm() with diagnostic_mode = "both" -> plot(x, type = "marginal_fit") -> plot(x, type = "marginal_pairwise").
- Scale and targeting review: plot(fit, type = "wright") -> plot(fit, type = "pathway") -> plot(fit, type = "ccc").
- Linking review: subset_connectivity_report() -> plot(..., type = "design_matrix") / mfrm_network_analysis() / build_mfrm_network_review() / plot(..., type = "network") -> plot(x).
- Interaction review: estimate_bias() -> plot(x, type = "bias") -> reporting_checklist().
- DFF / DIF review: analyze_dff() -> plot(x, type = "heatmap") / plot(x, type = "summary") -> inspect the explicit facet, level, and group-pair columns before writing interpretation.

Figures with separate titles and interpretation

Wright, expected-score pathway, CCC, and plot(x, type = "fair_average") displays accept `show_title = FALSE` and `show_notes = FALSE`. These options hide figure annotations while retaining axes, legends, structural panel labels, and the original title and notes in the returned `mfrm_plot_data` object. With `p <- plot(fit, type = "ccc", draw = FALSE, show_title = FALSE, show_notes = FALSE)`, inspect `p$data$notes` before writing a caption. `as_ggplot()` respects the display options and retains notes in `attr(as_ggplot(p), "mfrmr_notes")`. Hiding annotations does not change readiness or interval eligibility.

Use `preset = "monochrome"` for grayscale. Expected-score and CCC curves have six line types: solid, dashed, dotted, dotdash, longdash, and twodash. Fair-score plots have six point shapes. Encodings repeat beyond six series; use panels or select a facet instead of relying on gray levels alone. These controls are documented for the named routes, not every plotting helper in the package. Check the individual help page for other plots.

Fair-score intervals remain diagnostic-only with full-refit coverage unverified. RSM/PCM plots propagate the focal measure SE while holding thresholds and reference measures fixed. GPCM-MML plots can use structural table SEs for non-Person rows, conditioning on Person EAP/reference means. `fair_average_table(fair_se = TRUE)` does not provide RSM/PCM fair-score SEs. See `mfrmr_interval_guide()` and plot(x, type = "fair_average") for the status columns and the observed-minus-fair gap caveat.

Non-Latin and long labels

Select an installed font that contains the characters in your labels before drawing. For example, on macOS, open a PNG device and set `par(family = "Hiragino Sans")` before `plot(fit)`. Use an appropriate installed font on other systems. A successful plot call alone does not confirm that the device rendered every character; inspect the saved figure. Long labels also require enough space. Use a wider device for rater-severity profiles. For crowded FACETS-style maps, use a taller or wider device, increase `rows_per_logit`, or switch to `renderer = "native"`. Full labels remain in the tables returned with `draw = FALSE`.

Companion vignette

For a longer, plot-first walkthrough, run `vignette("mfrmr-visual-diagnostics", package = "mfrmr")`.

See Also

mfrmr_workflow_methods, mfrmr_reports_and_tables, mfrmr_reporting_and_apa, mfrmr_linking_and_dff, gpcm_capability_matrix, `visual_reporting_template()`, `mfrmr_interval_guide()`, plot(x, type = "guttman"), plot(x, type = "qc"), plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "marginal_fit"), plot(x, type = "marginal_pairwise"), plot(x, type = "agreement"), plot(x, type = "facets_chisq"), plot(x, type = "residual_pca"), plot(x, type = "bias"), plot(x, type = "heatmap"), plot(x, type = "summary"), plot(x), `plot.mfrm_fit()`, plot(x, type = "residual_qq"), `plot(fits[[1L]], type = "trajectory", comparison = fits[-1L], labels = names(fits))` (see `plot.mfrm_fit()`), plot(x, type = "agreement", view = "heatmap"), `response_time_review()`, plot(x)

mfrmr_plot_migration

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrmr_visual_diagnostics", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
toy <- load_mfrmr_data("example_operational")

# Fit the model
fit <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  model = "RSM"
)

# Start with the shared-scale view; run plot commands one at a time
plot(fit)

# Compute diagnostics once and check their interpretation status
diagnostics <- diagnose_mfrm(fit)
review <- summary(diagnostics)
review$decision

# Inspect an overview of model-fit issues
plot(fit, type = "qc", diagnostics = diagnostics)

# Follow up on observed versus model-expected category frequencies
plot(diagnostics, type = "marginal_fit")

## End(Not run)
```

---

mfrmr_workflow_methods

mfrmr Workflow and Method Map

Description

Start with one analysis: load ratings, fit a model, plot the results, and read the summary. The Examples section contains a complete runnable script. The later sections describe diagnostics, reporting, and specialist routes.

See `vignette("mfrmr-networks-and-reporting")` for Lamprianou-inspired network views, the FACETS/ConQuest reporting crosswalk and speaking/writing examples combining MFRM with G/D studies. `plot_data()` supports long-format extraction and original-value cluster profiles for custom distribution plots.

Connect the question to the result

These are alternative or complementary branches, not requirements to run every function. Start from the intended score, rating unit, rubric endpoints and the decision the study needs to support.

Question	Connected route
What was rated, and can the groups be linked?	describe_mfrm_data() and subset_connectivity_report() with
How do persons and named facets differ?	fit_mfrm() -> diagnose_mfrm() and eligible intervals -> mfrm_resu
How many raters/tasks does the reported score need?	mfrm_g_study() -> mfrm_d_study() -> summary(), plot() and elig
Do raters agree?	Matched units -> pairwise_kappa_table() for pairs, krippendorff
Can a scale be reused?	review_mfrm_anchors() -> anchored fit / build_linking_review()
Does a combination or group need follow-up?	Specified estimate_bias() or analyze_dff() -> signed tables and p
Does model choice change the answer?	Comparable fits -> compare_mfrm() -> build_model_choice_revie
What remains unexplained?	analyze_residual_pca() / q3_statistic() for residual structure; s

A Criterion facet in one-dimensional MFRM adjusts criterion locations and, when specified, steps/slopes; it does not create separate Person dimensions. Several `scores` in `mfrm_g_study()` are fixed score components with source-specific covariance matrices, not additional random facets. Read `vignette("mfrmr-gtheory")` for the criterion/facet distinction and the report structure. Pass G/D objects directly to `mfrm_report()` and `build_summary_table_bundle()`; supplied saved intervals retain source context. The editable `templates/gtheory-report.Rmd` joins tables, plots and exports.

Higher coded scores need not mean better performance. For Comprehensibility, record whether they mean greater ease or greater difficulty. `positive_facets` changes parameter orientation, not outcome coding. Keep coding consistent with anchors and report labels; see mfrmr_visual_diagnostics. The `vignette("mfrmr-networks-and-reporting")` supplies richer network and custom distribution examples, plus literature-based reporting connections. The workflow vignette places assignment networks at design review and score networks alongside observed agreement. Neither is a residual graph.

The question-to-report map in `vignette("mfrmr-reporting-and-apa")` identifies which sections exist, which remain standalone, and what to report together.

Choose a reading path

To choose a model and estimator before fitting, read `vignette("mfrmr-model-choice", package = "mfrmr")` or the online decision guide. It connects assessment questions to the available routes and explains why starting with the most complex model is not automatically preferable.

For a first analysis, read **Start here**, run the Examples, then read **From the first summary to diagnostics**. You do not need to work through every specialist route. For your own file, use **Use your own ratings** and the workflow vignette's CSV lesson. For an unfamiliar result or formula, open mfrmr_methods_reference: it links questions to calculations, their limits and original references. Use `mfrmr_interval_guide()` when the question specifically concerns uncertainty.

Choose by purpose and design

For named rater/task severity comparisons, consider fixed-facet RSM/PCM; CML avoids an ability-distribution assumption when structural comparisons are the target, but estimates neither Person

scores nor a population SD. MML is the route for supported population and GPCM structural inference, with explicit population assumptions. JML is useful for joint points and matched FACETS comparisons; fixed-exposure bias limits structural inference. Corrected JML is an experimental shared-owner GPCM route, not a general RSM/PCM correction or a guarantee of true-parameter interval coverage.

Use PCM when separate category-step patterns answer the question; use GPCM when discrimination differences matter and the design can distinguish them. Free slopes do not automatically model dependence. Known Person-specific block dependence or a shared sampled-rater population can instead justify `fit_mfrm_testlet()` or `fit_mfrm_random_rater()` from the outset. These extensions have different targets and supported outputs. A larger model can contain a simpler one without estimating its restrictions exactly in finite samples; extra effects require information and can add uncertainty. Choose the structure justified by the question and design, not complexity alone. See mfrmr_methods_reference for assumptions and `mfrmr_interval_guide()` for uncertainty availability.

Start here

1. Load the package with `library(mfrmr)` and example ratings with `ratings <- load_mfrmr_data("example_operati`
2. Fit with `fit_mfrm()`. The `person`, `facets`, and `score` arguments name columns in the data; each row represents one rating event.
3. Draw the Wright map with `plot(fit)` to view person abilities, rater severities, criterion difficulties, and category thresholds together.
4. Save `results <- summary(fit)` and inspect `results$person_overview` and `results$facet_overview` for the distributions of estimates.

These overview tables summarize distributions. Use `as.data.frame(fit)` for individual person, rater, and criterion estimates, identified by `Facet` and `Level`. Before interpreting or reporting estimates, read `results$decision` and follow its `NextAction`; the default summary does not compute diagnostics. For your own data, first check the rating design and score categories with `describe_mfrm_data()`. `head(ratings)` shows the input rows; `Study` and `Group` are extra labels unused by this model. `<-` saves an object and `$` selects a named part of it. `ratings` is the name used in this example, not a required object that you must prepare beforehand. The loading line creates it from packaged synthetic data. Use that line first; after the example works, replace it with your own data import and review the column names and score categories.

From the first summary to diagnostics

The ordinary workflow below uses MML/JML `mfrm_fit` results. Native `fit_mfrm(method = "CML", category_policy = "preserve")` instead returns a dedicated RSM/PCM structural-parameter result: use `summary(fit)$locations`, `summary(fit)$steps` and `mfrm_facet_intervals()`, then `plot(ci)` and RDS saving. CML conditions out Person abilities and estimates no population SD; ordinary scoring, diagnostics and consolidated reports do not apply. See `vignette("mfrmr-facet-intervals")` for its executable comparison example. For a deficient school/rater assignment, declare CML comparisons with `describe_mfrm_data()` and use `fit_scope = "comparisons"`; `summary(fit)` then reports only requested contrasts, with conditional-information intervals for identifiable targets and missing values for others. See `vignette("mfrmr-linking-and-dff")` for that separate workflow.

A `FormalInference = "No"` entry in `results$decision` can mean that precision has not yet been reviewed. Read `Why` and `NextAction` to distinguish that state from a detected problem. Run

`diagnostics <- diagnose_mfrm(fit)` and inspect `summary(diagnostics)$decision`. To reuse these checks in the fuller reporting object, call `res <- mfrm_results(fit, diagnostics = diagnostics)`. Here `results` is the basic summary and `res` is the comprehensive object accepted by `mfrm_report()` and `export_mfrm_results()`.

Understand the function names

Learn the operation first, then open its help page:

- `fit_` estimates model parameters, for example `fit_mfrm()`.
- `score_` estimates Person abilities using a fitted calibration, for example `score_mfrm_persons()` for people already in a supported fitted RSM.
- `pool_` combines eligible analyses, for example `pool_mfrm_imputed()`.
- `review_` checks supplied data or analysis choices, for example `review_mfrm_imputations()`; it does not generate missing scores.
- `export_` writes files, for example `export_mfrm_results()`.
- `summary(object)` and `plot(object)` select a method for the object you supply. You normally do not call `plot.mfrm_testlet()` directly.

The package name is **mfrmr**; **mfrm** in analysis functions refers to a many-facet Rasch model. Some help guides use the package prefix `mfrmr_`. The prefixes do not select different estimators. For a short ordinary-MFRM map, use `mfrmr_output_guide("beginner")[, c("Question", "MainFunction")]`. The no-argument guide lists all specialist routes and is not a first lesson. For rater feedback, use `mfrmr_output_guide("feedback")`: it distinguishes uncertainty in severity from unexpected rating patterns and the accuracy of a warning rule. `mfrm_facet_intervals()` concerns specified fixed raters; shared-rater intervals have a different target and retain their separate limitations. Ordinary and extended-model residuals also use different definitions. The guide names the matching summary, plot and saving routes.

`review_mfrm_imputations()` checks completed data supplied by you; it does not generate missing scores. Follow it with `fit_mfrm_imputed()` and then `pool_mfrm_imputed()`. `mfrm_screening_performance()` evaluates a warning rule against known simulation truth. `mfrm_screening_sensitivity()` repeats that evaluation across specified thresholds; it does not establish accuracy from real ratings alone. `mfrm_pca()` summarizes numeric external attributes; `mfrm_cluster_kmeans()` forms groups. PCA is optional before k-means. `mfrm_cluster_pam()` uses partitioning around medoids (PAM) for mixed attributes; it does not automatically choose a clustering algorithm. For a dendrogram, use `mfrm_cluster_hierarchical()`. A G-study (`mfrm_g_study()`) estimates sources of variation in observed scores; a D-study (`mfrm_d_study()`) uses them to compare future rater/task plans. Neither needs an MFRM fit.

Updating earlier scripts

Earlier call	Recommended call	Meaning re
`mfrm_cluster(x, k)`	`mfrm_cluster_pam(x, k)`	Same Gowe
`mfrm_response_imputations(..., impute = ids)`	`review_mfrm_imputations(..., impute_ids = ids)`	Same check
`keep_original = TRUE`	`category_policy = "preserve"`	Same categ
`keep_original = FALSE`	`category_policy = "collapse"`	Same colla

The old calls **in this table** remain supported, including positional arguments. Saved objects retain their classes and `summary()`/`plot()` methods. Do not pass conflicting old/new category choices. See `compatibility_alias_table()` for the complete migration map. This does not change what `predict()` returns: use the ability-scoring or response-prediction route named for your model. The former standalone drawing `plot_*()` functions have a different policy: they are no longer exported in 0.2.4. Use result-object `plot()` and mfrmr_plot_migration to update those calls.

A **calibration** is the fitted set of model parameters, such as rater severity and category thresholds. A **conditional** ability interval holds them fixed; it excludes uncertainty from estimating the calibration. **EAP** is the mean of the conditional ability distribution. An **SE** describes uncertainty in an estimate; an **SD** describes spread. Check whether an SD refers to differences among persons or one Person's posterior.

Check defaults before adapting an example

An omitted argument selects a convention, not the best choice for your assessment. Start with the choices that change the analysis:

- **Model and population:** `fit_mfrm()` defaults to RSM and MML. RSM uses shared category thresholds. With PCM, set `step_facet` explicitly instead of relying on facet-name inference or the first-facet fallback. Ordinary RSM/PCM MML with `population_formula = NULL` fixes the ability distribution at N(0,1). Both extended RSMs instead estimate ability variance by default (`person_sd = NULL`). Their model-comparison tutorials show how to match population assumptions. Changing only the fitting function can change more than the rater/dependence structure.
- **Estimator choice:** MML is the current default, not the result of an automatic comparison of estimators. It uses a population model for abilities. JML estimates each person's ability as a separate parameter; it does not fit a population distribution during calibration. Choosing JML does not automatically correct the bias associated with few ratings per person. Without `jml_correction_order`, JML is uncorrected and its location SEs are exploratory. Shared-owner GPCM also supports an explicit, experimental correction order. Its `RootSE` describes local variation around the adjusted-equation root; residual bias may remain and structural confidence intervals are not supplied. See the estimator discussion in `fit_mfrm()`. Neither a larger model nor a smaller reported SE establishes a better analysis. Choose from the assessment question, rating design and modeling assumptions, rather than from whether the user is a beginner or expert.
- **Rating scale:** supply `rating_min`, `rating_max` and `category_policy` from the rubric in both data review and ordinary fitting. Omitted bounds use the observed range. The ordinary default `category_policy = NULL` retains `keep_original = FALSE` and can collapse unobserved internal categories, for example observed 1, 3, 5 to 1, 2, 3. This changes the fitted category structure, not just labels. A warning and the stored score map identify the recoding. Inspect `CategoryPolicy` and `ScoreRecoded` in `data_review$overview` or `summary(fit)$settings_overview` to distinguish the chosen rule from an actual change to score values. With a complete contiguous scale, `"collapse"` can still have `ScoreRecoded = FALSE`. Use `category_policy = "preserve"` to preserve the declared ladder; an unsupported internal category then stops fitting and needs substantive review.
- **Missingness and assignment:** ordinary fitting excludes rows missing a score or required ID (and nonpositive-weight rows); inspect `fit$prep$row_retention`, `fit$prep$preparation_notes` and `fit$prep$score_map`. Missing-code conversion is off unless requested. In contrast, the

extended models stop on missing assigned scores by default, and feature/G-study routes also require an explicit omission choice. Omission does not correct informative missingness. In `review_mfrm_imputations()`, `assigned = NULL` declares every supplied row assigned: supply an assignment column if unassigned rows are present.

- **Included effects:** both extensions default to no additional fixed facets. For example, `testlet = "Task"` groups local dependence but does not itself add fixed task difficulty; request that with `facets = "Task"` when it is part of the intended model.
- **Feature geometry:** PCA and direct k-means default to `scale = TRUE` and equal feature weights. Thus years of experience and hours of training contribute in SD units; `scale = FALSE` retains their measurement units. PAM/hierarchies instead use Gower scaling, including numeric ranges. Choose `k` explicitly. PCA retains all numerically nonzero components unless `components` is specified; no automatic reduction is implied. K-means on a PCA result reuses its fitted transformation.
- **Planning:** multivariate G-studies default to complete balanced crossed ANOVA. Incomplete or nested data need the documented explicit choices; the function does not silently choose another estimator or design. D-study `weights = NULL` reports the original scores separately, not an equal-weight total. Supplied weights are not normalized: an average and a sum have different score/SEM units. Specify `design_grid` for the plans you want to compare.
- **Uncertainty and flags:** `mfrm_facet_intervals()` defaults to `method = "model"`; request `"sandwich"` explicitly, with the appropriate independent clusters. The default confidence level is pointwise 0.95, not simultaneous coverage across raters. Rubin pooling defaults to `df_complete = Inf`, a large-sample complete-data approximation, not a degrees-of-freedom estimate from the number of rating rows. Extended calibration bounds are omitted by default; conditional Person bounds describe a different target. Misfit bands can also depend on session options: inspect `mfrm_misfit_thresholds()` and the returned screening settings. With `fit_measures_table()`, specify `lower`, `upper` and `flag_basis` to record a chosen rule. In a dashboard, numeric `misfit_warn` uses a reciprocal lower bound; it is not just a replacement upper bound. None of these conventions establishes a universal rater-quality threshold.

Then choose the workload and display. The default `summary(fit)` is a lightweight fit summary; ordinary `mfrm_results(fit)` can compute missing diagnostics. Use saved diagnostics or `compute = "never"` for review without new diagnostics. Source-Person scoring requests everyone when `persons` is omitted. K-means defaults to 25 starts with `seed = 1`, and `silhouette = TRUE` adds pairwise distances; use `FALSE` when that optional calculation is not needed. A fixed seed provides reproducibility, not evidence that the partition is best. Plot styles, titles and labels affect presentation; changing the model, feature scaling or thresholds requires recomputing the affected analysis. Keep full result objects and their resolved settings with the script, not only the visible tables. For ordinary fits, start with `summary(fit)$settings_overview`; extended summaries also expose `settings` and `data_usage`.

Use your own ratings

The "Use your own CSV" section of `vignette("mfrmr-workflow", package = "mfrmr")` covers CSV import, column-name mapping, and reshaping a sheet with separate criterion columns. The following help pages are also available without an installed vignette:

- `describe_mfrm_data()` explains input and retained row counts, category usage, connectedness, and how to follow up input problems.

- recode_missing_codes() shows how to replace documented missing-score markers while preserving person and rater IDs.
- summary.mfrm_data_description() shows a missing-score example and explains the full data checks versus their compact summary tables.

Set the score bounds from your rubric and use the same columns, bounds, and category_policy setting for the review and the fit. Reviewing data does not change the original ratings. After correcting or recoding them, repeat the review and pass the corrected data frame to fit_mfrm(). A data_review object contains checks; it is not the rating data to fit.

Choose which ratings share an effect

The routine fit/diagnose/results/report workflow starts with ordinary RSM/PCM. Other fit_mfrm() models share the mfrm_fit class but have different restrictions. Check gpcm_capability_matrix() and vignette("mfrmr-gpcm-scope", package = "mfrmr") for one-family GPCM, two-family GPCM or corrected JML; the latter two do not supply ordinary fit tests or Wright/Pathway maps. Use mfrmr_output_guide("models") to compare:

- **Fixed facets:** fit_mfrm() describes the specified rater and task levels. For supported ordinary models, native diagnostics, Wright maps and comprehensive reports use the matching mfrm_fit result.
- **Shared random raters:** fit_mfrm_random_rater() models one rater effect shared across all Persons rated by that rater. For abilities of source Persons, use score_mfrm_persons(). predict() instead gives score probabilities at supplied abilities for observed or replacement raters. See vignette("mfrmr-random-raters", package = "mfrmr").
- **Person-specific testlets:** fit_mfrm_testlet() groups dependent ratings within each Person. Use score_mfrm_persons() for abilities of source Persons; predict() also accepts a supplied rating table for scoring. Both hold fitted calibration fixed. A Rater column can specify fixed severity and local within-Person membership. See vignette("mfrmr-testlets", package = "mfrmr").

Choose the next call from the input and the quantity you need:

Input and purpose	Entry
Source responses; estimate abilities in either extension	score_mfrm_persons()
Supplied shared-rater responses; estimate abilities	score_mfrm_random_rater(newdata = ...)
Supplied testlet responses; estimate abilities	predict(testlet_fit, newdata = ...)
Supplied abilities; predict shared-rater rating probabilities	predict(random_fit, ability = ...)

For the last row, choose rater = "observed" or rater = "new" explicitly. There is no corresponding supplied-ability prediction route for testlets.

A supplied scoring roster replaces source responses. In the shared-rater model, an existing rater ID alone does not retain that rater's earlier responses or saved conditional distribution; include all intended responses once. In the testlet model, a column used both for local membership and as a fixed facet still requires known fixed levels: a new Person can be scored, but a new fixed rater cannot be introduced by changing its testlet label.

Shared-rater ability scoring can be slow because each Person's calculation uses the complete rating table. Start with a few actual IDs in `persons`; this selects output rows without discarding other Persons' ratings. Omitting `persons` requests everyone.

All three have `summary()`, `plot()`, `plot_data()` and ordinary RDS saving. Extended-model plots additionally support `as_ggplot()`, preserving each interval's target, prior-only symbols and unavailable rows. Check `mfrmr_interval_guide()` for each interval's target and limitations. The two extended models have their own classes. `mfrm_results()` collects their saved calibration, numerical checks and interval meanings; attach separately computed `predictions` and random-rater `intervals` explicitly. Calibration tables retain estimates and approximate SEs but omit bounds by default. For either extension, `confint(fit, parm = "calibration")` explicitly requests pointwise normal approximations with unestablished finite-sample coverage. Use `mfrm_results(fit, calibration_intervals = "normal", calibration_level = 0.95)` to retain that choice in reports and saved output. `mfrm_report()` and `export_mfrm_results()` reuse these results without fitting, scoring or resampling. Older predictions need regeneration from the saved fit to carry matching source metadata; no refit is needed. `diagnose_mfrm()`, the viewer, ordinary Wright maps, response-MI pooling and portable-calibration extraction do not accept these model classes. `mfrm_response_diagnostics()` separately integrates latent uncertainty to produce same-data posterior predictive residuals and descriptive Infit/Outfit for either extension. Paired/scatter plots have no reference cutoffs; ordinary plug-in fit values are not directly comparable. Attach saved output with `mfrm_results(fit, diagnostics = response_review)`; no integration runs during reporting. Numerical checks are not substitute model-fit diagnostics. Refit when changing the statistical model; editing a saved object's class is invalid. Replotting or exporting an existing table does not require refitting.

Assessment planning and external features

To compare tasks, raters or score weights using numeric observed scores, start with `mfrm_g_study()` and `mfrm_d_study()`. Their help covers crossed facets, task-specific rater teams, incomplete source designs and reading G/Phi/SEM plots. For prespecified plan differences under normal random effects with two crossed facets, see `mfrm_multivariate_d_compare()`. These analyses do not require an MFRM fit and do not estimate reliability on its latent scale. In particular, a testlet variance is on the latent logit scale and cannot be substituted for an observed-score G-study component. Start G/D analyses from the ratings and their declared G-study design. Multivariate here means several observed score criteria, such as fluency and accuracy, whose source-specific covariances are estimated together. Multiple GPCM slope owners still describe a single latent ability. Both provisional two-family MML engines support summaries and conditional fitted curves with saved reports, separately checked experimental component log-Wald and location/contrast intervals, and descriptive posterior residuals. Component profiles preserve the fitted fixed-grid or adaptive integration; every constrained solution needs its own numerical checks. Separately checked fitted-object and portable new-Person EAP intervals condition on the calibration and prior; they exclude calibration uncertainty and have unqualified coverage and population transport. Neither inherits the one-family diagnostic/inference workflow below. See `gpcm_capability_matrix()` and the Two slope families section in `fit_mfrm()`. Neither the estimated slopes nor their sampling covariance can substitute for G-study score covariance components. Score weights specify the intended composite; they are not discrimination estimates. There is no automatic GMFRM-to-G-study conversion or joint estimation in these APIs. Hirai and Koizumi (2013, doi:10.1080/15434303.2013.824973) illustrate complementary multivariate G-theory and MFRM analyses. Their G-study omitted raters, so its generalization scope differs from a person-by-rater-by-task analysis. For the relationship and

literature, see `vignette("mfrmr-gpcm-scope", package = "mfrmr")`.

Does MFRM simulation replace a G-study? Not automatically, and an analysis need not use both. Choose the score and decision first:

- A G/D-study asks how a specified observed mean/composite generalizes over sampled tasks, raters or other supported conditions. Its variance and covariance components include interactions; relative and absolute error definitions determine G, Phi and SEM. Future counts reweight these components without repeatedly generating ratings and refitting MFRM.
- `evaluate_mfrm_design()` generates ordered ratings under a declared MFRM specification, refits models and summarizes recovery, separation and fit screens. It asks how that procedure behaves under those assumptions. Increasing repetitions reduces Monte Carlo error, not uncertainty about an incorrect generating model or whether future raters are exchangeable.
- If the decision concerns calibrated latent scores, a suitably specified simulation may be sufficient. If it concerns the dependability of a raw mean or composite, a G-study provides that target directly. Simulation can also evaluate an observed-score target, but it must explicitly define the score, sampling universe, dependence and error criterion; a facet separation coefficient is not a substitute for G or Phi.

For example, adding equally severe raters may improve Person precision without removing Person-by-Task variation. An additive MFRM generator does not acquire that latent interaction simply by increasing facet counts; a G-study estimates observed-score interaction components, whose scale is different from a latent interaction variance. Neither method validates its own assumptions merely by reproducing its own generated data. These APIs do not fit a joint MFRM/G-theory model. The current simulation planner uses Person, rater-like and criterion-like roles; changing their names or level counts does not provide an arbitrary-facet planner. The multivariate G-study separately supports one or two random measurement facets. See `build_mfrm_sim_spec()` and `mfrm_g_study()` for the respective design boundaries.

To group persons, raters or tasks by external attributes, use `mfrm_features()` followed by `mfrm_cluster_pam()` or `mfrm_cluster_hierarchical()` for mixed attributes. `mfrm_pca()` and `mfrm_cluster_kmeans()` use selected numeric features with explicitly chosen scaling and component counts. See `vignette("mfrmr-external-featu` `package = "mfrmr")` for imputation and setting comparisons. These are exploratory attribute groups, not estimated ability classes or rater-quality judgments. Use `mfrmr_output_guide("features")` or `mfrmr_output_guide("gtheory")` for their dedicated output routes. Use `summary()` for reviews and comparisons, and `plot()` or `plot_data()` for fitted PCA, partitions and D-studies. Save the full analysis with `saveRDS()`; these are not `mfrm_results()` inputs. Export a chosen summary with `write.csv(summary(result), ..., row.names = FALSE)` and keep the full source object for its settings, excluded rows and assumptions. Multivariate G-study component panels/covariance heatmaps and D-study curves support `as_ggplot()`, retaining score identities, G/Phi or SEM, facet-count panels and unavailable values. Plan-difference plots use base graphics or custom graphics from `plot_data()`; automatic conversion is unavailable. PCA scree/scores/loadings, saved dendrograms, imputation co-membership, silhouettes and numeric/categorical cluster profiles have dedicated `as_ggplot()` conversions. Use `mfrmr_output_guide("plots")` to choose a supported view; data extraction alone does not guarantee equivalent graphics.

Missing scores on assigned ratings

`review_mfrm_imputations()` reviews supplied ordinal completions; `fit_mfrm_imputed()` fits each dataset and retains failures. Inspect every completion before `pool_mfrm_imputed()` combines

eligible non-Person facet estimates or prespecified contrasts and their covariance. This route does not fill unassigned cells or pool Person EAPs. The imputation model and the fixed-standard-normal RSM/PCM MML analysis must be justified together. See `mfrmr_output_guide("imputation")` and `vignette("mfrmr-response-imputation")` for an executed example and its limits. A pooled result has dedicated `summary()`, `plot()` and `plot_data()` routes; use `saveRDS()` for the complete analysis or `write.csv(summary(pooled), ...)` for a selected table. It is not an `mfrm_results()` input. `as_ggplot()` supports the saved pooled interval view, including `component = "table"`, and preserves its t intervals, fixed targets and inference settings. Use `plot_data()` for other components; conversion does not refit or pool the analyses again.

Updating saved analyses for 0.2.4

Keep the original objects, data and analysis settings. Installing an update does not recalculate saved tables, figures or reports. For analyses based on a native MFRM fit, start by printing `summary(fit)` under the updated package and reading its interpretation decision. If the saved native fit lacks the current estimation checks, refit from the original data with the same intended model, category coding, anchors, weights and numerical settings. Running `diagnose_mfrm()` alone cannot establish checks missing from that fit. If the original data or settings are unavailable, retain the old result as historical output; its previous approval is not evidence for current inferential use. Observed-score G/D studies and external-feature groups instead use their own source objects; they do not require an MFRM fit.

Update the affected result at the earliest step below, then rebuild its dependent summaries, plots and exports. For an MFRM-based analysis, this assumes a fit with current estimation checks. A request to recompute diagnostics or scoring does not itself require a new calibration fit.

- **Display wording only:** reprint a saved fit summary. This updates labels and removes the duplicate inference decision; it does not recalculate stored diagnostics or establish missing precision evidence.
- **Diagnostics and QC:** recreate `diagnose_mfrm()` results with the original diagnostic settings before rebuilding reliability, precision, category, marginal-fit, person-fit, unexpected-response, fair-score and reporting results. Rerun separately requested `q3_statistic()`, `analyze_residual_pca()` and `compute_person_fit_indices()` calculations with their original options. In `run_qc_pipeline()`, request `separation_facets` only for facets whose levels need to be distinguished; it is no longer a default rater-quality requirement. Missing results remain unavailable and cannot be treated as passes.
- **Model comparisons:** rerun `compare_mfrm()` on the existing eligible fits, retaining the original comparison settings, then rebuild dependent `build_model_choice_review()` and `build_weighting_review()` results. The corrected RSM/PCM nesting review requires every RSM interaction to be retained in the PCM. More category-step parameters do not establish nesting. An affected pair can retain an eligible information-criterion comparison while losing its invalid chi-square test. This correction does not require refitting; printing an old comparison leaves its stored decision unchanged.
- **Group comparisons and equivalence:** recreate residual `analyze_dff()` / `analyze_dif()` and `dif_interaction_table()` results from the existing fit and original group data. They now describe residual differences without tests or confidence intervals. Recompute `analyze_facet_equivalence()` from an eligible MML fit with matching current diagnostics and the original practical bound; old equivalence bundles cannot supply the required joint covariance.

- **ICC and design effects:** rerun `compute_facet_icc()` or `analyze_hierarchical_structure()` from the original data and settings. Numeric score labels now retain their values. Missing scores or grouping values require an explicit `missing = "omit"` choice; malformed scores must be corrected. Inspect `attr(icc, "data_usage")`, then recreate `compute_facet_design_effect()` using the new ICC result. Older ICC tables lack the row accounting needed to calculate matching sample sizes. Recalculation also removes the fixed variance cutoff tied to score units and preserves small positive variances without decimal rounding. Constant retained scores have unavailable variance components and ICCs. Design effects remain per-facet approximations; their equivalent row counts do not estimate the precision of a complete crossed or unbalanced design. The former `ci_method = "profile"` transformed separate component bounds and did not calculate a profile-likelihood interval for the ICC ratio. Choose `"boot"` explicitly for parametric percentile intervals, then read `ICC_CI_Status` and `attr(icc, "icc_ci")`. Failed or nonconverged refits and fit warnings withhold intervals. Saved bootstrap results also require rerunning to obtain complete failure accounting and identify constant-response refits, which withhold intervals; reprinting is insufficient.
- **Observed-score design coefficients and shrinkage:** rerun `mfrm_generalizability()` with its original data and settings, then `mfrm_d_study()` with the planned counts and residual-scaling choice. This re-estimates the separate observed-score mixed model, not the MFRM. Reapply `apply_empirical_bayes_shrinkage()` with the original prior settings and explicit Person choice to refresh shrinkage reports, descriptive bands, and replay settings. Reapplication replaces the previous adjustment; switching Person shrinkage off removes its old adjustment columns. Regenerate replay scripts to retain the post-fit adjustment step.
- **Multivariate G/D studies:** to apply metric-specific G/Phi/SEM availability rules, rerun `mfrm_d_study()` from the saved G-study with the original planned counts and score weights. Changing only future counts or weights also reuses that G-study. Recreate dependent plan comparisons, plots and exports after recalculation; replotting alone preserves stored numbers. Refit `mfrm_g_study()` when changing the source data or G-study model, replacing an object with missing or incompatible design metadata, or correcting an earlier G-study affected by the single-score MINQUE(0) or period-containing interaction-ID bugs. In particular, a crossed result cannot be converted to a nested model by editing its labels; refit with an explicit `nesting` specification. Retain the G-study object and its data to calculate new prespecified plan comparisons with `mfrm_multivariate_d_compare()`. A coefficient table alone does not contain the information needed for those intervals.
- **External-feature groups:** saved clustering results retain their memberships and fitted hierarchy. Replot them to update labels; use `plot_data()` for their stored values. Automatic `as_ggplot()` conversion is not supported for these plots. A change to features, weights, group counts or method requires a new clustering call, followed by `mfrm_cluster_compare()` on the updated results. For imputation comparisons, reuse the original `mice` object to preserve pairing across completed datasets; do not generate unrelated completions for each setting.
- **Fitted-object Person scores:** re-summarize the original prediction object to recover stored interval settings and updated explanations. To replace older grid-endpoint intervals with continuous posterior intervals, rerun `predict_mfrm_units()` using the existing fit, scoring data and settings. Reprinting cannot change an already calculated interval. Rerun scoring when older estimated-population results lack numerical prior information or are refused because they claimed unrestricted scoring; these fits require explicit `readiness_policy = "review"`.
- **Plausible values:** use `summary(original_plausible_values)` on the saved draw object to obtain empirical quantiles at its requested interval level. No new draws are needed for this

correction. A saved derived summary alone cannot recover the original draws. If estimated-population restrictions require regeneration, repeat the original scoring/draw call with the same data, settings and seed, and `readiness_policy = "review"`.

- **Portable calibration:** `load_mfrm_calibration()` preserves a valid artifact's recorded scoring algorithm; older grid-based intervals remain grid-based. To adopt continuous intervals, create a new calibration using the reviewed source fits and mfrm_calibration_workflow, then score again. Retain the old artifact for reproducibility. Re-summarizing older score results recovers recorded algorithm/level labels without changing values.
- **External fits:** rerun `import_erm_fit()`, `import_tam_fit()` or `import_mirt_fit()` on the saved source-package fit and rebuild displays; source-model re-estimation is unnecessary. Use `compute_fit = TRUE` when imported measurement diagnostics are needed. Unsupported source models remain unsupported, and imports do not become native fits.
- **Agreement, networks and timing:** rebuild `interrater_agreement_table()`, `rater_network_analysis()` and `rater_halo_network_analysis()` from the existing native fit, matching diagnostics and original settings. Refresh design reviews with `build_mfrm_network_review()` to record whether the graph covers all observed subsets. Recreate `response_time_review()` from the original timed-event data and settings. No MFRM refit is required. Unavailable comparisons remain unassessed; a missing or excluded graph edge does not establish absence of a rater effect. Halo Welch-test columns are retained as missing values. Timing rates use valid times and describe cutoff rules, not calibrated rapid-guessing or low-effort classifications.
- **Simulation and design summaries:** re-summarize saved evaluation objects to retain attempted-run denominators and unavailable residual-DIF rates. Rebuilding design summaries also restores unrounded metrics for threshold decisions; rounded saved summaries alone cannot recover that precision. Missing workload or connectivity records cannot be reconstructed by summary formatting. If those records are required for a recommendation, repeat the original evaluation with its recorded design, settings and seeds.

After any required recalculation, regenerate dependent `mfrm_results()`, reports and exports with matching source objects. A newly created PDF, HTML file or CSV can still contain outdated calculations if built from an old derived object. Consult the affected function's help for its interpretation limits; updating an object does not broaden those limits.

Next steps for reporting

For ordinary MML/JML `mfrm_fit` results, use one basic route: `summary(fit)` -> `res` <- `mfrm_results(fit)` -> `summary(res)`. The summary's `triage` and `next_actions` identify what needs attention; `plot_map` lists supported figures. The collector computes missing requested diagnostics by default. If you already have matching diagnostics, pass them as `diagnostics = diagnostics`; `compute = "never"` keeps absent diagnostics unavailable rather than computing them. It does not certify readiness.

From `res`, choose `mfrm_report()` for a report, `plot(res, ...)` for a supported figure, `export_mfrm_results()` for an analyst archive, or `saveRDS(res, file)` for later R use. These are optional branches, not a checklist that must be run in full. The worked example in mfrmr_reports_and_tables follows this route. `build_apa_outputs()` and `build_summary_table_bundle()` serve additional manuscript/table assembly; they are not prerequisites for the ordinary report.

CML and comparison-only fits use their separate structural/comparison workflow above. For GPCM and extended models, consult `mfrm_results()` for the supported sections and attachments before collecting a report. If an earlier script uses `summary(fit, profile = "facets")`, reuse its

`$results` or `$results$diagnostics`. The historical profile name does not require FACETS, TAM, or sirt software.

Use `JML` only when its fixed-person-parameter estimand is methodologically intended, for example for a JMLE-oriented external comparison, descriptive or exploratory work, or a design with substantial information per person. Do not select it merely as a faster substitute for `MML`: a later `MML` run targets a different estimand rather than serving as stricter follow-up to the same analysis.

Canonical operational review route

When the main question is scale maintenance rather than manuscript reporting, branch from `review$results$diagnostics` into: `review_mfrm_anchors()` and/or `detect_anchor_drift()` -> `build_equating_chain()` when adjacent-link review is needed -> `build_linking_review()` -> inspect `review$group_view_index` for stable wave / link / facet rollups and `summary(review)$plot_routes` for the next plot helper -> plot(x) or `plot(anchor_review, ...)` for the specific flagged evidence family.

For `GPCM`, use `build_linking_review()` as a caveated exploratory synthesis over direct anchor, drift, and chain evidence. It is not an operational `GPCM` linking decision or evidence that anchor drift is absent.

Canonical misfit case-review route

When the main question is which observations, facet levels, or pairwise structures deserve follow-up, branch from `review$results$diagnostics` into: `build_misfit_casebook()` -> inspect `casebook$group_view_inde` `casebook$group_views`, and `summary(casebook)$plot_routes` for stable person / facet / wave rollups and the next plot helper -> plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "marginal_fit"), or plot(x, type = "marginal_pairwise") according to `casebook$plot_map` -> `build_summary_table_bundle()` / `export_summary_appendix()` when the flagged cases need appendix-style reporting support.

`build_misfit_casebook()` can still be used for `GPCM`, but it should be read as an operational exploratory screen rather than as a strict Rasch-style invariance report.

Latent-regression route

When the fit uses `population_formula = ...`, keep the distinction between the estimator and the forecast helpers explicit:

- `fit_mfrm()` estimates the current narrow latent-regression `MML` branch. In the returned fit object, `fit$population$person_table` is the complete-case estimation table, while `fit$population$person_table_r` retains the observed-person-aligned pre-omit background-data table for replay/export provenance.
- `predict_mfrm_units()` and `sample_mfrm_plausible_values()` can then score under the fitted population model when scored units also supply one-row-per-person background data. That scoring-time `person_data` contract remains separate from the fit object's stored replay table. Estimated-population scoring currently requires explicit `readiness_policy = "review"`. Scores and intervals hold the estimated calibration and population parameters fixed; their estimation uncertainty is omitted. These draws alone do not justify downstream group comparisons or regression without a compatible conditioning model and sampling design.
- `predict_mfrm_population()` remains a scenario-level simulation/refit helper rather than the latent-regression estimator itself.

Score-category support

If the intended rating scale includes categories not observed in the current data, make that support explicit. For example, use `rating_min = 1, rating_max = 5` for a 1-5 scale with only 2-5 observed. This preserves the declaration in the data-support review. A zero-count boundary is review evidence for the separate element-boundary contract; it is not by itself an unsupported free-step contrast. If an intermediate category is unobserved (for example 1, 2, 4, 5 with no 3), also set `category_policy = "preserve"` if the zero-count category should remain in the fitted support. `summary(describe_mfrm_data(...))` reports retained zero-count categories in `Notes`, printed `Caveats`, and `$caveats`; `summary(fit)` carries full structured rows into printed `Caveats` and `$caveats`, with `Key warnings` as a short triage subset. Summary-table exports route those rows through `score_category_caveats` or `analysis_caveats`. In a polytomous fitted ladder, a retained zero-count internal category creates an unsupported adjacent-step contrast and stops fitting before optimization.

Planned assignment and structural missingness

A long-format table alone does not reveal whether an absent Person x facet cell was expected or never assigned. When a score-free assignment roster is available, pass it as `expected_design` to `describe_mfrm_data()`. The summary then separates expected-but-unobserved cells from unexpected observations and reports observed versus declared Person-facet graph components. Without a roster, structural missingness is explicitly marked as not assessed; mfrmr does not assume a complete crossing.

Typical workflow

The steps below describe ordinary MML/JML `mfrm_fit` results. Native CML uses the separate route in **From the first summary to diagnostics**: read its structural or comparison summary, create `mfrm_facet_intervals()`, and use the interval object's plots/tables and RDS saving. CML has no Person abilities or population SD and does not support the ordinary diagnostic, Wright-map or consolidated-report steps. `mfrmr_interval_guide()` also maps full MML interaction contrasts and their explicit population-SD standardization.

1. Review the long-format data and intended score support with `describe_mfrm_data()`.
2. Fit a model with `fit_mfrm()`. Choose `MML` or `JML` from the prespecified estimand and assumptions; do not select `JML` merely to shorten runtime.
3. Read `summary(fit)$decision`. For a usable ordinary fit, collect `res <- mfrm_results(fit)` and inspect `summary(res)$triage` and `$next_actions`. The same `res` supports reporting and saving; use `$plot_map` to select an optional figure.
4. (Optional) Use `run_mfrm_facets()` or `mfrmRFacets()` only when a legacy-compatible one-shot wrapper is needed.
5. For a focused diagnostic question, reuse `res$diagnostics`. Call `diagnose_mfrm()` separately when its settings need to change. For final reporting, prefer `diagnostic_mode = "both"` so the legacy residual path and the strict marginal screen remain visible side by side. For `GPCM`, diagnostics are now available through `diagnose_mfrm()` together with `analyze_residual_pca()`, `interrater_agreement_table()`, `unexpected_response_table()`, `displacement_table()`, `measurable_summary_table()`, `rating_scale_table()`, `facet_quality_dashboard()`, `reporting_checklist()`

and plot(x, type = "qc") – the fair-average panel of the dashboard reports an explicit unavailability indicator under GPCM. Use fair_average_table() directly when you need the supported slope-aware element-conditional fair averages. Treat those residual-based summaries as exploratory screens because the discrimination parameter is free. Full FACETS-style score-side contract review remains blocked for GPCM; package-native scorefile export, fit-based reporting bundles, direct fair-average tables, and bias-screening tables carry their own caveats. Posterior scoring with predict_mfrm_units() / sample_mfrm_plausible_values(), design-weighted information via compute_information() / plot(x), Wright/pathway/CCC plots via plot.mfrm_fit(), direct category reports via category_structure_report() / category_curves_report(), and direct data generation through build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data() are also available when the simulation specification stores both thresholds and slopes. Use simulate_mfrm_recovery() and assess_mfrm_recovery() for direct recovery checks plus caveated role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers. Caveated APA/QC/export bundles are available for sensitivity reporting, while score-side FACETS helpers remain outside the documented GPCM boundary. Use gpcm_capability_matrix() as the formal capability map before branching into less common helpers. Residual DIF/DFF differences remain descriptive; their detection and false-positive rates are unavailable in evaluate_mfrm_signal_detection(). To evaluate your own declared rater-warning procedure against known simulation truth, use mfrm_screening_performance() with a complete planned roster. It distinguishes individual and any-target rates, Monte Carlo uncertainty and unavailable outcomes. See vignette("mfrmr-screening-performa package = "mfrmr").

6. (Optional, RSM / PCM; GPCM with caveat) Estimate interaction bias with estimate_bias().
7. Choose a downstream branch: reporting_checklist() for direct report preparation, or build_weighting_review() for Rasch-versus-GPCM weighting review, or build_misfit_casebook() / build_linking_review() for operational case review. For GPCM, use build_linking_review() only as an exploratory index over direct anchor/drift/chain evidence.
8. For ordinary reports and saving, use mfrm_report(), export_mfrm_results() or saveRDS(res, file). For additional manuscript and appendix assembly, select the relevant helper: build_summary_table_bundle() apa_table(), export_summary_appendix(), build_fixed_reports(), build_visual_summaries(). For GPCM, use the APA, visual, QC, and fit-based export bundles as caveated sensitivity-reporting surfaces; full score-side FACETS review stays blocked, while diagnostic/signal-detection design screening has its own caveated operating-characteristic route.
9. (Optional, RSM / PCM) Review report completeness with reference_case_review(). Use facets_output_contract_review() only when you explicitly need the compatibility layer.
10. (Optional, RSM / PCM) For operational linking follow-up, combine review_mfrm_anchors(), detect_anchor_drift(), and build_equating_chain() inside build_linking_review() before exporting appendix-style tables.
11. (Optional) Check packaged reference cases with reference_case_benchmark() when you want package-side reference checks.
12. (Optional) For design planning or future scoring, move to the simulation/prediction layer: build_mfrm_sim_spec() / extract_mfrm_sim_spec() -> simulate_mfrm_recovery() -> assess_mfrm_recovery() / evaluate_mfrm_design() / predict_mfrm_population() -> predict_mfrm_units() / sample_mfrm_plausible_values(). Current fit-derived simulation specs include direct GPCM data generation and recovery checks. Design-evaluation, population-forecasting, diagnostic- screening, and signal-detection helpers also support GPCM as caveated role-based simulation/refit evidence; inspect gpcm_boundary before using those

results in design claims. Unit scoring can use an ordinary MML fit directly, a latent-regression MML fit when you also supply one-row-per-person background data for the scored units, or a JML fit when a post hoc reference-prior EAP layer is acceptable. Estimated-population fits require explicit readiness_policy = "review"; this does not remove their interpretation limits. Intercept-only latent-regression fits (population_formula = ~ 1) can reconstruct that minimal person table from the scored person IDs. Keep predict_mfrm_population() conceptually separate from that scoring layer: it is a simulation-based scenario forecast helper, not the latent-regression estimator itself. Prediction export still requires actual prediction objects in addition to include = "predictions".

13. Use summary() for compact text checks and plot() (or dedicated plot helpers) for base-R visual diagnostics.

Three practical routes

- Quick first pass: RSM / PCM: fit_mfrm() -> diagnose_mfrm() -> plot(x, type = "qc") -> reporting_checklist() when you want the package to route the next figures. GPCM: fit_mfrm() -> diagnose_mfrm() -> plot(x, type = "qc") / unexpected_response_table() -> rating_scale_table() -> compute_information() -> plot(x) -> plot.mfrm_fit() / category_curves_report() -> fair_average_table() / estimate_bias() when those screening tables answer the question. For GPCM, the fit-based export family (build_mfrm_manifest(), build_mfrm_replay_script(), export_mfrm_bundle()) is available as caveated sensitivity-reporting output with explicit gpcm_boundary rows.
- Linking and coverage review: subset_connectivity_report() -> plot(..., type = "design_matrix") -> plot(x, type = "wright", view = "unified").
- Manuscript prep: RSM / PCM: reporting_checklist() -> inspect the "Visual Displays" and "Method Section" rows -> build_apa_outputs() -> build_summary_table_bundle() -> apa_table() or export_summary_appendix(). GPCM: reporting_checklist() -> direct table/plot helpers -> build_apa_outputs() / build_visual_summaries() -> export_mfrm_bundle() with gpcm_boundary caveats.
- Weighting-policy review: compare_mfrm() -> build_weighting_review() -> compute_information() / plot(x) when you want to inspect whether GPCM is introducing substantively acceptable discrimination-based reweighting relative to the Rasch-family reference. Eligible MML comparisons require a common grid of at least 31 points and a denser common-grid sensitivity check when close or consequential. GPCM MML ranking requires the separate likelihood and solution checks; nested = TRUE additionally checks the PCM/GPCM equal-slope test; confint.mfrm_fit() with parm = "slopes" separately checks approximate slope intervals, with explicit standardized-scale, contrast, sandwich and Bonferroni options. bootstrap_mfrm_gpcm() supplies a fitted-model bootstrap alternative; mfrm_curve_intervals() adds uncertainty to probabilities and per-rating information. mml_quadrature_sensitivity() preserves explicit population covariates and reports interval changes across grids. Attach the explicitly selected intervals with mfrm_results(fit, intervals = list(slopes = ci)); plot via type = "gpcm_slopes" or use apa_table() and plot_data(). Saved reports/exports retain methods and targets without refitting.
- Design planning and forecasting: build_mfrm_sim_spec() or extract_mfrm_sim_spec() -> simulate_mfrm_recovery() -> assess_mfrm_recovery() for parameter-recovery checks, then evaluate_mfrm_design() -> predict_mfrm_population() -> predict_mfrm_units() or sample_mfrm_plausible_values() under the fitted scoring basis (ordinary MML, latent-regression MML with person-level background data, or JML with the documented post hoc EAP

approximation). Here again, predict_mfrm_population() is the scenario-level forecast helper, whereas predict_mfrm_units() / sample_mfrm_plausible_values() are the scoring layer. Prediction export requires actual prediction objects. GPCM supports direct data generation via build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data(), simulate_mfrm_recovery(), assess_mfrm_recovery(), caveated role-based design evaluation and population forecasting, diagnostic/signal-detection design screening, residual diagnostics, and direct curve/report helpers. The current planning layer remains role-based for two non-person facets even though estimation itself supports arbitrary facet counts. Additional arbitrary-facet fields are structural design metadata, not Monte Carlo performance results.

Interpreting output

This help page is a map, not an estimator:

- use it to decide function order,
- confirm which objects have summary()/plot() defaults,
- identify when dedicated helper functions are needed,
- and treat reporting_checklist() as the package's readiness router for plot and report follow-up.

Objects with default summary() and plot() routes

- mfrm_fit: summary(fit) and plot(fit, ...).
- mfrm_diagnostics: summary(diag); plotting via dedicated helpers such as plot(x, type = "unexpected"), plot(x, type = "displacement"), plot(x, type = "qc").
- mfrm_bias: summary(bias) and plot(x, type = "bias").
- mfrm_data_description: summary(ds) and plot(ds, ...).
- mfrm_anchor_review: summary(review) and plot(review, ...).
- mfrm_misfit_casebook: summary(casebook) and print(casebook), with grouping views available through casebook$group_view_index and casebook$group_views, source-specific plotting routed through summary(casebook)$plot_routes and casebook$plot_map, and appendix/report handoff available through build_summary_table_bundle() and export_summary_appendix().
- mfrm_weighting_review: summary(review) and print(review), with information follow-up routed through compute_information() and plot(x) according to review$plot_map, and appendix/report handoff available through build_summary_table_bundle() and export_summary_appendix().
- mfrm_linking_review: summary(review) and print(review), with grouping views available through review$group_view_index and review$group_views, and plotting routed through summary(review)$plot_routes, plot(x), and plot(anchor_review, ...) according to review$plot_map.
- mfrm_facets_run: summary(run) and plot(run, type = c("fit", "qc"), ...).
- apa_table: summary(tbl) and plot(tbl, ...).
- mfrm_apa_outputs: summary(apa) for compact diagnostics of report text.
- mfrm_summary_table_bundle: print(bundle) for manuscript-oriented table index plus named tables from supported summary() outputs, summary(bundle) for table-role/numeric coverage, and plot(bundle, ...) for table-size or numeric-column QC.
- mfrm_threshold_profiles: summary(profiles) for preset threshold grids.

- mfrm_population_prediction: summary(pred) for design-level forecast tables.
- mfrm_unit_prediction: summary(pred) for unit-level posterior summaries under the fitted scoring basis.
- mfrm_plausible_values: summary(pv) for draw-level uncertainty summaries.
- mfrm_bundle families: summary() and class-aware plot(bundle, ...). Key bundle classes now also use class-aware summary(bundle): mfrm_unexpected, mfrm_fair_average, mfrm_displacement, mfrm_interrater, mfrm_facets_chisq, mfrm_bias_interaction, mfrm_rating_scale, mfrm_category_structure, mfrm_category_curves, mfrm_measurable, mfrm_unexpected_after_bias, mfrm_output_bundle, mfrm_residual_pca, mfrm_specifications, mfrm_data_quality, mfrm_iteration_report, mfrm_subset_connectivity, mfrm_facet_statistics, mfrm_facets_contract_revie mfrm_reference_review, mfrm_reference_benchmark.

plot.mfrm_bundle() coverage

Default dispatch now covers:

- mfrm_unexpected, mfrm_fair_average, mfrm_displacement
- mfrm_interrater, mfrm_facets_chisq, mfrm_bias_interaction
- mfrm_bias_count, mfrm_fixed_reports, mfrm_visual_summaries
- mfrm_category_structure, mfrm_category_curves, mfrm_rating_scale
- mfrm_measurable, mfrm_unexpected_after_bias, mfrm_output_bundle
- mfrm_residual_pca, mfrm_specifications, mfrm_data_quality
- mfrm_iteration_report, mfrm_subset_connectivity, mfrm_facet_statistics
- mfrm_facets_contract_review, mfrm_reference_review, mfrm_reference_benchmark

For unknown bundle classes, use dedicated plotting helpers or custom base-R plots from component tables.

See Also

fit_mfrm(), run_mfrm_facets(), mfrmRFacets(), diagnose_mfrm(), estimate_bias(), mfrmr_visual_diagnostics, mfrmr_reports_and_tables, mfrmr_reporting_and_apa, gpcm_capability_matrix, mfrmr_linking_and_dff, mfrmr_compatibility_layer, summary.mfrm_fit(), summary(diag), summary(), plot.mfrm_fit(), plot()

Examples

```
# Load the package
library(mfrmr)

# Load example ratings and look at the first six rows
ratings <- load_mfrmr_data("example_operational")
head(ratings)

# Fit the model
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
```

```
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM",
  population_formula = NULL # Fixed N(0,1) ability distribution
)

# Plot the results (Wright map)
plot(fit)

# Save the summary, then display its tables
results <- summary(fit)
results$person_overview # One row summarizing person ability estimates
results$facet_overview  # One row per facet: number of levels, mean, SD, range

# Check the interpretation status and recommended next step
results$decision
```

`mfrm_calibration_capabilities` *Portable fixed-calibration capabilities*

Description

Returns the model and estimator combinations supported by the portable calibration workflow. This matrix concerns saved calibration artifacts; it does not replace the wider fitted-object capabilities of `fit_mfrm()` or `predict_mfrm_units()`. For JML fits, fitted-object scoring is post-hoc EAP with a standard-normal reference prior by default, or an explicit `scoring_prior`. It is not ML/WLE scoring or a population distribution estimated by JML. Portable extraction supports RSM/PCM and shared-owner GPCM JML within their distinct source-check scopes. No artifact stores training Person estimates.

Usage

```
mfrm_calibration_capabilities()
```

Value

A data frame with one row per model, estimator, and scoring-basis combination. `PortableCalibration` is either `"available"` or `"unavailable"`; the listed scope and source checks still apply.

Examples

```
mfrm_calibration_capabilities()
```

`mfrm_calibration_methods` *Inspect a portable fixed calibration*

Description

`print()` gives a compact lifecycle and scope overview. `summary()` returns the same identifiers together with coordinate and anchor counts and the structured validation refusals, if any. Both methods inspect the stored artifact and do not refit a model or consult training responses.

Usage

```
## S3 method for class 'mfrm_calibration'
summary(object, ...)

## S3 method for class 'mfrm_calibration'
print(x, ...)

## S3 method for class 'summary.mfrm_calibration'
print(x, ...)
```

Arguments

`object`, `x`	An `mfrm_calibration` returned by the portable calibration workflow.
`...`	Reserved for generic compatibility.

Details

A calibration artifact intentionally has no `plot()` method. Its coordinate table mixes facet, step, and interaction parameter roles, and the artifact does not contain calibration-parameter uncertainty. Plot returned score batches with mfrm_calibration_score_methods instead; inspect `calibration$parameters$coordinates` when auditing stored point coordinates.

Value

`summary()` returns a `summary.mfrm_calibration`. `print()` returns its input invisibly.

See Also

mfrm_calibration_workflow, mfrm_calibration_score_methods

mfrm_calibration_score_methods

Review and plot portable fixed-calibration scores

Description

These methods provide the first review surface for the result returned by score_mfrm_calibration(). print() gives a compact batch disposition, summary() exposes readable score and review tables, and plot() shows conditional score uncertainty or numerical review quantities without refitting a model.

Usage

```
## S3 method for class 'mfrm_calibration_score'
summary(object, digits = 3L, ...)

## S3 method for class 'mfrm_calibration_score'
print(x, ...)

## S3 method for class 'summary.mfrm_calibration_score'
print(x, ...)

## S3 method for class 'mfrm_calibration_score'
plot(
  x,
  type = c("interval", "precision", "edge_mass"),
  top_n = 40L,
  sort_by = c("estimate", "sd", "person"),
  label_review = TRUE,
  main = NULL,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  ...
)
```

Arguments

object, x	An mfrm_calibration_score returned by score_mfrm_calibration().
digits	Number of decimal places in the summary.
...	Reserved for generic compatibility.
type	One of "interval", "precision", or "edge_mass".
top_n	Maximum number of scored Persons shown. Review rows are selected first when truncation is necessary. Use Inf to show all.
sort_by	Selection priority after review rows: absolute estimate, posterior SD, or Person identifier.

label_review	Whether review points are labelled in the precision and edge-mass views. Interval plots always label the displayed Persons.
main	Optional plot title.
draw	If TRUE, draw with base R graphics. The returned mfrm_plot_data is available invisibly in either case.
preset	Visual preset: "standard", "publication", "compact", or "monochrome".

Details

The default interval plot shows posterior EAP estimates and central intervals. type = "precision" plots the number of valid response rows against posterior SD. type = "edge_mass" compares posterior mass on the two outer quadrature nodes with the recorded review threshold. Review dispositions are highlighted in every view.

These are score-batch review displays, not calibration-fit diagnostics. Posterior SDs and intervals are conditional on the frozen point calibration and the actual scoring prior. They exclude calibration-parameter uncertainty. For GPCM or an explicitly supplied scoring prior, summaries and plot data retain both original and actual prior values without rounding, the per-Person integration checks, and (for GPCM) the source decision recorded at extraction. These records describe conditional scoring, not a new evaluation of the training fit or the suitability of its population for another cohort. summary(x)$settings$score_integration_review compares the reported EAP/SD with higher-order adaptive references. The separate quadrature_review compares adaptive and fixed grids; for an adaptive scorer, that fixed grid did not produce the reported scores. Printing distinguishes the two checks. Persons with no valid responses have no score coordinate and are retained in summary(x)$review and in the plot payload's unplotted_dispositions component.

The summary object retains every returned score and review disposition. Its estimates table preserves the available estimate and uncertainty basis columns, calibration identifiers, scoring algorithm and requested interval level when extracted or written to CSV. The interval level is not rounded. Older score results recover these fields from their recorded settings when re-summarized. Saved grid-based intervals retain their original endpoints; printed results and interval plots explain that their posterior mass can differ from the requested level. Its print method shows at most ten rows from each table so routine console output stays compact. Base and ggplot2 renderers distinguish scored and review states by shape as well as colour. as_ggplot(p) and as_ggplot(p, component = "data") preserve the same selected view, saved bounds, review symbols and unplotted dispositions. Use plot_data() to extract other components. Changing the interval level requires an explicit scoring call; conversion does not recompute intervals. Notes remain in the returned object and are not automatically drawn.

Value

print() returns its input invisibly. summary() returns a summary.mfrm_calibration_score containing overview, estimates, review, row_review, settings, and notes. When adaptive integration was requested and scored rows exist, it also retains unrounded quadrature_review and a compact quadrature_overview. When cut scores were requested, decisions preserves every probability, cut boundary and actual scoring prior without rounding; unavailable Persons remain in that table with NA probabilities. plot() returns an mfrm_plot_data with the selected plotting table, selection accounting, unplotted Person dispositions, interpretation guidance, and plotting settings.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

mfrm_calibration_workflow, mfrm_calibration_capabilities, `plot_data()`, `plot_data_components()`, `as_ggplot()`, `apa_table()`, `build_summary_table_bundle()`, `export_summary_appendix()`, `predict_mfrm_units()` for conditional-decision interpretation

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrm_calibration_score_methods", run.dontrun = TRUE)
## Not run:
dat <- load_mfrmr_data("example_core")
ids <- unique(dat$Person)
training <- dat[dat$Person %in% ids[1:18], , drop = FALSE]
fit <- fit_mfrm(
  training, "Person", c("Rater", "Criterion"), "Score",
  model = "RSM", method = "MML", quad_points = 31
)
q_review <- mml_quadrature_sensitivity(
  fit, training, quad_points = c(31, 61), theta_points = 41
)
summary(q_review) # Inspect movement; these grids are not universal cutoffs.
fit <- q_review$fits$q61
calibration <- freeze_mfrm_calibration(
  validate_mfrm_calibration(extract_mfrm_calibration(
    fit, quadrature_review = q_review, scoring_quad_points = 121
  ))
)
```

```
new_rows <- dat[dat$Person %in% ids[19:20], , drop = FALSE]
scores <- score_mfrm_calibration(calibration, new_rows,
  cut_scores = c(Basic = -0.5, Advanced = 0.8))
summary(scores)
apa_table(scores, which = "decisions")
build_summary_table_bundle(scores)$table_index
plot(scores, type = "interval", draw = FALSE)
plot(scores, type = "edge_mass", draw = FALSE)

## End(Not run)
```

`mfrm_calibration_workflow` *Create and use a portable fixed calibration*

Description

These functions implement a strict lifecycle for a saved, versioned calibration. `extract_mfrm_calibration()` creates a draft from an eligible `RSM` or `PCM` MML fit under the fixed standard-normal scoring basis, or a one-family GPCM MML fit with an estimated intercept-only normal population that passes the conditional source checks in `predict_mfrm_units()`. Experimental two-family GPCM MML retains a fixed N(0,1) prior and its own source checks. RSM/PCM and shared-owner GPCM JML support portable post-hoc EAP with a standard-normal reference prior; explicit corrected GPCM JML preserves its correction order and uses adjusted-equation checks, as described below. `validate_mfrm_calibration()` and `freeze_mfrm_calibration()` are separate, fail-closed transitions. Only a frozen artifact can be passed to `score_mfrm_calibration()`.

Usage

```
extract_mfrm_calibration(
  fit,
  calibration_id = NULL,
  source_fit_id = NULL,
  created_at_utc = NULL,
  scoring_quad_points = 31L,
  quadrature_review = NULL
)

review_mfrm_calibration(calibration)

validate_mfrm_calibration(calibration, validated_at_utc = NULL)

freeze_mfrm_calibration(calibration, frozen_at_utc = NULL)

supersede_mfrm_calibration(calibration, record_id, superseded_at_utc = NULL)

retire_mfrm_calibration(calibration, record_id, retired_at_utc = NULL)
```

```
save_mfrm_calibration(calibration, file, overwrite = FALSE)

load_mfrm_calibration(file)

score_mfrm_calibration(
  calibration,
  new_data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  interval_level = 0.95,
  missing_response = "error",
  event_id = NULL,
  adaptive_quad_points = NULL,
  scoring_prior = NULL,
  cut_scores = NULL
)
```

Arguments

`fit`	An eligible `mfrm_fit` produced by `fit_mfrm()`.
`calibration_id`	Optional nonempty calibration identifier.
`source_fit_id`	Optional nonempty source-fit identifier.
`created_at_utc`	Optional RFC3339 UTC timestamp. Omit it to use the current time.
`scoring_quad_points`	Integer quadrature order of at least 2 used for later artifact scoring. It is independent of the fit-time quadrature and defaults to 31. The fixed or adaptive integration mode is inherited from the source fit and stored in the artifact's scoring algorithm identity. With adaptive scoring, stored nodes and weights define the base Hermite rule; their person-specific locations are recomputed for each new batch.
`quadrature_review`	Required for RSM/PCM MML, optional for GPCM MML (which checks the fitted calibration directly); not used when checking JML fits. An `mfrm_quadrature_sensitivity` from `mml_quadrature_sensitivity()` for the same data and model. `fit` must be the exact highest-grid fit stored in this object. The package checks this procedural evidence but leaves acceptable numerical movement to the user.
`calibration`	An `mfrm_calibration` object.
`validated_at_utc, frozen_at_utc`	Optional RFC3339 UTC timestamps for reproducible lifecycle records. Omit them to use the current time.
`record_id`	A nonempty identifier for a terminal superseded or retired record. It must differ from the frozen parent identifier.
`superseded_at_utc, retired_at_utc`	Optional RFC3339 UTC timestamps.

`file` A path ending in `.rds`.

`overwrite` Whether an existing persistence target may be replaced.

`new_data` A data frame of response rows for new Persons.

`person`, `score`, `weight`, `event_id`
Optional input-column names. Stored source-column names are used when applicable. `event_id` distinguishes otherwise duplicate response events.

`facets` Optional named character vector mapping stored facet names to input columns.

`interval_level` Central posterior interval level, strictly between 0 and 1.

`missing_response`
Either `"error"` or `"omit"`.

`adaptive_quad_points`
Optional vector of at least two distinct integer orders >= 3, for example `c(31, 61)`, used only by `score_mfrm_calibration()`. Adds an unrounded `quadrature_review` comparing stored-grid results with mode/curvature-adapted integration for each Person. Inspect both fixed/adaptive differences and changes between adaptive orders. This does not change the artifact or replace the reported scores/intervals; adaptive results condition on the same point calibration. For GPCM or an explicitly supplied scoring prior, reference orders are added automatically and the comparison governs numerical score acceptance. Only Persons with scored responses have numerical review rows. `Status = "computed"` means the calculation finished, not that accuracy is certified; an unavailable row retains the reason in `Detail`.

`scoring_prior` Optional `list(mean = ..., sd = ...)` for `score_mfrm_calibration()`. `NULL` retains the artifact's normal prior. Values use the unchanged calibration scale and must be finite with positive SD and representable positive variance. A supplied prior automatically receives numerical scoring checks and does not modify the frozen artifact. Legacy artifacts with discrete grid-endpoint intervals require re-extraction before using this option. The prior is common to all scored persons; covariate-dependent overrides are not supported. Two-family GPCM format 6 requires `scoring_prior = NULL` and retains the fixed N(0,1) prior.

`cut_scores` Optional named, strictly increasing finite ability cuts for `score_mfrm_calibration()`. Ordinary RSM/PCM with unit weights and continuous posterior intervals only. Returns a `decisions` table of probabilities above cuts and in successive bands on the calibration scale. Calibration, prior and cuts are held fixed; this excludes their estimation uncertainty. Unscored Persons remain unavailable. Cut names do not establish CEFR alignment. See `predict_mfrm_units()` for interpretation and reporting.

Details

The portable workflow supports one observed score scale, one latent dimension and known non-Person facet levels. RSM/PCM MML retain stored anchors and two-way facet interactions in file format 1. One-family GPCM MML uses file format 2, retaining both step and slope owners, positive geometric-mean-one relative slopes and the estimated population mean and SD. The slope multiplies the entire adjacent-category predictor. Portable GPCM requires unit weights and no anchors or interactions. One-family MML uses an estimated intercept-only normal population. Shared-owner GPCM JML uses file format 4 and a post-hoc N(0,1) reference prior; its distinct checks are

described below. Experimental two-family GPCM MML uses file format 6, with two ordered slope owners, second-owner steps and the fixed N(0,1) prior. The first owner's locations and log slopes are centered; the second owner's locations and slopes remain free. Both fitting engines retain their fixed or adaptive scoring algorithm. Source identity/convergence and finer-integration checks must pass; review-only sources cannot be exported for frozen scoring. No separate `quadrature_review` is accepted for this route. The file contains component slopes and observed facet combinations, without training responses, Person estimates or an executable fit. Known levels may form new combinations; `row_dispositions$ObservedContext` identifies whether each pair occurred in calibration. New combinations are model-based, not empirically validated. Prior overrides and `event_id` extensions are unsupported; repeated Person- facet cells are rejected. Facet names must not collide with scoring/disposition columns (`Person`, `Score`, `Weight`, `InputRow`, `EventId`, `Disposition`, `ReasonCode`, `CalibrationId`, `ObservedContext`).

For format 6, use the same extract, validate, freeze, save, load and score functions as below. Each new batch must pass its own integration checks; validation and freezing do not establish sampling coverage, global identification, boundary absence or population transport. Posterior intervals exclude calibration uncertainty. Older readers refuse format 6; existing formats 1–5 keep their original interpretation. RSM/PCM JML uses file format 3, with unit weights, no anchors or interactions, a finite identified source and a post-hoc N(0,1) reference prior. This prior is not estimated by JML. Estimated-population RSM/PCM and latent regression use fitted-object routes; other structures follow their documented capabilities. See `mfrm_calibration_capabilities()`.

One-family GPCM MML extraction freshly evaluates the native fit's local likelihood, gradient, unregularized information and source-integration stability using the same conditional source checks as `predict_mfrm_units()`. A failed or unresolved source cannot be frozen by selecting a review policy. The artifact retains the passing extraction decision, numerical comparison, and original global boundary/identification states. Reading it validates these records against its stored identity; it does not refit the source or claim a global maximum, valid population transport or calibration intervals. Every GPCM scoring call also checks EAP/SD against two adaptive reference orders under the actual scoring prior. Failed score integration stops with `SCORING_INTEGRATION_FAILED`; extract with more `scoring_quad_points` before retrying. Endpoint and sparse-pattern review labels remain separate from numerical accuracy. File format 1 retains its original default semantics.

For RSM/PCM MML extraction, run `mml_quadrature_sensitivity()` on user-selected grids and inspect its continuous differences. Pass the exact highest-grid fit in that object together with `quadrature_review`. The retained fits must use their recorded grid sizes and the same data, score map and fitting settings, including anchors and integration mode, and must all have converged. This procedural requirement does not declare the fit numerically stable: the package does not choose the application-specific tolerance or decide whether more grids are needed. Archive the review separately when it is part of the audit trail, because response-linked fits are deliberately not embedded in the portable artifact. GPCM uses the fresh source checks above and does not require an additional refitting study. If supplied, `quadrature_review` is still validated.

RSM/PCM JML extraction requires current passing input, identification, category, boundary and numerical checks from the fitted-object scoring policy. It freshly evaluates the joint likelihood and gradient: the stored objective must agree within 1e-6 and the maximum absolute gradient must be <= 1e-4. These are numerical source criteria, not interval-coverage or bias guarantees. Boundary or unresolved sources are refused; there is no review override. No MML quadrature review is used because JML does not integrate over Persons during calibration. Every new JML score batch receives adaptive-reference EAP/SD checks under the actual scoring prior. Increase `scoring_quad_points` on extraction if the scoring grid is inadequate. Source checks and the prior origin persist through

saving, loading and score summaries; replay checks the stored evidence without re-estimating the training model.

Experimental corrected GPCM JML uses file format 5 and its own source checks. Extraction reconstructs the adjusted equation from the observed ratings and checks the saved root, full equation Jacobian and parameter tables. The maximum mean-equation residual must be <= 1e-7 and the remaining Newton step <= 1e-5. These are numerical checks, not a global uniqueness or bias guarantee. An unavailable covariance does not erase a valid calibration. The artifact records correction order, assignment-sampling assumption, local-root status and numerical checks without training responses or Person estimates. It is never labelled an ordinary JML likelihood maximum. Its EAP and continuous posterior intervals use a separate normal scoring prior; they exclude calibration uncertainty and do not supply corrected Person ML/WLE or structural confidence intervals. Older file formats retain their original checks. See `fit_mfrm()` for corrected estimation limits.

For shared-owner uncorrected GPCM JML, extraction checks the joint likelihood and gradient, and positive-definite, unregularized curvature over all free Person and structural parameters (reciprocal condition number > 1e-10). Recorded Person, additive or slope boundary certificates prevent automatic scoring; finite optimizer traces cannot override them. Detected identification failure, inadequate categories, failed numerical checks and unsupported structures also prevent extraction. Otherwise, global identification/boundary audits labelled as incomplete stay incomplete in the artifact. This qualifies conditional EAP only, not global optimality or formal JML slope inference. Source curvature currently uses a dense joint matrix, so its memory cost grows quadratically with the number of free parameters, including Persons. Frozen-artifact scoring does not repeat that calculation or use training data. The objective, gradient and condition-number cutoffs are package numerical criteria, not statistical thresholds established by the scoring literature. A refused source does not imply that all its structural parameters are inestimable: portable scoring does not currently qualify boundary-profile calibrations with infinite training-Person estimates.

Posterior EAP estimates, posterior standard deviations, and intervals are conditional on the frozen point calibration and the scoring prior. For GPCM MML, the prior mean and SD estimated during calibration are frozen; a new cohort does not reestimate or replace them. Use `scoring_prior` to examine another shared normal prior without changing the artifact. The original and scoring prior identities remain separate in estimates and settings, including summaries, plot data and exports. A supplied prior is an analyst assumption, not a population estimate. Intervals exclude both population-parameter and calibration-parameter uncertainty. Loading validates structure and semantic consistency, but does not authenticate an artifact from an untrusted source. New v2 scoring algorithms invert the continuous posterior CDF for equal-tail intervals. EAP and SD retain the stored quadrature rule. Saved v1 artifacts preserve their discrete grid interval endpoints and carry a note that the continuous posterior mass can differ from the requested interval level.

Value

`extract_mfrm_calibration()`, `validate_mfrm_calibration()`, `freeze_mfrm_calibration()`, `supersede_mfrm_calibration()`, `retire_mfrm_calibration()`, and `load_mfrm_calibration()` return an `mfrm_calibration`. `review_mfrm_calibration()` returns a data frame of structured refusals, with zero rows for an acceptable object. `save_mfrm_calibration()` invisibly returns the normalized path. `score_mfrm_calibration()` returns an `mfrm_calibration_score` containing estimates plus row and Person dispositions and scoring identities. With `cut_scores`, it also retains `decisions`: conditional event probabilities with boundaries, labels, actual priors and availability. When requested and scored rows exist, `quadrature_review` contains the fixed/adaptive comparison; `summary()` preserves it and adds a compact `quadrature_overview`.

Anchors, centering and score uncertainty

New exports distinguish whole-facet centering, legacy remainder centering and uncentered facets in `constraints$identification`. Direct values and group means are retained in `constraints$anchors`; validation checks the explicit facet maps against the stored coordinates and free dimensions. For example, if four raters are centered as a whole and the first three have mean 0.10, the fourth is fixed at -0.30. Freezing retains all four calibrated values, including the estimates of the free group members.

Scoring uses those expanded values without centering or estimating them again. A returned `SD` is a Person posterior SD conditional on the entire frozen calibration and scoring prior. It is not a rater SE and excludes uncertainty in both the estimated calibration and supplied anchors. Use `mfrm_facet_intervals()` for supported conditional facet intervals. With an anchored fitted model, `predict_mfrm_units()` accepts supplied response rows; the narrower `score_mfrm_persons()` entry excludes anchors.

Loading an older artifact preserves its values and original constraint labels. Older generic labels do not fully identify the centering convention; consult the source fit or re-extract it to obtain explicit labels. Changing a label cannot convert a calibration to another scale.

Statistical basis and external comparisons

Bock and Mislevy (1982, pp. 432-433) describe posterior-mean and posterior-SD scoring from fixed response functions and a specified prior, including multiple-category responses. This supports the scoring layer; it does not validate JML calibration bias, the default reference prior for a new cohort, or the numerical tolerances used to check a fitted calibration. Their adaptive testing simulations do not establish coverage for sparse many-facet designs. Muraki (1992) supplies the GPCM model basis and an EM estimation method, not evidence that many-facet JML intervals have accurate coverage.

External checks of fixed probabilities, likelihoods and EAP/SD must align the response functions and prior. They do not compare free JML estimation. ConQuest documents that JML cannot estimate item scores (the free scores used for GPCM discrimination). TAM's `tam.jml()` uses a supplied loading array and defaults to extreme-score adjustment and item-bias correction; its default output is not the same estimator as unadjusted JML. Do not infer agreement of estimators from agreement of fixed-calibration scores.

Choose what to save

Save a fitted analysis or `mfrm_results()` with `saveRDS()` when you need to revisit its diagnostics and reports. A portable `mfrm_calibration` has a different purpose: scoring new response rows without the training fit or training responses. Use `save_mfrm_calibration()` and `load_mfrm_calibration()` for that artifact. Save an already computed `mfrm_calibration_score` with `saveRDS()` to reproduce its tables and figures without scoring the responses again. A CSV is a table, not a replacement for any of these complete objects.

Validation checks the saved parameters, scoring settings and model checks; freezing marks that calibration as ready for scoring without changing it. These steps do not establish model fit, lack of bias or suitability for another population. New Persons may use known calibrated facet levels. An unknown rater, task or score category requires an appropriate calibration workflow; assigning an existing level's name is not a way to extend the calibration. Missing responses are not zero scores. With `missing_response = "omit"`, inspect row dispositions and Persons who could not be scored.

Older saved objects are checked against their recorded format and evidence. Inspect `review_mfrm_calibration()` if a file has missing or inconsistent records. Do not change saved check results to force acceptance. If checks of the original fitted model are missing, re-extract from that saved model if it passes the current checks; when only the score records are incomplete, score again from a valid frozen artifact. Neither action means that model refitting is automatically necessary.

Conditional ability decisions

Supply `cut_scores` on the calibration's ability scale to obtain `decisions`, with one row per Person and requested event. `Type = "above_cut"` gives $P(\theta > c \mid y, \widehat{\psi})$; `Type = "ability_band"` gives the probability between successive cuts, including the two outer bands. `Lower < ability <= Upper` defines every row. `Label` uses the supplied cut names (or `Cut1`, ...) and `Band1`, ... in increasing order. These events overlap: only the ability-band probabilities sum to one.

`PriorMean` and `PriorSD` record the actual Person-specific normal prior; `Status` and `Reason` retain scoring availability. `available_conditional` means the applicable numerical checks passed, not that the model or the cut's substantive meaning has been validated. `review_only` requires inspection before interpretation. Source-roster scoring may return `prior_only` for missing-only Persons; portable scoring instead retains them as `unavailable` with `NA` probabilities. Fitted-object scoring omits Persons with no retained rows; inspect its row and population reviews. A Person absent from the input is never invented.

Probabilities integrate the continuous posterior, rather than counting `sample_mfrm_plausible_values()` draws or applying a normal approximation to EAP/SD. They condition on point calibration, population/prior parameters and supplied cuts, as recorded in `Uncertainty`. They do not propagate estimation error, compare two Persons or predict passing a future test. CEFR labels require externally justified cuts on this same scale. Very small tails have absolute, not relative, numerical accuracy; a returned zero or one is not a claim of impossibility or certainty.

Review the scoring population as well as the calibration. An estimated normal prior is treated as known during scoring; its sampling uncertainty can matter especially with a small calibration cohort. A larger calibration cohort can improve parameter precision without correcting a prior that mismatches a new population's mean, spread or shape. Passing numerical checks does not establish that probabilities are accurate for that new population. A fixed reference prior is not a universal remedy: justify it for the target population on the calibration scale. If a prior override is supported, comparing externally justified alternatives is a sensitivity analysis, not evidence that the alternative giving the preferred classification is valid. Population-averaged accuracy also does not guarantee accuracy near each cut, within every subgroup or for each individual Person.

Use `score_mfrm_persons()` for eligible ordinary RSM MML source Persons, `predict_mfrm_units()` for ordinary RSM/PCM MML or JML fitted-object scoring, and `score_mfrm_calibration()` for eligible continuous-posterior RSM/PCM artifacts. Each route retains its existing source and population restrictions. Unit weights are required; GPCM, shared-rater and testlet cut probabilities are not supported. JML uses a separate scoring prior, not a population estimated by JML. `NULL` leaves the existing output unchanged.

`summary(x)$decisions` and `apa_table()` with `which = "decisions"` reuse saved probabilities without rounding them. Fitted-object and portable score batches also support `build_summary_table_bundle()` and `export_summary_appendix()`. Save the complete result with `saveRDS()` to retain cuts, priors and review states for later reporting. Existing score plots display EAPs and intervals, not these event probabilities.

References

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, 6(4), 431-444. doi:10.1177/014662168200600405.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159-176. doi:10.1177/014662169201600206.

TAM joint maximum likelihood documentation and ACER ConQuest command reference.

See Also

`mml_quadrature_sensitivity()`, mfrm_calibration_score_methods for an executable scoring example, concise review and visualization of returned score batches; `predict_mfrm_units()`, `score_mfrm_persons()`, `apa_table()` and `export_summary_appendix()` for saved-score tables.

`mfrm_cluster` *Group mixed external features using Gower distances and PAM*

Description

Partition an entity-level feature table using Gower dissimilarities and partitioning around medoids (PAM) from the optional `cluster` package. Use `mfrm_cluster_pam()` for this method. `mfrm_cluster()` is retained as an identical compatibility alias; it does not select an algorithm for you.

Usage

```
mfrm_cluster_pam(x, k, weights = NULL, missing = c("error", "omit"))

mfrm_cluster(x, k, weights = NULL, missing = c("error", "omit"))

## S3 method for class 'mfrm_clusters'
print(x, ...)

## S3 method for class 'mfrm_clusters'
summary(object, ...)
```

Arguments

`x`	An object returned by `mfrm_features()`.
`k`	Number of groups, an integer from 2 to one less than the number of included entities, and no greater than their number of distinct profiles. The number is chosen by the user, not optimized automatically.
`weights`	Optional named, strictly positive finite numeric vector with one weight per selected feature. Defaults to equal weights. Names, not vector order, identify features. Remove a feature to exclude it.
`missing`	Either `"error"` (default) or `"omit"`. The latter excludes incomplete entities explicitly, retaining their IDs, reasons, and missing group membership in the result. No values are imputed.

...	Reserved for method compatibility.
object	An object returned by mfrm_cluster_pam().

Details

Numeric differences are divided by the feature range among included entities; nominal features use match/mismatch and logical features use symmetric binary differences. Ordered factors use equally spaced integer codes in their declared level order, divided by the observed code range. Unused intermediate levels retain their spacing. The Gower dissimilarity is the weighted mean of these feature contributions, dividing by the sum of the selected weights. Weights apply to features, not entities. Numeric 0/1 features are treated as numeric, not asymmetric presence/absence. Every selected feature must vary among included entities. Gower scaling, feature types, weights, omission policy, and numeric ranges are retained. Numeric ranges and relative weights must be representable without overflow or underflow; rescale features or revise extreme weight ratios if refused.

PAM uses deterministic BUILD/SWAP initialization; tied distances may admit alternative partitions. Group numbers are arbitrary labels. These are exploratory groups, not latent classes, ability estimates, or assessments of rater quality. No inference, uncertainty propagation, new-entity classification, or resampling stability is provided. Omission may change both the sample and numeric ranges and does not correct missing-data bias.

Pairwise distances require quadratic memory. This interface is limited to 5,000 included entities. It does not silently sample larger inputs. This input limit does not guarantee low memory use or acceptable run time.

For feature selection, missingness review, and multiple-imputation examples, see `vignette("mfrmr-external-features", package = "mfrmr")`.

Value

An `mfrm_clusters` object containing `membership` (ID, Cluster, Medoid, Silhouette), `cluster_summary`, numeric and categorical `profiles`, `medoids`, the reviewed `feature_data`, and `settings`. Silhouette values describe within-sample separation, not membership probabilities or resampling stability. `summary()` returns the cluster-size/silhouette table. Numeric profiles report the mean, median and sample SD in the original feature units. SD is `NA` for a one-entity group and zero for identical values in a group with at least two entities. A singleton's silhouette is zero by convention; it is not evidence of group separation.

Reading clustering metrics

For entity i, a(i) is its mean dissimilarity to other members of its own group; b(i) is the smallest mean dissimilarity to any other group. Silhouette = (b - a) / max(a, b), between -1 and 1. Values near 1 indicate compact, separated placement in this fitted geometry; near zero indicates a boundary; negative values indicate closer average proximity to another group. A singleton receives 0 by convention; zero does not establish overlap. A zero denominator also receives 0. Overall means weight entities equally; cluster means use that cluster's included entities. Excluded IDs have NA. A silhouette is neither a membership probability nor evidence of stability.

Gower distance is unitless in `[0, 1]` here: numeric absolute differences divided by included-sample ranges, nominal/binary mismatches, and ordinal rank differences are combined using the specified feature weights. The PAM medoid is an actual included entity minimizing within-group total dissimilarity; it is not a feature-wise mean. Profile Mean/Median/SD retain original units; categorical

Proportion uses included group size as denominator. A large group or a high silhouette is not a better-performing group. Compare partitions under a fixed feature/distance definition; changing the sample, weights or scaling changes what these numbers measure. See `cluster::silhouette()` and Rousseeuw (1987), doi:10.1016/03770427(87)901257, for the silhouette definition.

See Also

`mfrm_features()`, `mfrm_cluster_imputed()`, `mfrm_cluster_compare()`, `mfrm_cluster_hierarchical()`, `plot.mfrm_clusters()`, `cluster::daisy()`, `cluster::pam()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_cluster", run.dontrun = TRUE)
## Not run:
if (requireNamespace("cluster", quietly = TRUE)) {
  # Fictional raters; R2 has unrecorded experience, not zero years.
  raters <- data.frame(Rater = paste0("R", 1:6),
    ExperienceYears = c(1, NA, 3, 12, 13, 14),
    Specialty = rep(c("Language", "Science"), each = 3))
  features <- mfrm_features(raters, "Rater", c("ExperienceYears", "Specialty"))
  # The default stops on missing features. Here omission is explicit.
  groups <- mfrm_cluster_pam(features, k = 2,
    weights = c(ExperienceYears = 2, Specialty = 1), missing = "omit")
  groups$membership  # R2 remains present with unavailable membership.
  summary(groups)
  groups$profiles
  groups$medoids
}

## End(Not run)
```

---

mfrm_cluster_compare *Compare exploratory groups across settings and clustering methods*

Description

Compare existing external-feature clustering results without refitting. Review how group membership changes when the selected features, number of groups, feature weights, or clustering method changes, including paired comparisons across the same imputations.

Usage

```
mfrm_cluster_compare(analyses)

## S3 method for class 'mfrm_cluster_comparison'
print(x, ...)

## S3 method for class 'mfrm_cluster_comparison'
summary(object, ...)
```

Arguments

analyses	A named list of at least two results from mfrm_cluster_pam() and/or mfrm_cluster_hierarchical() or mfrm_cluster_kmeans(), or a named list of results from mfrm_cluster_imputed(). Do not mix the two result types. Names must be unique and nonblank. All results must use the same entity IDs and included entities. Selected features may differ; features with the same name must retain their original values and types. Row and feature order may differ. IDs must refer to the same entities, not, for example, persons and raters that happen to share ID labels. For imputed results, shared completed features must also match by imputation number. When feature selections differ, all results must retain the same mids object, including its data and settings; each retained completion is checked against it using the optional mice package. Reuse one fitted imputation model. Results saved in separate RDS files can be compared after reopening when their retained imputation data, settings and formula expressions are unchanged. Formula environments are not evaluated in this comparison; no completions are regenerated.
x, object	An object returned by mfrm_cluster_compare().
...	Reserved for method compatibility.

Details

Comparisons align entities by ID and do not compare numeric group labels directly. Zero changed fraction and an adjusted Rand index of one indicate identical partitions, allowing arbitrary renumbering of groups. The changed fraction can be small when many pairs are separated in both partitions. The adjusted Rand index corrects agreement against random partitions with fixed group sizes; it can be negative. Neither measure identifies the correct group count or establishes group validity.

Mean silhouette and group sizes help describe each partition. Silhouette values calculated with different features or weights use different distances; their maximum is not an automatic criterion for selecting features or weights. Review group profiles in the retained analyses alongside the comparison.

With multiple imputations, feature selection is applied to the same completed data on both sides, without refitting the imputation model. A feature omitted from clustering can remain an imputation predictor; this comparison does not evaluate its removal from the imputation model. Every imputation is retained. Summary means and ranges describe sensitivity to settings across those imputations; they are not Rubin-pooled estimates, confidence intervals, or sampling stability. Conflicting shared feature values or different inclusion masks cause an error rather than a silent intersection of entities or imputations. No preferred setting, consensus partition, or hypothesis test is returned.

For an executable rater-attribute example that pairs multiple imputations across settings, see vignette("mfrmr-external-fe
package = "mfrmr").

Value

An mfrm_cluster_comparison object containing:

- analysis_summary: method, distance, fitted space, scaling, retained component count, linkage (unavailable for PAM and k-means), group count, selected feature count (Features), included/excluded entity counts, smallest/largest group sizes, and mean silhouette for each analysis and imputation. Imputation is NA for ordinary clustering results.

- `weights`: selected features and their supplied weights for each analysis. Unselected features have no row; they are not zero-weight inputs.
- `comparisons`: all pairs of analyses, with one row per imputation. `Pairs` counts unordered pairs of included entities, excluding self-pairs. `SplitPairs` were together in `First` and apart in `Second`; `JoinedPairs` were apart in `First` and together in `Second`. `ChangedFraction` is their sum divided by `Pairs`. `AdjustedRand` is the adjusted Rand index.
- `comparison_summary`: number of compared partitions, mean/minimum/ maximum changed fraction, and mean adjusted Rand index for each analysis pair. `summary()` returns this table.
- `analyses`: the original results, including omitted IDs and all imputation-specific partitions, profiles, and settings.

References

Hubert, L. and Arabie, P. (1985). Comparing partitions. Journal of Classification, 2, 193–218. doi:10.1007/BF01908075.

See Also

`mfrm_features()`, `mfrm_cluster_pam()`, `mfrm_cluster_imputed()`, `mfrm_cluster_hierarchical()`

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  # Fictional attributes; experience is measured in completed years.
  raters <- data.frame(Rater = paste0("R", 1:8),
    ExperienceYears = c(1, 2, 3, 4, 11, 12, 13, 14),
    Specialty = rep(c("Language", "Science"), 4))
  features <- mfrm_features(raters, "Rater", c("ExperienceYears", "Specialty"))
  fits <- list(
    TwoGroups = mfrm_cluster_pam(features, k = 2),
    ThreeGroups = mfrm_cluster_pam(features, k = 3),
    ExperienceWeighted = mfrm_cluster_pam(features, k = 2,
      weights = c(ExperienceYears = 3, Specialty = 1)))
  comparison <- mfrm_cluster_compare(fits)
  summary(comparison)
  comparison$analysis_summary
  comparison$comparisons
  # Hold the entities and group count fixed while changing the feature set.
  experience <- mfrm_features(raters, "Rater", "ExperienceYears")
  feature_comparison <- mfrm_cluster_compare(list(
    BothFeatures = fits$TwoGroups,
    ExperienceOnly = mfrm_cluster_pam(experience, k = 2)))
  feature_comparison$analysis_summary
  feature_comparison$weights
}
```

mfrm_cluster_hierarchical *Hierarchical grouping of external features*

Description

Build an agglomerative hierarchy from weighted Gower dissimilarities, cut it into a chosen number of groups, and inspect the retained dendrogram.

Usage

```
mfrm_cluster_hierarchical(
  x,
  k,
  weights = NULL,
  missing = c("error", "omit"),
  linkage = c("average", "complete")
)

## S3 method for class 'mfrm_hierarchical_clusters'
plot(
  x,
  type = c("dendrogram", "silhouette", "profile"),
  feature = NULL,
  labels = NULL,
  draw = TRUE,
  preset = "standard",
  ...
)
```

Arguments

x	For clustering, a table reviewed with mfrm_features(). For plotting, an mfrm_hierarchical_clusters result.
k	Number of groups, an integer from 2 to one less than the number of included entities, and no greater than their number of distinct profiles. The number is chosen by the user, not optimized automatically.
weights	Optional named, strictly positive finite numeric vector with one weight per selected feature. Defaults to equal weights. Names, not vector order, identify features. Remove a feature to exclude it.
missing	Either "error" (default) or "omit". The latter excludes incomplete entities explicitly, retaining their IDs, reasons, and missing group membership in the result. No values are imputed.
linkage	"average" (default, UPGMA) or "complete". Average linkage uses the mean dissimilarity over all cross-group pairs; complete linkage uses their maximum. Neither is selected automatically from the data.

type	"dendrogram" (default), "silhouette", or "profile".
feature	A single selected feature name, required for type = "profile". Numeric features show means and medians in their original units; categorical features show within-group proportions in the original level order (including unused factor levels).
labels	Whether to display entity IDs on dendrograms and silhouettes. The default labels at most 50 included entities. Hiding labels does not remove entities. Profile plots always label groups and levels.
draw	Draw the plot when TRUE; FALSE only returns plotted values.
preset	Plot style: "standard", "publication", "compact", or "monochrome".
...	Reserved for future use; additional arguments are rejected.

Details

Uses stats::hclust() followed by stats::cutree() at the requested k. Feature types, scaling, weights, missingness handling, silhouette definition, and the 5,000-included-entity limit follow mfrm_cluster_pam(). No distance transformation or automatic sampling is performed. This limit is not a memory or speed guarantee.

Only average and complete linkage are supported. Ward's minimum-variance criterion requires an appropriate Euclidean geometry; this mixed-feature Gower interface does not define a Euclidean conversion or a Ward analysis. Clustering MFRM bias estimates is a separate methodological question from grouping the external attributes accepted here. Measurement uncertainty is not propagated.

The tree is fitted independently of PAM. Its merge heights describe the selected linkage on Gower dissimilarities, not significance or branch support. Tied distances can produce alternative hierarchies and input order can affect their resolution. A cut at k uses merge order even when heights tie, so a horizontal height threshold need not uniquely identify that cut.

The default plot draws this stored tree, with boxes marking the stored k groups. Labels are shown for at most 50 included entities by default; hiding labels does not sample or remove entities. Excluded entities have no leaves but remain in the result and plot data. Silhouette and feature-profile views reuse plot.mfrm_clusters(). Plots do not refit or choose groups. as_ggplot() converts the stored dendrogram without refitting. The default and component = "tree" retain the full tree and group boxes. Leaf order, heights, label settings and excluded IDs are preserved. Dashed boxes differ from the solid tree branches even in monochrome. Box widths and physical text sizes can differ from base graphics. Tied heights retain the merge-order partition; boxes do not imply a unique horizontal height cut. Use ggplot2::labs(title = NULL, subtitle = NULL, caption = NULL) to remove headings and annotations, and plot_data() to inspect the retained source evidence. Silhouette/profile conversions use the same dedicated summary renderers as plot.mfrm_clusters().

Use mfrm_cluster_compare() to compare this partition with PAM or another linkage on the same data. For multiple imputations, use mfrm_cluster_imputed(..., method = "hierarchical", linkage = "average"). Each completion retains its own tree; no pooled tree or branch support is estimated. See vignette("mfrmr-external-features", package = "mfrmr").

Value

An mfrm_hierarchical_clusters object inheriting from mfrm_clusters. It retains the hclust object in tree, ID-aligned membership (ID, Cluster, Silhouette), cluster_summary, profiles,

feature_data, and settings including the linkage. Hierarchical clustering does not select medoids; medoids is NULL and membership has no Medoid column. summary() returns the size/silhouette table. plot() invisibly returns mfrm_plot_data; dendrogram data include the tree, leaf order, memberships, group count, and excluded IDs.

Reading clustering metrics

For entity i, a(i) is its mean dissimilarity to other members of its own group; b(i) is the smallest mean dissimilarity to any other group. Silhouette = (b - a) / max(a, b), between -1 and 1. Values near 1 indicate compact, separated placement in this fitted geometry; near zero indicates a boundary; negative values indicate closer average proximity to another group. A singleton receives 0 by convention; zero does not establish overlap. A zero denominator also receives 0. Overall means weight entities equally; cluster means use that cluster's included entities. Excluded IDs have NA. A silhouette is neither a membership probability nor evidence of stability.

Gower distance is unitless in [0, 1] here: numeric absolute differences divided by included-sample ranges, nominal/binary mismatches, and ordinal rank differences are combined using the specified feature weights. The PAM medoid is an actual included entity minimizing within-group total dissimilarity; it is not a feature-wise mean. Profile Mean/Median/SD retain original units; categorical Proportion uses included group size as denominator. A large group or a high silhouette is not a better-performing group. Compare partitions under a fixed feature/distance definition; changing the sample, weights or scaling changes what these numbers measure. See cluster::silhouette() and Rousseeuw (1987), doi:10.1016/03770427(87)901257, for the silhouette definition.

Session plot defaults

Set options(mfrmr.plot_preset = "publication") to choose a session default for plotting functions that expose the common preset argument. The supported values are "standard", "publication", "compact" and "monochrome". Precedence is an explicit call argument, then the session option, then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of plot() for report bundles use the same option through .... Plots without a common preset argument, including extended-model plots with their own palette controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with as_ggplot() uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply preset explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with old <- options(mfrmr.plot_preset = "monochrome") followed by options(old). Use options(mfrmr.plot_preset = NULL) to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

References

Murtagh, F. and Legendre, P. (2014). Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? Journal of Classification, 31, 274–295. doi:10.1007/s003570149161z.

See Also

mfrm_cluster_pam(), mfrm_cluster_imputed(), mfrm_cluster_compare()

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  # Fictional rater backgrounds; R7 has unrecorded experience.
  raters <- data.frame(Rater = paste0("R", 1:7),
    ExperienceYears = c(1, 2, 3, 12, 13, 14, NA),
    Specialty = factor(c(rep("Language", 3), rep("Science", 4))))
  features <- mfrm_features(raters, "Rater", c("ExperienceYears", "Specialty"))
  hierarchy <- mfrm_cluster_hierarchical(features, k = 2, missing = "omit")
  plot(hierarchy)
  plot(hierarchy, type = "silhouette")
  plot(hierarchy, type = "profile", feature = "ExperienceYears")
  comparison <- mfrm_cluster_compare(list(
    PAM = mfrm_cluster_pam(features, k = 2, missing = "omit"),
    Average = hierarchy,
    Complete = mfrm_cluster_hierarchical(features, k = 2,
      linkage = "complete", missing = "omit")))
  comparison$analysis_summary
  summary(comparison)
}
```

---

mfrm_cluster_imputed *Compare exploratory groups across external-feature imputations*

Description

Apply the same external-feature clustering analysis to each completed data set from mice, retaining all analyses and the fraction of imputations in which each pair of entities belongs to the same group.

Usage

```
mfrm_cluster_imputed(
  x,
  imputed,
  impute,
  k,
  weights = NULL,
  missing = c("error", "omit"),
  method = c("pam", "hierarchical", "kmeans"),
```

```
  linkage = NULL,
  components = NULL,
  scale = TRUE,
  nstart = 25,
  iter.max = 100,
  seed = 1,
  silhouette = TRUE
)

## S3 method for class 'mfrm_imputed_clusters'
print(x, ...)

## S3 method for class 'mfrm_imputed_clusters'
summary(object, ...)
```

Arguments

x	An original feature table reviewed with mfrm_features().
imputed	A mids object from the optional mice package, containing the identifier, selected features, and at least two imputations. Its original data must match x by ID. Auxiliary variables may be included in the imputation model without becoming clustering features.
impute	A data frame with ID and Feature columns explicitly listing the missing cells to impute. For example, select appropriate rows from x$missing after reviewing their reasons. Extra columns are ignored. The selected-feature entries in imputed$where must match this selection; other missing cells must remain missing in every completed data set. If all selected features are already complete, supply the empty x$missing table. This retains one partition per imputation so the result can be compared with other feature selections from the same model.
k, weights	As in mfrm_cluster_pam(). The same choices apply to every imputation.
missing	Either "error" (default) or "omit", applied to missing features remaining after imputation. Explicit omission retains excluded IDs with unavailable memberships and pairwise proportions.
method	"pam" (default), "hierarchical", or "kmeans". The same method is used for every completion. Hierarchical analyses retain separate trees.
linkage	For method = "hierarchical", "average" (the default when NULL) or "complete"; see mfrm_cluster_hierarchical(). Must be NULL for PAM and k-means, which have no linkage.
components	For k-means, NULL uses numeric features directly. An integer fits PCA separately in each completion and uses that many leading components. No whitening, pooled PCA basis or component-sign matching is performed.
scale, nstart, iter.max, seed, silhouette	For method = "kmeans", as in mfrm_cluster_kmeans(). These arguments must not be supplied for other methods. The same seed and settings are used for every completion; sensitivity still includes possible changes in local optima. Review alternative seeds separately.

...	Reserved for method compatibility.
object	An object returned by mfrm_cluster_imputed().

Details

Fit and review the imputation model using mice::mice() before calling this function. Choose methods, predictors (including relevant auxiliary variables), iteration count, and number of imputations for the intended analysis. Exclude the identifier from imputation and prediction. Inspect model diagnostics, including loggedEvents and chain behavior; this adapter checks data consistency, not convergence or model adequacy. The original model and its diagnostics remain available in the result.

Missing reasons do not identify a statistical missing-data mechanism. Do not impute structurally undefined attributes such as "not applicable". Specify eligible cells through where when fitting mice and list the same cells in impute. An incomplete predictor that is not imputed can prevent imputation of other variables; configure predictors accordingly. Assumptions about nonresponse require substantive justification; this function does not correct bias automatically or impute rating responses. Nonresponse related to unobserved values needs separate sensitivity assumptions; observed-data checks cannot establish their adequacy.

IDs, observed feature values, feature types, and factor levels/order must be preserved. Numeric 0/1 completions of originally logical features are restored to logical values. Row order is restored by ID. Every selected cell must be completed in every imputation. Any invalid completion or failed clustering stops the comparison with its imputation number; no failures are discarded. Remaining missingness, and thus the included sample, is the same across imputations. Gower numeric ranges are recalculated in each completed sample; differences may reflect changes in both feature values and scaling. Numeric k-means likewise recalculates centers, standard deviations and, when requested, the PCA basis in each completion. Only partitions are compared; component scores/loadings and group labels are not averaged or Rubin-pooled. Hierarchical trees belong to individual completions; no pooled tree, consensus hierarchy, or branch-support estimate is returned.

Co-membership proportions are invariant to arbitrary group numbering. They describe sensitivity to the supplied imputations, conditional on the imputation model, features, weights, and group count. They are not posterior membership probabilities, sampling stability, or Rubin-pooled estimates. No consensus partition, confidence interval, or automatic group selection is produced. Both the full ID-indexed matrix and pairwise distances require quadratic memory, so this comparison is limited to 5,000 total entities. This input limit is not a memory or run-time guarantee. Retaining all completed analyses also increases memory use with the imputation count.

The short example below illustrates the interface using mice::nhanes2. For a complete example with fictional rater attributes, explicit missingness reasons, imputation diagnostics, and comparisons of group counts and weights, see vignette("mfrmr-external-features", package = "mfrmr").

Value

An mfrm_imputed_clusters object containing analyses (one mfrm_clusters object per imputation), co_membership (a symmetric matrix indexed by ID), analysis_summary, original feature_data, imputed_cells with original reasons, the full imputation_model, and settings. Every available matrix entry uses all settings$imputations analyses as its denominator; pairs involving excluded entities are NA, including their diagonal entries. summary() returns per-imputation counts and mean silhouette widths. Numeric ranges are retained in each analysis. Hierarchical partitions inherit from mfrm_clusters and retain their trees. Method and linkage are retained in settings.

See Also

mfrm_features(), mfrm_cluster_pam(), mfrm_cluster_compare(), mfrm_cluster_hierarchical(), plot.mfrm_clusters(), mice::mice(), mice::complete()

Examples

```
# Optional integration example (loads additional modelling or rendering packages).
# example("mfrm_cluster_imputed", run.dontrun = TRUE)
## Not run:
if (requireNamespace("mice", quietly = TRUE) &&
    requireNamespace("cluster", quietly = TRUE)) {
  attributes <- mice::nhanes2
  attributes$Person <- paste0("P", seq_len(nrow(attributes)))
  review <- mfrm_features(attributes, "Person", c("bmi", "chl"))
  # Here all missing selected attributes are assumed eligible after review.
  cells <- review$missing
  method <- mice::make.method(attributes)
  method["Person"] <- ""
  predictors <- mice::make.predictorMatrix(attributes)
  predictors[, "Person"] <- 0
  predictors["Person", ] <- 0
  # Small settings illustrate the API, not an adequacy recommendation.
  model <- mice::mice(attributes, m = 3, maxit = 2, method = method,
    predictorMatrix = predictors, seed = 42, printFlag = FALSE)
  result <- mfrm_cluster_imputed(review, model, cells, k = 2)
  summary(result)
  result$co_membership[1:4, 1:4]
  result$imputation_model$loggedEvents
}

## End(Not run)
```

---

mfrm_cluster_kmeans *K-means groups from numeric features or retained principal components*

Description

Divide people, raters or tasks into k groups with similar numeric attributes. The algorithm minimizes squared distances within groups, using the numeric features or retained principal components supplied by you. The groups are descriptive; they do not measure ability or rater quality.

Usage

```
mfrm_cluster_kmeans(
  x,
  k,
  weights = NULL,
  missing = c("error", "omit"),
```

```
  scale = TRUE,
  nstart = 25,
  iter.max = 100,
  seed = 1,
  silhouette = TRUE
)
```

Arguments

x	For `mfrm_pca()`: A table reviewed with `mfrm_features()`. For `mfrm_cluster_kmeans()`, a result from `mfrm_pca()` is also accepted. For `plot()`: A result from `mfrm_pca()`.
k	Number of groups, an integer from 2 to one less than the number of included entities, and no greater than their number of distinct profiles. The number is chosen by the user, not optimized automatically.
weights	Optional named, strictly positive finite numeric vector with one weight per selected feature. Defaults to equal weights. Names, not vector order, identify features. Remove a feature to exclude it.
missing	Either `"error"` (default) or `"omit"`. The latter excludes incomplete entities explicitly, retaining their IDs, reasons, and missing group membership in the result. No values are imputed.
scale	Divide centered numeric features by their sample standard deviations (`TRUE`, default). With `FALSE`, distances retain the original feature units. Numeric coding must have a meaningful Euclidean interpretation; factors, ordered factors, characters and logical features are refused.
nstart	Number of random starts; default 25.
iter.max	Maximum iterations per start; default 100.
seed	Nonnegative integer seed. The default is 1. The caller's random number state is restored; the RNG kind is retained in `settings`.
silhouette	Compute Euclidean silhouettes (`TRUE`, default) using the optional `cluster` package. This requires pairwise distances and is limited to 5,000 included entities. Explicit `FALSE` avoids that quadratic allocation; silhouettes then remain unavailable, not zero. Other memory/time limits still depend on the dimensions and numerical workload.

Details

Uses `stats::kmeans()` with Hartigan-Wong updates and the stated random starts. Warnings, nonconvergence and nonfinite sums of squares stop the analysis; no failed run is silently presented as a usable result. Multiple starts reduce sensitivity to initialization without guaranteeing a global optimum. Review alternative seeds separately when needed. Labels are arbitrary and results can depend on input order, ties and RNG kind.

Numeric preprocessing matches `mfrm_pca()`. With a PCA input, its retained scores are used exactly: do not respecify `weights`, `scale` or `missing`. Full-rank PCA preserves the distance objective; truncation changes it. No component whitening is performed. The centers of a truncated PCA are coordinates in that subspace; `profiles` describe the original attributes. Categorical codes must not

be converted to numeric solely to pass this interface. Use `mfrm_cluster_pam()` for Gower/PAM mixed-feature groups.

Compare fitted partitions using `mfrm_cluster_compare()`. Silhouettes calculated in different geometries are not directly comparable evidence of which feature set, scale or component count is correct. Grouping does not propagate measurement uncertainty or establish rater quality.

Value

An `mfrm_clusters` object with memberships, original-unit profiles, cluster sizes and silhouettes, `centers` in the fitted space, per-cluster `withinss`, `totss`, `tot.withinss`, the input PCA when used, original feature data, transformation and settings. K-means has no medoids. Numeric profiles use original feature units and sample SDs: SD is `NA` for a one-entity group and zero for identical values in a larger group. A singleton's silhouette is zero by convention, not evidence of separation.

Reading clustering metrics

`withinss` is the sum of squared Euclidean distances from members to their cluster center; `tot.withinss` sums it across clusters. `totss` uses the grand center instead. The descriptive ratio 1 - tot.withinss/totss is the fraction of fitted-space scatter between groups, not predictive accuracy, latent reliability or a test of the chosen k. Increasing k usually improves this ratio even without meaningful groups. These sums have squared units of the transformed coordinates (SD-scaled/weighted features or retained PCs), not necessarily original feature units. Compare them only for the same rows and geometry. For silhouette a(i), b(i), range, singleton conventions and profile denominators, see the metric section of `mfrm_cluster_pam()`.

See Also

`mfrm_pca()`, `mfrm_cluster_pam()`, `mfrm_cluster_imputed()`, `mfrm_cluster_compare()`, `plot.mfrm_clusters()`, `stats::kmeans()`

Examples

```
raters <- data.frame(Rater = paste0("R", 1:8),
  ExperienceYears = c(1, 2, 3, 4, 10, 12, 13, 15),
  WorkshopHours = c(4, 12, 8, 20, 12, 28, 16, 32),
  AnnualRatings = c(80, 120, 90, 160, 280, 350, 300, 400))
features <- mfrm_features(raters, "Rater", names(raters)[-1])
direct <- mfrm_cluster_kmeans(features, 2, seed = 17, silhouette = FALSE)
reduced <- mfrm_cluster_kmeans(mfrm_pca(features, components = 2), 2,
  seed = 17, silhouette = FALSE)
summary(mfrm_cluster_compare(list(Features = direct, TwoPCs = reduced)))
direct$centers
direct$profiles$numeric
```

`mfrm_curve_intervals` *Uncertainty in GPCM category probabilities and information curves*

Description

Evaluate specified rating contexts at known ability values and propagate the full calibration covariance to category probabilities or per-rating Fisher information. This is uncertainty in a fitted curve, not a Person score interval.

Usage

```
mfrm_curve_intervals(
  fit,
  newdata,
  type = c("probability", "information"),
  method = c("model", "sandwich"),
  clusters = NULL,
  adjust = FALSE,
  level = 0.95,
  simultaneous = c("none", "bonferroni")
)

## S3 method for class 'mfrm_curve_intervals'
summary(object, ...)

## S3 method for class 'mfrm_curve_intervals'
print(x, ...)

## S3 method for class 'mfrm_curve_intervals'
plot(
  x,
  title = "GPCM curve uncertainty",
 subtitle = paste("Approximate calibration intervals; fixed ability", x$settings$method,
    paste0(100 * x$settings$level, "%"), if (x$settings$simultaneous == "none")
    "pointwise" else "Bonferroni grid points", sep = " | "),
  palette = NULL,
  draw = TRUE,
  caption = NULL,
  ...
)
```

Arguments

`fit`	A native GPCM MML fit. Two-family fits supply provisional point curves only; intervals and covariance adjustments are unavailable.
`newdata`	Data frame with each fitted non-Person facet and a numeric `Theta` column. Each row is one rating context at one known ability value. Use original facet labels.

Unknown levels and missing inputs are refused. Previously unobserved combinations of known levels are allowed and labelled in the returned `contexts` table.

`type` `"probability"` (one row per category) or `"information"` (one row per supplied context/ability). Information is per rating, not a sum over all observed exposures as in `compute_information()`.

`method`, `clusters`, `adjust`, `level`, `simultaneous`
As in `confint.mfrm_fit()`.

`object`, `x` A saved `mfrm_curve_intervals` result.

`...` Unused.

`title`, `subtitle` Optional returned text; NULL removes it. Figure text is not drawn. Use `plot_data(p, "title")` and `plot_data(p, "notes")`.

`palette` Optional vector of colors, one per category/series. Line types distinguish series in addition to color. The default is colorblind friendly.

`draw` Draw the ggplot immediately? Default TRUE; FALSE returns it only.

`caption` Optional returned caption. If omitted, unavailable intervals are counted and explained, along with unobserved combinations of fitted levels. Two-family point-only curves instead state that calibration intervals are unavailable for the entire plot, without marking every curve point. NULL removes the caption without removing markers or the reasons in the saved table.

Details

The complete GPCM predictor, including fitted facet interactions, is evaluated from the free parameter vector. A central-difference Jacobian propagates the full joint covariance. Probability limits use a logit delta approximation; information limits use a log delta approximation for `slope^2 * Var(category | Theta, context)`. Bounds respect their support. Rounded probabilities zero/one and nonpositive information keep the point but have unavailable intervals. The supplied Theta values are fixed on the fitted native scale; their estimation uncertainty is not included. Estimated population parameters enter the joint covariance as nuisance parameters, not as a request to transform or integrate the Theta grid.

Bonferroni applies to the finite collection of all output rows in this call, not to the continuous curve between them. Ribbons connect grid-point intervals for display. Sandwich interpretation and independence assumptions are those of `confint.mfrm_fit()`. These are asymptotic approximations; numerical agreement does not establish sampling coverage. A reanalysis of saved correctly specified GPCM fits found substantial undercoverage for probability intervals in small incomplete designs, including Bonferroni-adjusted finite-grid families. Adjustment cannot repair inaccurate marginal approximations. Refitting every dataset in the affected unequal-rater-slope condition with the updated optimizer left its interval results unchanged. See the GPCM scope vignette for the evaluated grid, denominators and source limitations; an available interval is not a finite-sample coverage certification. Printing and the returned plot description identify the approximation. Custom plot text may omit that description; retain the method and its limitations in the figure legend or accompanying report. For one-family curves, crosses mark retained estimates whose intervals are unavailable. Two-family curves have no calibration intervals; the returned subtitle and caption state this without covering curves in crosses. A context with only one supplied ability value retains colored points. Panel labels put each facet on its own line. With many rating contexts, supply the comparisons of

interest in newdata or enlarge the exported figure; plotting does not select or discard contexts automatically. Ribbons stop at unavailable grid points; a missing ribbon does not mean zero uncertainty. Consult the table's InferenceReview for each reason.

A numerically verified but ill-conditioned information inverse can supply intervals with a warning, as described in confint.mfrm_fit(). The warning is saved in cautions and the table's InferenceReview, and appears in printing and the returned plot description. A custom subtitle, including NULL, changes the display without removing saved diagnostics. In a one-family fit with fixed facet anchors or group means, model and sandwich intervals condition on those supplied values. Their source calibration uncertainty is not propagated. Fixing a facet location does not fix its slope, thresholds or whole response curve. Availability still depends on the joint-information and numerical checks described above.

Value

A mfrm_curve_intervals list with table, settings, and the exact newdata, plus contexts identifying each InputRow as an observed or unobserved combination of fitted facet levels (ObservedContext). This records the retained rating design, not statistical identification or interval reliability. Its plot method returns a ggplot with the plotted data available in plot$data; descriptions and cautions are available through plot_data(p, "notes"), separately from the figure. summary() returns settings, the complete interval table, contexts, numerical cautions and the anchor note without recalculating information.

Two slope families

For a two-family fit, the two component slopes are multiplied in each specified context. The shared evaluator returns provisional fitted values, including after numerical nonconvergence; it does not certify their reliability. SE, Lower and Upper remain missing and CIEligible is false. Printing and default plot text identify this scope without claiming nominal intervals. Sandwich and simultaneous adjustments are not available. The existing plot, saved-data, report and export routes preserve these values and missing intervals. A slope value of one is not a rater-quality threshold.

See Also

confint.mfrm_fit(), category_curves_report(), compute_information()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_curve_intervals", run.dontrun = TRUE)
## Not run:
# How do category probabilities vary with ability in one rating context?
ratings <- load_mfrmr_data("example_core")
# Reuse the synthetic GPCM fit; estimation uses 61 quadrature points.
# Full regeneration recipe: examples/measurement-fits.R in the package.
fit <- readRDS(system.file("examples", "measurement-fits.rds",
  package = "mfrmr", mustWork = TRUE))$gpcm
# Select actual labels from the example; replace them with your context.
grid <- expand.grid(Theta = seq(-2, 2, length.out = 21),
  Rater = unique(ratings$Rater)[1],
  Criterion = unique(ratings$Criterion)[1])
```

```
curves <- mfrm_curve_intervals(fit, grid)
# At ability zero, read the probabilities of the original score categories.
subset(curves$table, Theta == 0,
  select = c(Category, Estimate, Lower, Upper, CIEligible, InferenceReview))
plot(curves)
# Ribbons describe calibration uncertainty at fixed ability, not uncertainty
# about an individual's ability. They are pointwise, not simultaneous bands.
file <- tempfile(fileext = ".rds")
saveRDS(curves, file) # Keep the complete result, including context and notes
saved <- readRDS(file)
saved$settings

## End(Not run)
```

`mfrm_dff_methods` *Review differential-functioning results*

Description

`summary()` returns the contrast, cell, summary, and configuration tables without refitting. A summary from `analyze_dff()` keeps its DFF label when printed; a summary from `analyze_dif()` keeps its DIF label.

Usage

```
## S3 method for class 'mfrm_dif'
summary(object, ...)

## S3 method for class 'mfrm_dff'
summary(object, ...)

## S3 method for class 'summary.mfrm_dif'
print(x, ...)

## S3 method for class 'summary.mfrm_dff'
print(x, ...)

## S3 method for class 'mfrm_dif'
print(x, ...)

## S3 method for class 'mfrm_dff'
print(x, ...)
```

Arguments

`object`, `x`	An `mfrm_dff` or `mfrm_dif` result, or its summary.
`...`	Reserved for generic compatibility.

Value

summary() returns a structured differential-functioning summary. print() returns its input invisibly.

See Also

analyze_dff(), analyze_dif()

---

mfrm_d_study *Plan measurement conditions with a D-study*

Description

Would adding tasks or raters make assessment scores more dependable? Use an estimated G-study to compare specified future designs for individual score components and optional weighted composites. Each scenario reports dependability and measurement error for mean scores over common random measurement conditions, such as tasks, raters, or occasions. Projection from an estimated G-study does not refit. Supplying an MFRM first estimates the separate main-effects G-study described below.

Usage

```
mfrm_d_study(x, design_grid = NULL, ...)

## Default S3 method:
mfrm_d_study(x, design_grid = NULL, ...)

## S3 method for class 'mfrm_multivariate_gstudy'
mfrm_d_study(
  x,
  design_grid = NULL,
  weights = NULL,
  negative_components = c("raw", "genova_algorithm", "genova_ems"),
  ...
)

## S3 method for class 'mfrm_multivariate_d_study'
print(x, digits = NULL, row.names = FALSE, ...)
```

Arguments

x	A result from mfrm_g_study(). A main-effects result from mfrm_generalizability() or a fitted MFRM uses the separate main-effects method, with its collapsed-residual assumptions. Dispatch preserves the source model; it does not select a new estimator.

`design_grid` A nonempty data frame or named list with one positive integer count column per G-study facet. Both G-study routes accept the exact facet label or `n_<facet>`, such as `n_Rater` and `n_Task`. Existing saved count-column names are also accepted: with the `rater/task` interface, use `Raters` and/or `Tasks` for the included facets. With `facets`, use its exact labels, for example `Rater` and `Occasion` for `facets = c(Rater = "Assessor", Occasion = "Session")`. The mapping is available in `x$design$count_columns`. Supply exactly one column per facet; extra columns, duplicate aliases and ambiguous matches are refused. Output keeps the saved count-column names, so existing plot and interval selections retain their meaning. Each row is one future complete balanced design preserving the G-study's crossed or nested structure. For `nesting = c(Rater = "Task")`, `Raters` means raters per task, not the total rater pool. `NULL` uses the G-study counts (children per parent for a nested facet) only when its retained data are complete and balanced; incomplete or unequal designs require an explicit grid. Counts need not match or exceed the G-study counts. Conditions are shared across persons and scores in every scenario.

`...` Arguments for the selected method. The fitted-MFRM method accepts arguments to `mfrm_generalizability()`; already estimated G-studies reject unused arguments. Printing accepts arguments for its coefficient table.

`weights` Optional named numeric vector for one composite, or a numeric matrix for several named composites. A vector must name every original score exactly once; matrix rows must name every score exactly once, and columns must have unique, nonempty composite names. Rows are matched by name, regardless of order. Every weight must be finite and each composite must have at least one nonzero weight. Signed weights allow difference scores, for example `c(Content = 1, Organization = -1)`. Weights are used as supplied, without normalization. `NULL` reports original scores only. Otherwise, each scenario reports the original scores once, followed by each composite. A vector's composite is named `"Composite"`; a matrix preserves its column names and order.

`negative_components` Component treatment for D-study point projections. `"raw"` (default) retains the original estimates. `"genova_algorithm"` replaces negative variance components with zero; `"genova_ems"` first recalculates components by source-design expected-mean-square equations, substituting zero as each negative component is encountered. The latter choices require a complete balanced univariate ANOVA G-study with one or two supported crossed or nested random facets. Original G-study estimates remain unchanged. These options do not supply adjusted-estimator intervals.

`digits` Display precision passed to the printed tables; `NULL` uses R's current setting. Stored estimates are unchanged.

`row.names` Logical; print table row names? Defaults to `FALSE`.

Details

Start with a concrete comparison, such as six versus twelve tasks for every examinee. Each row of `design_grid` is one scenario. With both facets, `data.frame(Raters = c(2, 4), Tasks = c(3, 3))` compares two versus four raters at three tasks; `expand.grid(Raters = c(2, 4), Tasks = c(3,`

6)) requests all four combinations. These counts apply to every person and score, not to the total number of observed ratings. For facets = c(Rater = "Assessor", Occasion = "Session"), use expand.grid(Rater = c(1, 2), Occasion = c(2, 4)). The same workflow applies to any supported one- or two-facet model; extra columns cannot introduce conditions absent from the G-study.

This is the primary D-study entry for one or several scores. It replaces the public mfrm_multivariate_d_study() name without changing its result class, component estimates, projections or uncertainty methods. For either source model, summary(d) returns its full-precision coefficient table. Optional interval summaries remain specific to the interaction-component route. Main-effects projections do not acquire those interval methods.

Read summary(d) together with plot(d):

- G concerns consistency of relative ordering, such as ranking examinees. It is not the probability of identifying the best person or the winner of a competition. Close leading scores can remain hard to distinguish even when G is high; this function provides no Person-rank intervals.
- Phi concerns absolute score levels and also counts shifts from easier tasks or more lenient raters as error. It is not pass/fail accuracy.
- RelativeSEM and AbsoluteSEM express these errors in the units of the mean score or composite. Smaller SEMs mean less error. They are not confidence intervals for G or Phi.

Larger G/Phi and smaller SEM indicate greater dependability under the model. There is no universally acceptable coefficient: consider the use of scores, consequences of error, and workload before choosing a design. Compare a chosen score/composite and metric across feasible plans. For example, data.frame(Raters = c(2, 3, 4), Tasks = c(6, 4, 3)) holds the number of ratings per person at twelve. It does not hold examinee task burden or total cost constant. The largest projected coefficient is a point-estimate ranking, not evidence that one plan is reliably better. Consider the size of the projected difference and which differences would matter for the assessment. Small rank changes need not imply large practical losses. If a metric is unavailable for any candidate, comparing only the remaining values does not resolve the full planning question. plot(d, type = "sem") shows the SEMs. Plots select a sole composite by default, or the first score when there are no composites. With several composites, specify one with plot(d, composite = "Equal"); use plot(d, score = "V") to inspect an original score named V. The title metadata identifies the plotted score and any composite weights; it is returned with the plot, with titles and notes omitted from the figure by default. Check GStatus, PhiStatus, RelativeSEMStatus, and AbsoluteSEMStatus: each explains the corresponding metric's availability. NA means unavailable, not zero dependability. ComponentPSD separately flags covariance components needing review. See plot.mfrm_multivariate_d_study() for interpreting curves and unavailable estimates.

For a balanced one-facet ANOVA source, confint.mfrm_multivariate_d_study() returns exact F intervals for G and generalized confidence intervals for Phi (parm = "Phi") under independent normal random effects, using the source study's degrees of freedom at each requested future count. Phi includes Monte Carlo endpoint diagnostics and is not an exact-coverage method. Add reference = 1 for intervals on changes from the first plan: G maps the F set; Phi maps the generalized set and retains approximate coverage. Both include interior extrema; Phi differences retain joint numerical endpoint bands. These methods also apply to balanced nested plans when future child counts per parent equal the source count, using the parent count in the one-facet reduction. Complete balanced crossed two-facet sources default to approximate MLS null inversion for individual G/Phi intervals. Undefined MLS widths are conservatively retained and flagged by FormulaComplete; original estimates and numerical failures remain explicit. Small-source intervals may span almost all of [0,1]. These sources and general balanced nested plans also have

explicitly selected experimental method = "approximate_f" and "mean_square_region" options for G/Phi; a vector requests both and plot() compares them. The confidence-interval help explains their assumptions, limitations and unavailable results. Use mfrm_multivariate_d_compare() separately for its supported paired-plan differences. For balanced or identifiable incomplete sources, explicit method = "reml_profile" fits approximate G/Phi or SEM intervals from the used ratings under normal effects. It retains original ANOVA/MINQUE estimates and reports the separate REML estimates as IntervalCenter. Numerical diagnostics and experimental finite-sample/boundary qualifications remain explicit; it is not a default. With reference, it supplies shared-profile interval images for one-facet or common-child nested plans, and direct difference profiles for other crossed/nested comparisons. The latter need optional nloptr and retain experimental calibration, especially at zero. SEM uses a direct error-variance profile followed by square roots; the numerical interval surrounds the REML center without enumerating remote disconnected accepted regions. Checked boundary populations showed below-nominal inclusion despite high numerical availability. Select parm = "RelativeSEM" or "AbsoluteSEM" for intervals on SEM itself from supported balanced crossed/nested sources. A single contributing mean square uses chi-square limits; otherwise MLS limits for the combined error variance are transformed by the square root. MLS coverage is approximate. One-facet SEM differences also accept reference = 1; nested MLS comparisons require a common future child count across plans. For one-facet RelativeSEM, explicit method = "residual_contrast" also accepts retained ANOVA/MINQUE ratings, including incomplete MINQUE sources. It transforms one residual chi-square interval and retains original points, a separate projected IntervalCenter, unavailable states and shared uncertainty across plans. Normal independent residuals and noninformative omissions are required; this does not supply AbsoluteSEM, G/Phi or multifacet intervals. Explicit method = "mean_square_region" supplies conservative individual SEM intervals and G/Phi/SEM difference enclosures for general balanced crossed/nested two-facet plans. The enclosure retains common source uncertainty but can be much wider than the tight difference image. plot(d, type = "sem", metric = "AbsoluteSEM", intervals = TRUE, assumption = "normal") adds the corresponding ribbon without refitting.

Value

An mfrm_multivariate_d_study list containing coefficients (scenario, counts of included facets, score/composite identity, universe/relative/absolute variances under the selected treatment, G/Phi, relative/absolute SEM, ComponentTreatment, and row and metric-specific status, and ComponentPSD), covariances (three matrices per scenario), design_grid, weights (the supplied vector or matrix, reordered to the G-study score order), component_diagnostics, negative_components, effective_components, effective_component_diagnostics, and gstudy (the unchanged source result). summary() returns coefficients. print() returns the same D-study invisibly without computing intervals. summary.mfrm_multivariate_d_study() with intervals = TRUE explicitly calculates and collects supported intervals, methods and source context. print() also shows source level/row counts, future-plan conventions and the supplied composite weights. Save the full result with saveRDS(d) and reopen with readRDS() to preserve that context. A CSV of summary(d) contains the coefficient table only; accompany it with the weights, source design and row usage, future design and score convention. Main-effects inputs instead retain their mfrm_d_study data-frame result; see mfrm_d_study_main_effects for its fields and limitations.

Calculation and interpretation limits

For crossed covariance component matrices P, R, T, PR, PT, RT, and E, universe-score covariance is P. Relative-error covariance is PR/n_r + PT/n_t + E/(n_r*n_t). Absolute-error covariance adds

R/n_r + T/n_t + RT/(n_r*n_t). The G-study number of persons does not divide individual-score universe variance. Holding a count constant does not turn its random facet into a fixed facet. Composite weights define linear score combinations. They do not implement score-dependent panel trimming or an entire competition scoring rule. For a Person-by-Task G-study, E combines Person-by-Task interaction and within-cell error: relative-error covariance is E/n_t, absolute-error covariance is (T + E)/n_t, and universe-score covariance remains P. The same formulas apply to a Person-by-Rater design or to facets selected by facets, with names replaced in the declared order. For a single facet F and its planned count n, relative error is E/n and absolute error is (F + E)/n. A D-study cannot introduce an absent facet. For fixed tasks represented by score columns in a Person-by-Rater model, vary only Raters. The task set remains fixed, and each planned rater scores all tasks. See "Fixed tasks as score components" in mfrm_g_study() for the target and allocation requirements.

For raters nested within tasks, use the five components P, T, R(T), PT, and E, where E includes the Person-by-Rater-within-Task interaction. With n_r raters per task and n_t tasks, relative error is PT/n_t + E/(n_r*n_t); absolute error adds T/n_t + R(T)/(n_r*n_t). Universe covariance remains P. Other nested facet labels follow the same rules. All persons receive the same tasks and their nested raters; two raters per task at six tasks means twelve distinct raters, not two raters each judging six tasks. Future unequal child counts, changes of nesting are not supported. Nested G/Phi defaults and their differences require unchanged source child counts. Nested MLS SEM differences need a common future child count across plans, which may differ from the source. General nested coefficient intervals require explicit experimental methods; general difference enclosures use method = "mean_square_region". Individual nested SEM intervals allow both counts to change. See confint() above for the derivations and limitations.

For incomplete source data, counts of distinct observed levels are not per-person replication counts. This function projects a future complete design; it does not estimate the reliability of the observed sparse roster, heterogeneous person-specific assignments, or averages of them.

Each score uses diagonal entries; a composite uses w' Sigma w for each of the three covariance matrices. Thus between-score covariance affects composite dependability. G = UniverseVariance / (UniverseVariance + RelativeErrorVariance) and Phi uses absolute error instead. SEMs are square roots of the corresponding error variances, in the units of the score or specified composite of mean scores. Multiplying all weights by a nonzero constant multiplies SEMs by its absolute value but leaves G/Phi unchanged. Weights express a substantive choice; this function neither chooses them nor identifies an optimal design. Interpret a difference only when subtraction is meaningful on the supplied score scales. Comparing several composites holds the estimated components and planned designs constant, but different weights may change the construct being assessed. A higher coefficient alone does not justify that change. Compare SEMs across composites only when their scales are comparable.

The same weights define the universe-score composite and its observed mean-score estimate (the equal WWTS/AWTS case in mGENOVA). Different target and estimation weights, profile reliability, and accuracy at a cut score are not provided. Unlike mGENOVA's default D-study procedure, negative variance estimates are not replaced with zero by default. mGENOVA replaces negative variance components with zero for the D-study unless its NEGATIVE option is selected (manual, pages 15 and 21). That rule concerns variances, not all negative covariances, and does not guarantee PSD component matrices. Do not transfer that option's meaning to GENOVA 3.1 or jGENOVA: their NEGATIVE option controls printing of negative estimates. Their D-studies zero negative variances with either ALGORITHM or its default EMS convention; EMS also recalculates other components using the zero replacements. Matching a raw G-study estimate therefore need not give

matching default D-studies. A matched negative-interaction example gives Phi = 0.73096 with the raw components retained here, 0.72727 in GENOVA 3.1 ALGORITHM, and 0.73532 with its default EMS. This is a component-treatment difference, not a coefficient-formula discrepancy. Select negative_components explicitly to reproduce either GENOVA D-study convention in supported univariate designs. EMS recalculation uses the source counts, before division by the future-plan counts. It is not REML or a positive-semidefinite matrix repair. Multivariate and MINQUE sources require "raw"; negative cross-score covariance is not a negative variance. GENOVA's NEGATIVE printing option is separate from this component-treatment choice.

gstudy retains raw estimates and diagnostics. effective_components contains the matrices actually used in projections; their diagnostics are in effective_component_diagnostics. ComponentTreatment identifies the choice in coefficient tables. ComponentPSD still describes the original G-study, so zero replacement does not erase evidence of problematic raw components. Save the complete result to preserve both sets. For GENOVA-adjusted results, confint() retains population-parameter intervals based on the original source study alongside adjusted points. IntervalEstimate records the original projection; bounds are not recentered on the adjusted Estimate. Any IntervalSE and other sampling diagnostics retain the original interval-method basis. Summary and plot methods accept intervals with the same recorded treatment. Supported population plan differences use confint(d, reference = ...); the separate delta-estimator comparison API still requires raw components. These intervals do not characterize the adjusted estimator's sampling law. See confint.mfrm_multivariate_d for assumptions and distinctions.

Original covariance matrices and the selected projected variances remain available. G requires nonnegative universe and relative-error variances and a positive sum; Phi uses absolute-error variance instead. Each SEM requires only its corresponding nonnegative error variance. A failed requirement makes only that metric NA. Zero universe variance yields a coefficient of zero when its error variance is positive; a zero total variance leaves the coefficient undefined. Under "raw", no estimates are clipped or replaced.

Status summarizes a row as "Available", "Partially available", or "Unavailable"; the four metric-specific status columns give the reasons. ComponentPSD is TRUE only when all G-study component matrices pass their numerical PSD check. A FALSE value flags components for review without suppressing otherwise calculable projections. Print and plot methods display a note for non-PSD components. Calculability does not validate the joint covariance model or establish precise estimation; inspect component_diagnostics before interpreting estimates. Raw estimates can yield Phi > G when projected absolute error is less than relative error. Under the stated model, the extra absolute-error contributions are nonnegative; such a reversal calls for review of the estimated components, not a conclusion that absolute decisions are more dependable. Passing the matrix checks does not establish model fit. Outputs are point projections conditional on the estimated components, without confidence intervals, missing-data correction, or MFRM latent inference.

References

Brennan, R. L. (2001). *Generalizability theory*. Springer. Chapters 9–11. doi:10.1007/97814757-34560.

Brennan, R. L. (2001). *Manual for mGENOVA, Version 2.1*. Iowa Testing Programs Occasional Papers, No. 50. Pages 16 and 20–22; Appendices E and F, pages 74–81. Crick, J. E., and Brennan, R. L. (2021). *Manual for jGENOVA, Version 1.0*. Pages 2-10–2-11, 3-5, and A-22–A-23. Both manuals are available from the author's GENOVA Suite.

See Also

plot.mfrm_multivariate_d_study(), mfrm_g_study(), mfrm_multivariate_d_compare(), mfrm_d_study()

Examples

```
# Published synthetic two-score data: the same six tasks for ten persons.
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)

# Question: would doubling tasks improve the dependability of W minus V?
d <- mfrm_d_study(g, data.frame(Tasks = c(6, 12)),
  weights = c(V = -1, W = 1))
summary(d)
plot(d)
# G rises from 0.300 to 0.462, conditional on these estimated components.
# This is a planning projection, not evidence from twelve observed tasks.

# Optional alternative views, GENOVA conventions and composite weights.
# example("mfrm_d_study", run.dontrun = TRUE)
## Not run:
plot(d, type = "sem")
# Without weights, report the original scores without creating a composite.
individual <- mfrm_d_study(g, data.frame(Tasks = c(6, 12)))
plot(individual, score = "V")

# Explicit GENOVA convention for a single observed score.
g_one <- mfrm_g_study(tasks, "V", rater = NULL)
d_ems <- mfrm_d_study(g_one, data.frame(Tasks = c(6, 12)),
  negative_components = "genova_ems")
summary(d_ems)
ci_ems <- confint(d_ems, assumption = "normal")
# Adjusted point and original-source population interval have separate bases.
ci_ems[, c("Estimate", "IntervalEstimate", "Lower", "Upper", "IntervalBasis")]
d_ems$effective_components # Compare with d_ems$gstudy$components.
plot(d_ems, metric = "G", intervals = ci_ems) # Original-source population bounds.

# Compare weight choices on the same task-count scenarios.
weights <- cbind(Equal = c(V = 0.5, W = 0.5),
  W_focus = c(V = 0.2, W = 0.8), Difference = c(V = -1, W = 1))
alternatives <- mfrm_d_study(g,
  data.frame(Tasks = c(6, 12)), weights = weights)
summary(alternatives) # Original scores plus all three named composites.
plot(alternatives, composite = "Equal")
plot(alternatives, composite = "Difference", type = "sem")
# Choose weights for the intended interpretation, not solely for higher G.

## End(Not run)
```

mfrm_d_study_main_effects *Main-effects D-study approximation*

Description

mfrm_d_study() applies a practical D-study projection to the variance components from mfrm_generalizability(). It answers questions such as "what happens to G and Phi if we use 2, 3, or 4 raters?" without re-fitting the Rasch/MFRM model.

Usage

```
## S3 method for class 'mfrm_generalizability'
mfrm_d_study(
  x,
  design_grid = NULL,
 residual_scaling = c("highest_order", "single_condition", "none", "sensitivity"),
  ...
)

## S3 method for class 'mfrm_fit'
mfrm_d_study(
  x,
  design_grid = NULL,
  object_facet = "Person",
  random_facets = NULL,
 residual_scaling = c("highest_order", "single_condition", "none", "sensitivity"),
  ...
)

## S3 method for class 'mfrm_d_study'
summary(object, ...)
```

Arguments

x	Output from mfrm_generalizability() or an mfrm_fit. If an mfrm_fit is supplied, mfrm_generalizability() is called first.
design_grid	Data frame or named list giving planned counts for each random measurement facet. Column names may be the facet names themselves (for example Rater) or n_ plus the facet name (for example n_Rater). Supply exactly one count column per facet, with unique names and an unambiguous one-to-one correspondence. Extra columns, duplicate aliases for a facet, or a column serving two facets are refused. Counts must be positive integers. When NULL, one row using the observed number of levels is returned.

residual_scaling
: How the collapsed residual variance should be scaled when planned facet counts increase. "highest_order" treats the residual as highest-order person-by-all-conditions/error variance and divides by the product of planned counts. "single_condition" divides by the smallest planned facet count, a sensitivity assumption for variation associated with the least-replicated condition. It is not a confidence bound. "none" leaves the residual unscaled. "sensitivity" returns all three assumptions for each design row.

...
: Additional arguments passed to mfrm_generalizability() when x is an mfrm_fit. With an already estimated main-effects G-study or its D-study summary, unused arguments are rejected. In particular, composite weights are unsupported here; they require the score-component route from mfrm_g_study().

object_facet, random_facets
: Passed to mfrm_generalizability() when x is an mfrm_fit.

object
: A main-effects D-study result.

Details

The projection uses the variance decomposition already estimated by mfrm_generalizability(). For a random measurement facet j, main-effect variance contributes sigma2_j / n_j to the absolute-error denominator. Under the fitted additive model, residual errors are independent across observations. For a complete future crossing with common facet levels for all persons, their mean has variance sigma2_Residual / prod(n_j). Facet-wide shifts cancel from relative comparisons but enter absolute error as sum(sigma2_j / n_j). This justifies "highest_order" within that model. If omitted interactions actually exist, the fitted residual does not recover their separate variances or correlations; dividing it by the product is then an assumption. "single_condition" and "none" are alternative planning assumptions, not estimates of these correlations, confidence bounds, or a guarantee that the true projection lies between them. The relative-decision denominator uses only this scaled residual term. Missing or invalid required components leave the affected coefficient unavailable; they are not zero variance. These are point projections conditional on estimated components, not uncertainty bounds or automatic recommendations for sample size. Returned G/Phi retain full precision, as do the values used for their reference-status labels. Round only for presentation, after any comparisons.

Error variances are evaluated independently of object variance: they remain available when object variance is zero or missing. Zero object variance with positive error gives a coefficient of zero; zero total variance is undefined. For checked sources, only identified variance sums enter the output. For example, two confounded facet main effects can still contribute their identified sum when their planned counts are equal; unequal counts require the separate, unidentified components. Coefficients use scaled sums to avoid overflow. A coefficient can therefore remain calculable even when its error variance exceeds floating-point range. Such unrepresentable quantities are returned as NA with reasons.

This is a pragmatic D-study planning layer, not a full p x r x i ANOVA decomposition. If person-by-rater or person-by-item interactions are a primary estimand, consider mfrm_g_study() and mfrm_d_study() for one or two common random facets, including Person-by-Task, Person-by-Rater, and Person-by-Rater-by-Task. Those functions also accept a single score and estimate the corresponding interaction components explicitly. Changing residual_scaling here only explores assumptions; it does not estimate the omitted interactions.

Pass the result directly to mfrm_report() for full projection tables and their assumptions. Newly created results retain source variance components, estimation settings and observed facet counts. Reports of older compatible results mark missing source details not_recorded; report creation never reconstructs components or computes confidence intervals.

The G and Phi values returned here belong to the generalizability-theory metric family. They should not be interpreted as coefficient alpha, omega, KR-20, or IRT marginal/separation reliability, even though all of those summaries may be displayed on a 0–1 scale in broader reporting dashboards. They remain on the observed numeric score scale inherited from mfrm_generalizability(), not the fitted MFRM latent scale.

Value

An object of class mfrm_d_study, a data.frame with one row per design scenario and columns for planned facet counts, variance terms, projected G, projected Phi, interpretation bands, and identification status inherited from mfrm_generalizability(). InputRows, UsedRows, and ExcludedRows describe the source G-study, not the planned D-study sample; GStudyDataSource identifies their scope. The data_usage attribute retains its row accounting, including after subsetting. Older source results without accounting have NA counts; they are not assumed to have used all rows. summary(d) returns the coefficient data frame with these saved attributes, rather than generic per-column descriptive statistics. No intervals or new variance-component estimates are computed. GReason, PhiReason, ObjectVarianceReason, RelativeErrorReason, AbsoluteErrorReason and ResidualDivisorReason explain unavailable quantities; empty strings mean no calculation failure. Reasons distinguish unidentified targets, missing/invalid components, zero total variance and numerical range. Fit cautions remain separate in IdentificationStatus and IdentificationNote. ComponentIdentificationChecked and the component_identification attribute retain the source design check.

References

Cronbach, L. J., Gleser, G. C., Nanda, H., & Rajaratnam, N. (1972). *The dependability of behavioral measurements: Theory of generalizability for scores and profiles*. Wiley.

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560. This framework does not recover the interaction components omitted by the source main-effects model.

See Also

plot.mfrm_d_study(), mfrm_generalizability(), mfrm_d_study(), evaluate_mfrm_design(), recommend_mfrm_design(), plot_data()

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
if (requireNamespace("lme4", quietly = TRUE)) {
  gt <- mfrm_generalizability(fit)
  ds <- mfrm_d_study(gt, data.frame(Rater = c(2, 3, 4), Criterion = 4))
  ds[, c("n_Rater", "n_Criterion", "G", "Phi",
```

```
        "GStatus", "PhiStatus", "IdentificationStatus")]
  # If IdentificationStatus is not "identified", even large G/Phi
  # values remain identification warnings, not decision-ready evidence.
}
```

`mfrm_facet_intervals` *Pointwise intervals for fixed facet estimates and contrasts*

Description

Describe uncertainty about a fixed facet estimate, such as rater severity, or a specified difference between two raters. Start from a supported fitted model; the point estimates stay the same when you change the interval method.

Usage

```
mfrm_facet_intervals(
  fit,
  facet = NULL,
  contrasts = NULL,
  method = c("model", "sandwich"),
  clusters = NULL,
  adjust = FALSE,
  level = 0.95,
  scale = c("native", "standardized"),
  comparison_review = NULL,
  anchor_covariance = NULL,
  independent_anchors = FALSE,
  interaction = NULL
)

## S3 method for class 'mfrm_facet_intervals'
print(x, ...)

## S3 method for class 'mfrm_facet_intervals'
summary(object, ...)

## S3 method for class 'mfrm_facet_intervals'
confint(object, parm = NULL, level = object$settings$level, ...)

## S3 method for class 'mfrm_facet_intervals'
vcov(object, ...)

## S3 method for class 'mfrm_facet_intervals'
plot(
  x,
  comparison = TRUE,
```

```
  draw = TRUE,
  preset = "standard",
  title = paste(x$settings$facet, "estimates and contrasts"),
  subtitle = paste0(format(100 * x$settings$level, trim = TRUE),
    "% pointwise normal intervals | ", if (x$settings$method == "sandwich")
   paste0("Sandwich | ", x$settings$clusters, " independent clusters assumed") else if
    (identical(x$settings$procedure, "cml_conditional_information"))
    "Conditional information" else if (identical(x$settings$uncertainty,
    "independent_anchor_delta")) "Information + anchor source" else
    "Observed information"),
  caption = if (any(!is.finite(x$table$Estimate)))
    paste0("Not estimated: no point or interval | ",
   "Crosses: estimated points with unavailable intervals") else if (any(x$table$Status
    != "available"))
   "Open diamonds: fixed targets | Crosses: unavailable selected intervals" else NULL,
  reference = 0,
  show_legend = TRUE,
  ...
)
```

Arguments

`fit`	An inference-ready RSM/PCM MML fit from `fit_mfrm()`, using a fixed standard-normal person distribution, fixed quadrature and unit observation weights. A fit whose category review concerns singleton counts can additionally use `method = "model"` after the separate checks described under "Small samples and singleton categories". Experimental two-family GPCM MML also supports `method = "model"` with fixed-grid EM or direct adaptive integration; see "Two-family location targets" below. Direct MML native locations additionally support estimated normal RSM/PCM populations, including conditional-normal latent regression, and one-family GPCM with fixed N(0,1) or an estimated intercept-only normal population, using fixed or adaptive integration; see "Native locations" below for their scope and separate numerical checks. Native RSM/PCM CML results additionally support `method = "model", scale = "native"`, including the saved targets from `fit_scope = "comparisons"`. Matching full RSM/PCM MML interaction fits use the `comparison_review` extension below; their explicit model and source checks remain separate.
`facet`	A non-person facet, for example `"Rater"`. May be omitted when `comparison_review` supplies a facet-by-origin interaction target, or for a CML comparison-only fit, which already retains its targets.
`contrasts`	Optional numeric matrix with distinct target row names and columns named by every level of `facet`. Columns are aligned by name. A row with coefficients `c(1, -1, 0)` estimates the first level minus the second. By default, each level is reported.
`method`	`"model"` (default) uses ordinary observed information; `"sandwich"` uses independent-cluster marginal-likelihood scores. The native-location and experimental two-family extensions support only `"model"`; sandwich support remains limited to the ordinary fixed-population RSM/PCM route.

clusters — For sandwich inference, an optional data frame with exactly one row per fitted person and complete Person and Cluster identifiers. Without it, persons are the independent units. A larger cluster could be a school or clinic if different schools or clinics are independent. A person cannot be split across clusters. Supply original person labels, even when the input person column has another name.

adjust — Logical; with method = "sandwich", optionally multiply the covariance by G/(G-1), where G is the cluster count. Default FALSE. This scaling does not guarantee small-sample coverage.

level — Pointwise confidence level, strictly between zero and one.

scale — "native" (default) retains the fitted ability units. "standardized" divides an explicit within-facet zero-sum contrast, or the interaction contrasts declared by comparison_review, by the population SD, using full joint delta-method uncertainty. Requires method = "model" and a fixed N(0,1) or estimated intercept-only normal population; individual locations and latent regression are not supported.

comparison_review — Optional describe_mfrm_data() review or its $comparison_review for a matching full native RSM/PCM MML interaction fit. Uses its declared facet-by-origin contrasts instead of facet locations; omit contrasts and optionally facet. Requires method = "model". See "MML interaction contrasts" for scope and assumptions. Omit this argument for a CML comparison-only fit; its saved review is reused.

anchor_covariance — Optional full sampling covariance of the supplied anchor estimates, including their correlations, in native ability units squared. Row and column names must match every directly anchored level on the one noncentered facet. Requires the fixed-grid direct RSM/PCM anchor-origin route with an intercept-only population and fixed SD. NULL retains the existing conditional-on-anchors calculation.

independent_anchors — Logical, default FALSE. Set TRUE with anchor_covariance only when source calibration and receiving responses are independent. Shared respondents or dependent clusters require cross-sample covariance and are unsupported. This is an explicit assumption, not a check inferred from different Person identifiers.

interaction — Optional exact name of one fitted two-facet interaction, for example "Rater:Criterion". Omit facet and comparison_review. This selects jointly fitted cell effects rather than fixed-baseline bias screens. Requires method = "model", scale = "native", full fixed-grid RSM/PCM MML (direct, EM or hybrid) with fixed N(0,1), one interaction, no anchors, groups, dummy facets or shrinkage. Every engine must pass the common numerical and full-joint information checks; EM stopping alone does not qualify intervals. See the joint-cell section below.

x — Output from mfrm_facet_intervals().

Also for print(): An object returned by mfrm_facet_intervals().

... — For print.mfrm_facet_intervals(): Unused by print and summary methods.

For plot(): Unused.

object	An object returned by mfrm_facet_intervals().
parm	For confint(), optional distinct target names from object$table$Target. Extracts the stored limits without recalculation; level must match the saved level. Use mfrm_facet_intervals() to calculate a different level. Unavailable limits remain NA; inspect summary(object) for their eligibility and reasons.
comparison	Logical. With a sandwich result, also show its ordinary model-based interval. With source-propagated anchors, show the separate conditional-on-anchors interval. Both use the same point estimate.
draw	Logical; FALSE returns exact plotted data without opening a device.
preset	A package plot preset, for example "standard" or "monochrome".
title, subtitle, caption	Optional returned text; NULL omits it. Defaults describe the facet, exact confidence level, method and fixed/unavailable symbols. This text is not drawn; use plot_data(p, "notes") and plot_data(p, "title") to retrieve it.
reference	Optional finite vertical reference; default zero. Use NULL to omit it. This is not a rater-quality threshold.
show_legend	Logical; show the method legend (default TRUE).

Details

Compare ordinary observed-information intervals with a sandwich covariance that treats a person's complete response vector, or an explicitly declared larger cluster, as the independent sampling unit. Estimates are not refitted.

For cluster score s_g and observed negative-log-likelihood Hessian H, the unadjusted sandwich is solve(H) %*% sum_g(s_g %*% t(s_g)) %*% solve(H). Scores aggregate all responses of a person before forming the outer products; grouping individual rating rows would be a different and incorrect calculation for this marginal likelihood. The full covariance is transformed through the fitted constraints and requested contrasts. Both methods use normal critical values. No automatic method selection, multiplicity adjustment or significance flag is supplied.

The sandwich requires many independent sampling units and appropriate regularity. It permits dependence within the declared unit but does not establish independence between units. A small number of units can give poor intervals even with nonsingular covariance. If their scores do not span the free-parameter space, sandwich intervals are unavailable; point estimates and the reason remain. For ordinary fixed-population RSM/PCM, an ineligible source fit still causes an error. If the source is eligible but joint information is unavailable, nonpositive or regularized, finite points remain with Status = "information_unavailable", CIEligible = FALSE, missing model and sandwich uncertainty, and the explanation in InferenceReview. This includes information-workspace limits. A sandwich also needs the inverse information; switching methods cannot repair its absence. Failed information can limit confidence in point accuracy along weak directions. No inverse is regularized for this output. Invalid arguments and cluster mappings remain errors. Targets fixed by constraints have no inferential interval. Known anchors exclude anchor uncertainty. An ill-conditioned but numerically verified unregularized inverse can be used with a warning. Successful inversion does not establish reliable normal intervals. Review interval width, boundary proximity and quadrature sensitivity; changing to sandwich covariance does not remove this concern.

plot() method: Plot fixed-facet interval methods

For `plot()`: Fixed targets use open diamonds without inferential intervals. A conditionally fixed anchor can have a source-propagated interval; the comparison displays its conditional fixed value separately. Method labels distinguish source propagation from conditional observed information. Unavailable intervals in either displayed method use crosses; an ordinary comparison interval does not substitute for a missing sandwich interval. The zero line is a reference, not a threshold of practical importance or a rater-quality rule. No fit or covariance is recalculated. Target order is preserved. Weak-information cautions remain in the returned data and appear in the default subtitle. Custom or omitted subtitles change display only. For experimental two-family GPCM locations, the axis uses fixed-N(0,1) ability units and the default subtitle states the location reference and unqualified coverage, with a deficient Person-score rank caution when applicable. These are not uniform expected-rating differences. Standardized contrast axes use population SD units; returned notes identify whether SD uncertainty is estimated or the known SD is one. Reviewed interaction contrasts retain group direction, panel weights and scale in their source result; returned labels distinguish them from facet locations. Native CML intervals are labeled conditional information. Their returned notes describe asymptotic validity without implying weak information or an estimated ability population SD. When a comparison has no estimate, its row is labeled "not estimated" and has no point or interval. An empty display range is only an axis placeholder; it does not assign a value to unavailable targets. Joint interaction-cell results retain cell identities, counts and sparse flags in `plot_data(p, "interaction_cells")`. Their axis uses additive predictor shifts; row labels show the two level names in fitted facet order. Units follow the fixed N(0,1) ability scale. Positive effects raise expected coded scores relative to the jointly fitted main effects. These pointwise intervals are not fixed-baseline bias screens or fairness labels.

Value

An `mfrm_facet_intervals` object with `table`, selected target `covariance`, `model_covariance`, free-parameter `parameter_covariance`, `model_parameter_covariance`, `person_scores`, `cluster_scores`, the `clusters` mapping, exact `contrasts`, `settings`, and source `fit`. The model-based SE and interval remain alongside the selected method. If ordinary MML joint information is unavailable, free-parameter covariances are `NULL` and target covariances contain `NA` (zero for constraint-fixed rows/columns); `unavailable_reason` and `checks` explain the failure. Sandwich scores and their rank are then uncomputed, not zero. Score rows are derivatives of a person's marginal log likelihood, not observed category scores or ability estimates. Scores are absent for `method = "model"`. When weak information passes numerical refinement, `cautions` and `information_review` retain the warning and checks. `InferenceCaution` also accompanies the interval table when applicable. Each row retains `Status`, `CIEligible` and a plain-language `InferenceReview`. `CIEligible` concerns the selected interval, not the validity of the point estimate or a guarantee of coverage. `fixed` means the target is fixed by constraints, with no inferential interval. `insufficient_cluster_rank` withholds sandwich bounds because independent cluster scores do not span all free parameters. The point and separate `ModelSE`/`ModelLower`/`ModelUpper` are retained; those model bounds are not a replacement robust interval. Review the independent sampling units and model dimension; do not split dependent clusters to obtain an interval. `nonpositive_variance` or `nonfinite_bounds` denotes an unusable target-specific calculation. Missing bounds are not zero uncertainty. Two-family results additionally retain `checks`, `numerical_checks`, `score_rank`, source identity, scale references and interval refusal reasons. Their table includes `CIEligible`; failed numerical checks leave bounds and SEs missing while preserving the fitted point estimates. Native-location results retain checks, numerical comparisons, full and target covariance, target Jacobian and source identity, without an empirical Person-score rank requirement. Standardized results also retain `NativeEstimate`, the population SD, whether it was estimated, and the full transformation Jacobian. `CIEligible` de-

scribes numerical availability; CoverageQualified = FALSE does not claim finite-sample coverage for the transformed interval.

For plot(): Invisibly, an mfrm_plot_data object retaining the table, contrast coefficients, settings and labels. Use plot_data() for custom graphics; as_ggplot() returns a customizable ggplot with the same saved values.

Jointly fitted interaction cells and contrasts

Fit a prespecified two-facet interaction with fit_mfrm(facet_interactions = "Rater:Criterion"), then use mfrm_facet_intervals(fit, interaction = "Rater:Criterion"). The default returns all fitted cells, including the dependent last row and column. Cell labels, counts, sparse flags and identification constraints are retained in interaction_cells. This bounded route uses a fixed standard-normal ability population; latent regression, estimated population variance, adaptive integration, JML, CML, GPCM and multiple fitted interaction terms are outside this extension.

An interaction effect is an additive predictor shift after jointly fitting the main effects. Positive values increase expected coded ratings; they do not automatically imply favorable performance or unfairness. Effects have zero row sums and zero column sums across the declared levels, without weighting those constraints by cell counts. Their values depend on that definition and the included main effects. This differs from estimate_bias(), which holds the fitted calibration and person measures fixed while estimating an additional residual shift. Neither result proves substantive rater bias.

Write free interaction coordinates as gamma and their expansion to all cells as G gamma. Cell covariance is G V_gamma G', where V_gamma is the interaction block of the inverse full marginal information, including main effects and thresholds. It is not the inverse of only the interaction information block. Dependent cells consequently have correlated estimates and a singular joint cell covariance; this alone does not invalidate individual pointwise intervals. For a prespecified cell contrast C, the covariance is C G V_gamma G' C'. Supply every cell Target as a named column of contrasts, including zeros. A contrast fixed at zero by the row/column constraints has Status = "fixed" and no inferential interval; it is not evidence for a null interaction.

Source consistency, identification/boundary records, full information, stationarity and finer-quadrature checks must pass. Unavailable intervals retain finite point estimates and reasons. Sparse-cell flags are descriptive, not a sample-size guarantee. Full-design identification and numerical success do not establish nominal coverage. These are pointwise Wald intervals, not simultaneous bounds or multiplicity-adjusted cell decisions. Saved tables, plots, apa_table() and mfrm_results() retain the same values and cautions.

A bundled-data example uses the existing estimation and interval APIs:

```
ratings <- load_mfrmr_data("example_core")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  model = "PCM", step_facet = "Criterion", noncenter_facet = "Criterion",
  facet_interactions = "Rater:Criterion", quad_points = 61)
cells <- mfrm_facet_intervals(fit, interaction = "Rater:Criterion")
summary(cells)[, c("Target", "Estimate", "SE", "Lower", "Upper", "Status")]
plot(cells)
# Compare the first two displayed cells; choose the scientific contrast first.
C <- matrix(0, 1, nrow(cells$table),
  dimnames = list("First minus second", cells$table$Target))
```

```
C[1, 1:2] <- c(1, -1)
difference <- mfrm_facet_intervals(fit, interaction = "Rater:Criterion",
  contrasts = C)
vcov(difference) # Uses covariance between both cells
```

Uncertainty from independently estimated anchors

The default intervals condition on the exact anchor values. Supplying `anchor_covariance` and `independent_anchors = TRUE` adds source-calibration uncertainty without changing any point estimates. Use the full covariance, not only a vector of SEs or a diagonal replacement for correlated anchors. Its numerical validity is checked; its statistical validity, units and correspondence to the supplied anchor values remain the analyst's responsibility.

Write the free receiving parameters as u, supplied anchors as a, and the negative log likelihood Hessian blocks as H_uu and H_ua. Implicit differentiation gives S = -inverse(H_uu) H_ua. For a reported target t(u,a), its source derivative is D = t_u S + t_a. The selected target covariance is t_u V_conditional t_u' + D V_anchor D'. The direct derivative t_a matters for targets containing anchored elements; adding variance only to the free parameters would omit that contribution and their cross-covariance. This is a first-order approximation for independent samples, not joint recalibration, a bootstrap or a correction for anchor drift or biased anchors.

Source derivatives use central differences of the analytic marginal score, checked by halving the difference step and refining the quadrature. Failed source-derivative checks preserve point estimates and the conditional results, with unavailable selected uncertainty. `ConditionalSE`, `ConditionalLower`, `ConditionalUpper` and `ConditionalStatus` retain that separate calculation; facet `Model*` columns also retain it. `AnchorSourceSE` is the square root of the added variance contribution, not a separate confidence interval. The selected `SE`, bounds and `vcov()` include both components. A conditionally fixed anchor may therefore have a selected interval for its source-estimated value. A zero covariance contribution does not create an interval for a target fixed under both specifications.

`anchor_uncertainty` retains the supplied and numerically symmetrized/PSD covariance, any floating-point adjustment, anchor values, sensitivity, independent-sample assumption and derivative checks. Singular PSD matrices are allowed. The free-parameter covariance includes source propagation; it does not include anchored values as extra optimizer coordinates. Saved population intervals can reuse this sensitivity and the conditional information: `mfrm_population_intervals(rater_ci)` propagates it once, without refitting or differentiating again. Reports and exported tables retain the supplied covariance and source-derivative checks; RDS retains the full result.

Only fixed-grid direct RSM/PCM MML with unit weights, no interactions, shrinkage or groups, an intercept-only population, fixed SD and all anchors on its noncentered facet is supported. Adaptive integration, free population variance, GPCM, dependent calibration samples and uncertainty in the supplied SD are outside this extension. No nominal coverage guarantee follows from numerical agreement or positive covariance. Other facets retain their centering; changing an item pool still requires a defensible scale link.

A complete interface example starts from bundled ratings. This split is illustrative: independent Persons and stable anchors must be justified for the actual sampling design, and different IDs alone do not establish them.

```
ratings <- load_mfrmr_data("example_core")
ids <- unique(ratings$Person)
source <- ratings[ratings$Person %in% ids[1:20], ]
```

```
receiving <- ratings[!ratings$Person %in% ids[1:20], ]
baseline <- fit_mfrm(source, "Person", c("Rater", "Criterion"), "Score",
  model = "RSM", noncenter_facet = "Rater", quad_points = 61)
baseline_ci <- mfrm_facet_intervals(baseline, "Rater")
common <- c("R01", "R02")
anchors <- data.frame(Facet = "Rater", Level = common,
  Anchor = baseline_ci$table$Estimate[match(common, baseline_ci$table$Target)])
cohort <- fit_mfrm(receiving, "Person", c("Rater", "Criterion"), "Score",
  model = "RSM", anchors = anchors, noncenter_facet = "Rater",
  population_formula = ~1, person_data = unique(receiving["Person"]),
  population_sd = 1, quad_points = 61)
rater_ci <- mfrm_facet_intervals(cohort, "Rater",
 anchor_covariance = vcov(baseline_ci)[common, common], independent_anchors = TRUE)
summary(rater_ci) # Selected and conditional intervals side by side
population_ci <- mfrm_population_intervals(rater_ci)
```

Small samples and singleton categories

A small sample or a wide interval is not, by itself, a reason to suppress an estimate or an otherwise available asymptotic interval. RSM/PCM model intervals can retain singleton categories (one observed response in a fitted or local category cell) when all declared categories are represented in the fitted and local scopes. They require fresh category review, source and convergence checks, positive unregularized joint information and a finer-quadrature score/covariance comparison. Empty categories, unsupported coordinates, known identification failures and failed numerical checks are not excused by this route. Failed interval checks retain the point with unavailable uncertainty and a reason.

The fit keeps its category-review state and can still have `InferenceReady = FALSE`. Inspect this interval object's `CIEligible`, `checks`, `numerical_checks` and `InferenceCaution` for the specific calculation. This does not qualify other analyses or guarantee small-sample coverage. The same model covariance is used by `vcov(fit)` and default `confint(fit)`. Singleton counts do not gain a sandwich or GPCM Wald extension here. For supported one-Person RSM/PCM calibration, model-information intervals may be available after the same numerical checks. Returned `cautions` and `InferenceCaution` state that calibration relies on one response vector and its assumed conditional independence; `CoverageQualified` is `FALSE`. A finite point or information matrix does not establish small-sample coverage or between-Person variation. One Person cannot provide sandwich uncertainty; its ratings must not be relabeled as independent Persons.

MML assumptions and numerical approximation

MML integrates each Person's joint response probability under the declared ability distribution. Model-information intervals require the response and population models to hold for the actual rating assignments and covariates, including local independence given ability. An overall normal ability distribution does not by itself justify using the same distribution within rosters selected according to ability. Response-dependent assignment, stopping or missingness requires a separate justification of the likelihood.

The regular asymptotic interpretation requires independent Persons, an identifiable fixed-dimensional marginal model, a consistent interior solution, smooth likelihood derivatives and information growing in every free direction. With bounded ratings per Person and stable design proportions, a

positive-definite limiting information per Person is a sufficient information-growth condition. Correctly specified MML can then support inference with fixed exposure per Person; increasing Person count alone cannot repair misspecification or an unidentified design. A positive observed Hessian at one fitted vector does not prove global identification.

The continuous-normal likelihood is evaluated numerically. A fixed finite quadrature order can retain approximation error as Person count grows. Optimizer and integration error must be negligible relative to sampling uncertainty for the continuous-model Wald interpretation. Agreement at two quadrature orders is a sensitivity diagnostic, not an error bound or a proof of asymptotic equivalence. Standardization does not remove either numerical approximation error or bias from a misspecified population model. The ordinary sandwich route can estimate variation around a working-model target under its own regularity conditions; that target need not be the true structural parameter.

Conditional likelihood

Native `fit_mfrm(method = "CML")` results use the inverse full conditional information, transformed through the fitted facet constraints and requested contrasts. Other facets and steps remain estimated. Only `method = "model"`, `clusters = NULL`, `adjust = FALSE` and `scale = "native"` are supported. CML estimates no ability population SD: `scale = "standardized"` cannot be computed from that fit. For MML with an estimated intercept-only normal population, that option divides by the same fit's estimated SD and includes its covariance with the contrast; see "Population-SD-standardized contrasts". A full CML fit can retain finite numerical point estimates when its identification, finite-MLE and stationarity checks pass but conditional information fails. Its non-fixed targets have `Status = "information_unavailable"`, missing SE/bounds, `CIEligible = FALSE` and the reason in `InferenceReview`. The fit's `checks$information_status` and `checks$information_reason` record the information decision. Constraint-fixed targets retain zero SE and no inferential interval. An information warning also limits confidence in the numerical accuracy of points along weak directions; retaining a point does not qualify it for inference. CML intervals are asymptotic, requiring the correct conditional response law, independent Persons and information growing in every free direction of a fixed-dimensional identified model. More Persons alone is insufficient; extreme totals add no information. Response-dependent assignment, stopping or missingness requires justification that the conditional law is preserved. See the sufficient regular regime in `fit_mfrm()`. Neither numerical acceptance nor a distribution-free ability treatment ensures small-sample coverage. `plot(ci)`, `summary(ci)` and RDS reuse apply to CML interval objects; attaching them to ordinary `mfrm_results()` is not supported. Non-fixed CML targets require finite, strictly positive representable variance. Rescale contrast coefficients if their units cause variance underflow or overflow; numerical zero variance is not a constraint-fixed target.

For `fit_scope = "comparisons"`, call `mfrm_facet_intervals(fit)` without new `contrasts` or `comparison_review`. The saved estimates, joint target covariance and qualifications are reused without optimization or information recalculation. `level` changes only the pointwise normal critical value. Unidentified, boundary or unqualified-stationarity comparisons keep missing estimates and intervals. If only information failed, identified numerical points remain, with missing uncertainty, their reasons and `CIEligible = FALSE`; no unavailable value is zero-filled. `parameter_covariance` is the covariance of the fit's identified subspace, not of separately identified cell effects. Use `summary(ci)`, `plot(ci)`, `apa_table(ci)` and `saveRDS(ci)`; the saved review retains group order, contrast weights and observed/empty cells. JML/MML comparison-only fits remain descriptive and are not covered by this interval route.

Anchors and group means

The ordinary fixed-population RSM/PCM route supports location anchors and group means. Write the fitted facet locations as $b = b_0 + J\eta$ and a requested comparison as Cb. Its covariance is $CJV_\eta J^\mathsf{T} C^\mathsf{T}$, where V_η is extracted after computing the full joint covariance. Other facets and thresholds remain estimated; inverting only the selected facet's information would omit their estimation uncertainty. The affine offset b_0 affects estimates, but has no variance when its supplied values are treated as exact.

A comparison with $CJ = 0$ is fixed: its `Status` is `"fixed"`, `SE` is zero, and interval bounds are missing. This includes values implied by centering, not only rows in an anchor table. For example, with four raters, whole-facet centering and a fixed mean of 0.10 for the first three, the fourth rater is fixed at -0.30. The three group members can still differ unless other constraints fix them; their mean has no inferential interval. A difference between a free rater and an anchored rater can remain uncertain. These are conditional statements, not evidence that the source calibration or group-mean assumption is error-free. Native-location anchors use only the fixed-SD RSM/PCM extension below; groups remain excluded there.

Native locations

For estimated normal RSM/PCM and intercept-only normal one-family GPCM MML, `method = "model"` supplies pointwise Wald intervals for locations or contrasts on the model's native ability scale. One-family GPCM also admits the explicit fixed-N(0,1) restriction (`gpcm_mml_identification = "fixed_standard_normal"`). Shared or separate slope/step owners are supported, with geometric-mean-one relative slopes. Use direct MML, fixed or adaptive integration, unit weights, additive centered facets and centered steps. RSM/PCM also accepts a fitted full-rank conditional-normal regression, retaining its coefficient and residual-variance cross-information when computing structural uncertainty. The saved population table identifies coefficients and residual variance; it does not label regression coefficients as marginal population means. GPCM latent regression and sandwich covariance remain unavailable here. The native location procedure performs its own qualification even when the source fit has `InferenceReady = FALSE`. Read `CIEligible`, `Status`, `checks`, `numerical_checks` and cautions on this result; interval availability does not change the source fit's overall readiness or qualify other procedures. Interactions use the separate `comparison_review` route below. Population-coefficient and variance intervals are separate targets and are not returned by this helper. RSM/PCM populations may fix residual SD with `population_sd`. For an intercept-only population (`population_formula = ~1`), a direct anchor on the selected non-Person `noncenter_facet` identifies the origin while free levels on that facet remain uncentered. These anchored intervals require no groups or dummy facets and condition on the supplied anchors and SD; their source-calibration uncertainty is not propagated. Other facets keep their declared centered constraints. Fixed SD has no variance coordinate in the joint information.

The covariance transforms the inverse full joint marginal information, including all estimated population, slope and step coordinates. A location-only inverse or independent printed SEs cannot replace it. The source, category support, local solution and unregularized information must pass checks; covariance and score changes are also checked at a finer integration order at the same fitted parameter vector. This model-information procedure has no empirical Person-score rank requirement. For RSM/PCM populations with estimated variance, a zero-residual-variance candidate at the retained nuisance estimates is compared with the fitted likelihood. A higher boundary likelihood beyond the numerical allowance prevents intervals. This check does not optimize or exclude the entire boundary. A numerical refusal preserves the native point and its reason with unavailable SE/bounds. Constraint-fixed targets have no inferential interval.

The asymptotic justification requires independent Persons, a correctly specified response and population model, an identified interior true parameter, a fixed number of structural parameters, nonsingular limiting information and sufficiently accurate integration and optimization. Information must grow in all target directions as Persons are added; connected facet labels alone do not establish this condition. These numerical checks do not establish finite-sample coverage, global identification or robustness to misspecification. Native locations are not divided by the estimated population SD unless `scale = "standardized"` is requested for a zero-sum contrast (see below). A location difference need not produce a uniform rating difference with unequal slopes or steps. The calculation does not refit or change the saved fit/readiness. Save and attach `ci` with `mfrm_results(fit, intervals = list(locations = ci), compute = "never")`; select `plot(res, type = "facet_locations")` for RSM/PCM or `"gpcm_locations"` for one-family GPCM. Reports and exports retain the native scale, inferential qualifications and unavailable outputs.

Population-SD-standardized contrasts

Set `scale = "standardized"` and supply explicit contrast rows whose coefficients sum to zero. For native contrast d, population SD σ and free log-variance $\lambda = \log(\sigma^2)$, the target is $t = d/\sigma$. Its Jacobian has location entries J_d/σ and log-variance entry $-t/2$. The covariance is transformed from the inverse full joint observed information, including the covariance between the contrast and log-variance. Dividing native interval endpoints by the fitted SD omits this uncertainty. In scalar form,

$$\widehat{\mathrm{Var}}(\hat{t}) = \frac{\widehat{\mathrm{Var}}(\hat{d}) + \hat{d}^2\widehat{\mathrm{Var}}(\hat{\lambda})/4 - \hat{d}\widehat{\mathrm{Cov}}(\hat{d},\hat{\lambda})}{\hat{\sigma}^2}.$$

The covariance term can increase or decrease the SE relative to dividing the native SE by the fitted SD. A zero estimated contrast can still have positive variance. With fixed N(0,1), the result equals the native contrast because the known SD is one. With an explicit fixed `population_sd`, divide the native contrast and SE by that SD; no estimated-SD derivative or uncertainty is introduced. Conditional-normal regression is excluded: its residual SD is not a marginal population SD. Standardization does not equate expected ratings across models with different slopes or category steps.

The existing source and numerical checks still apply. These are pointwise normal delta-method intervals, not profile intervals or finite-sample coverage guarantees. The regular delta method assumes a fixed true population SD strictly above zero and an identifiable, interior joint solution under the stated marginal model. A positive fitted SD alone does not establish these conditions. At zero SD the target is undefined; near zero the ratio and normal approximation can be unstable. No universal SD cutoff certifies validity. Standardization does not resolve weak identification or boundary problems; see the MML assumptions above.

MML interaction contrasts

Supply `comparison_review` to obtain native or population-SD-standardized intervals for its explicitly weighted facet-by-origin interaction contrasts. The review must match the fitted responses, group identities, constraints, model, PCM step owner, signs and population design. Its model has one interaction, zero origin main facet and centered remaining facets and interaction margins. This extension requires a full, unadjusted RSM/PCM direct MML fit, unit weights and preserved categories, without shrinkage or additional anchors. Comparison-only MML fits, JML and GPCM are not admitted. CML comparison intervals remain available through `fit_mfrm()` but cannot estimate a population SD.

Full conditional-map identification is necessary here. Source tables, categories, local solution, unregularized full information and finer-grid stability must pass the native-location checks above. A failed numerical check retains the point estimate with missing SE/bounds and a reason. Positive conditional rank and local information do not establish global marginal identification or finite-sample interval coverage; the output retains `MarginalIdentificationState` and `CoverageQualified = FALSE`. These are model-based asymptotic intervals, not a calibrated DFF classification.

For `scale = "standardized"`, the denominator is the SD estimated in the **same intercept-only normal MML fit** (`population_formula = ~1`). Such a fit assumes one common ability distribution across the declared origin groups. It is a different model from `~Group`, which allows group means. For `~Group`, native interaction intervals are supported, but its residual SD is not substituted for a marginal population SD. A fixed N(0,1) fit has known SD one; no estimated-SD uncertainty is introduced in that case.

The joint delta method uses interaction derivatives divided by the fitted SD and derivative `-Estimate/2` for log population variance. All population, step and structural nuisance coordinates remain in the full information inverse. It includes the covariance between the numerator and denominator; dividing the native endpoints or SE by the fitted SD is insufficient.

Read `table` with `comparison_review`: group subtraction order, panel weights, observed/empty cells and assumptions are retained. Use `plot(ci)`, `saveRDS(ci)` to retain `IdentificationBasis` and its explanation with the intervals. A free-Person response-map basis does not establish distribution-free MML inference; population-dependent conditional identification does not establish global marginal identification. Display the result directly or attach it as `mfrm_results(fit, intervals = list(school = ci), compute = "never")`. Plotting and reports reuse the saved uncertainty; they do not refit or recalculate information. These results do not extend the separate `mfrm_population_intervals()` consumer to models with fitted interactions.

Two-family location targets

For native two-family GPCM MML, select either modeled non-Person facet. The first slope owner's locations sum to zero; the second owner's locations are uncentered on the fixed N(0,1) ability scale. These locations are measured before multiplication by both slope components. Increasing a location lowers expected ratings when ability, slopes and other effects stay fixed. A contrast between actual raters need not imply a uniform difference in expected ratings: their slopes and category-step offsets may also differ. Neither zero nor a location difference defines rater quality or accuracy.

Let `J` expand the free location coordinates under their fitted constraints and `C` be the requested contrast matrix. The target covariance is `C J V J' C'`, where `V` is the location block extracted *after inverting the full joint marginal information*. Slopes, steps and the other facet remain estimated nuisance parameters. Inverting a location-only Hessian, adding independent printed SEs, or using the EM conditional-objective Hessian would omit their joint estimation uncertainty.

The source fit must pass the same identity, category support, score derivative, stationarity, full-information and finer-quadrature checks used for experimental two-family slope intervals in `confint.mfrm_fit()`. The full covariance is checked; deficient empirical Person-score rank is a retained diagnostic and warning, not a veto on the inverse-observed-information method. Passing these numerical checks does not establish repeated-sample coverage for locations or contrasts. Evidence for component slope intervals does not qualify these different targets. These are reported separately as `CIEligible` (numerical interval availability) and `CoverageQualified = FALSE` in `table`. The latter means that nominal sampling coverage has not been established for the target and design; it is neither a coverage estimate of zero nor a numerical-failure flag. Saving, reporting or exporting the result retains

both fields. These are experimental pointwise normal approximations, with no sandwich, profile, multiplicity or automatic significance decision. A failed check retains estimates and reasons with unavailable intervals, even if the coarse-grid information could be inverted. Constraint-fixed targets have no interval. The calculation does not refit or change quadrature in the saved fit.

For example, use `ci <- mfrm_facet_intervals(fit, "Rater")`, or supply a named zero-sum row of `contrasts` for a difference. Plot and save `ci` directly, or attach it with `mfrm_results(fit, intervals = list(locations = ci))` and select `plot(res, type = "gpcm_locations")`. The model must actually contain the specified facet. Location intervals do not add category-step/curve bands, Person uncertainty, Wright maps or tests of rater competence. Individual two-family rater sheets currently display component-slope intervals only; use the analyst report for these location and contrast results.

What robustness means here

Under model misspecification, the sandwich describes sampling variation around the working model's limiting parameter (its pseudo-true target). That target need not equal the generating rater severity or criterion difficulty. Changing the SE does not correct a biased estimate, informative assignment, unmodeled population differences or MNAR nonresponse. Check the model and assignment before interpreting a severity contrast.

These intervals condition on the observed fixed facet levels. They do not generalize to replacement raters sampled from a rater population. They are not multiway crossed-cluster, G/D-study, variance-boundary, EAP or multiple-imputation intervals. Shared random rater/task effects spanning the declared clusters violate this one-way independence assumption. Numerical success is not a general coverage or rater-diagnosis guarantee. Review quadrature sensitivity separately; the helper reuses the fitted grid.

Bounded evaluation

For the RSM/PCM route, a 1,600-dataset comparison used 80/320 independent persons, three fixed raters and two criteria. In its combined skewed-ability/sparse-design scenario, sandwich coverage of generating contrasts ranged from 87.0 to 95.5 percent despite all intervals being available. Coverage of the independently calculated working-model targets ranged from 93.0 to 97.5 percent. These are ranges across conditions/contrasts, not uncertainty bounds or universal operating characteristics. At 200 datasets per condition, MCSE near 95 percent is about 1.54 percentage points. See `vignette("mfrmr-facet-intervals", package = "mfrmr")` for the design, interpretation and a complete rater-feedback example.

Save, display and report the selected intervals

Use `plot()`, `as_ggplot()` and `plot_data()` to display or extract the saved result, and `apa_table()` for tables. Set `title = NULL`, `subtitle = NULL` or `caption = NULL` in the plot to omit that text. For supported RSM/PCM MML fits, attach one result or a named list to `mfrm_results()`, for example `mfrm_results(fit, intervals = list(raters = intervals), include = c("fit", "plots"), compute = "never")`. The route `plot(res, type = "facet_raters")` shows the selected intervals. For two-family GPCM, the corresponding plot type is `"gpcm_raters"`. `mfrm_report()` and `export_mfrm_results()` retain the method, level, contrast coefficients, cluster mapping and unavailable outcomes. The source fit must match; replay reloads the saved results without refitting or recomputing covariance. These intervals do not replace uncertainty in ordinary Wright maps, fit diagnostics or Person scores. For CML, keep the interval object itself: use `summary(intervals)`,

`plot(intervals)`, `apa_table(intervals)` and `saveRDS(intervals, file)`. Reopen it with `readRDS(file)`. Ordinary `mfrm_results()`, `mfrm_report()` and `export_mfrm_results()` do not accept CML fits or their intervals.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

References

White, H. (1982). Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1), 1–25. Zeileis, A. (2006). Object-oriented computation of sandwich estimators. *Journal of Statistical Software*, 16(9), 1–16. doi:10.18637/jss.v016.i09. Zeileis, A., Koell, S. and Graham, N. (2020). Various versatile variances: An object-oriented implementation of clustered covariances in R. *Journal of Statistical Software*, 95(1), 1–36. doi:10.18637/jss.v095.i01.

See Also

`analyze_facet_equivalence()`, `pool_mfrm_imputed()`, `plot.mfrm_facet_intervals()`, `mfrm_population_interval`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrm_facet_intervals", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  model = "RSM", method = "MML", rating_min = 1, rating_max = 4,
  category_policy = "preserve")
intervals <- mfrm_facet_intervals(fit, "Rater", method = "sandwich")
summary(intervals)
```

```
plot(intervals)

# Compare two selected raters, rather than each rater against the origin.
# Here choose the first two example labels; prespecify your own comparison.
raters <- unique(as.character(ratings$Rater))
contrast <- matrix(0, 1, length(raters), dimnames = list(
  paste(raters[1], "minus", raters[2]), raters))
contrast[1, 1:2] <- c(1, -1)
difference <- mfrm_facet_intervals(fit, "Rater", contrasts = contrast)
summary(difference)[, c("Target", "Estimate", "SE", "Lower", "Upper", "Status")]
# With the default negative rater sign, a positive contrast means the first
# rater is stricter. Units are native logits. An interval containing zero
# leaves the direction uncertain; it does not establish equivalence.

# Estimate the population SD in this same model before standardizing.
# Even an intercept-only population model needs one row per person.
persons <- unique(ratings["Person"])
population_fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"),
  "Score", model = "RSM", method = "MML", population_formula = ~ 1,
  person_data = persons, person_id = "Person",
  rating_min = 1, rating_max = 4, category_policy = "preserve", quad_points = 61)
standardized <- mfrm_facet_intervals(population_fit, "Rater",
  contrasts = contrast, scale = "standardized")
summary(standardized)
# Estimate is now in fitted ability-population SD units; its delta-method
# interval includes uncertainty in that SD. It is not a Wald z statistic.

# CML estimates the native contrast without an ability distribution.
if (requireNamespace("lpSolve", quietly = TRUE)) {
  cml_fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
    model = "RSM", method = "CML", rating_min = 1, rating_max = 4,
    category_policy = "preserve")
  cml_ci <- mfrm_facet_intervals(cml_fit, "Rater", contrasts = contrast)
  summary(cml_ci)
  apa_table(cml_ci)
  file <- tempfile(fileext = ".rds") # Choose a lasting path for your analysis
  saveRDS(cml_ci, file)
  saved <- readRDS(file)
  summary(saved) # Same conditional-information result, without refitting
  # Use plot(saved) for a figure; ordinary mfrm_results() does not accept CML.
}

## End(Not run)
```

`mfrm_features` *Prepare external features for exploratory grouping*

Description

Keep one row per Person, rater, or other entity, review feature availability, and optionally attach user-supplied reasons for missing values.

Usage

```
mfrm_features(data, id, features, missing_reasons = NULL)

## S3 method for class 'mfrm_features'
print(x, ...)

## S3 method for class 'mfrm_features'
summary(object, ...)
```

Arguments

data	A data frame with one row per entity. Repeated rating rows must be summarized or joined to an entity-level table before calling this function.
id	Name of the unique, nonmissing identifier column. Character, factor, or finite numeric identifiers are stored as character strings without trimming.
features	Explicit character vector of feature column names, excluding the identifier. Numeric, character, factor, ordered factor, and logical columns are supported. Character columns become nominal factors; logical columns represent symmetric binary features. Ordered factor levels retain their declared order. Dates, lists, and matrices must be converted explicitly.
missing_reasons	Optional data frame with columns ID, Feature, and Reason, one row per annotated missing cell. IDs refer to id; features must be selected columns. Reasons for observed or unknown cells are refused. Unannotated missing cells are labelled "Not supplied". Reasons are supplied by the user, not inferred missing-data mechanisms.
x, object	An object returned by the corresponding function.
...	Reserved for method compatibility.

Details

Numeric NA and NaN are missing. Infinite numeric values and blank categorical labels are refused; replace missing markers with NA explicitly. Constant and entirely missing features remain available for review but cannot be used by mfrm_cluster_pam(). IDs and unselected columns do not enter distances. These functions are intended for external attributes such as training, experience, or specialization. They do not propagate uncertainty from estimated ability, severity, or fit statistics. See vignette("mfrmr-external-features") for a complete rater-attribute example including missingness review and multiple imputation.

Value

An mfrm_features object containing data, id, features, a feature_summary, a row_summary, and missing (all missing cells and their reasons). No values are imputed or rows discarded.

See Also

mfrm_cluster_pam(), mfrm_cluster_imputed(), mfrm_cluster_compare(), mfrm_pca(), mfrm_cluster_kmeans()

Examples

```
# Fictional rater attributes; experience is measured in completed years.
raters <- data.frame(
  Rater = paste0("R", 1:6),
  ExperienceYears = c(1, NA, 3, 12, 13, 14),
  Specialty = c("Language", "Language", "Language", "Science", "Science", "Science")
)
reasons <- data.frame(ID = "R2", Feature = "ExperienceYears", Reason = "Not recorded")
features <- mfrm_features(raters, "Rater", c("ExperienceYears", "Specialty"),
                          missing_reasons = reasons)
summary(features)
features$missing
features$row_summary
```

---

mfrm_fit_extractors *Extract coefficients, covariance, likelihood and observation counts*

Description

Standard R accessors for native many-facet fits. They retain the estimator's coordinates and limitations; extracting a number does not qualify inference.

Usage

```
## S3 method for class 'mfrm_fit'
coef(object, type = c("free", "locations", "steps", "slopes", "persons"), ...)

## S3 method for class 'mfrm_cml_fit'
coef(object, type = c("free", "locations", "steps", "slopes", "persons"), ...)

## S3 method for class 'mfrm_comparison_fit'
coef(object, ...)

## S3 method for class 'mfrm_fit'
vcov(object, ...)

## S3 method for class 'mfrm_comparison_fit'
vcov(object, ...)

## S3 method for class 'mfrm_fit'
nobs(object, type = c("likelihood", "persons", "ratings"), ...)

## S3 method for class 'mfrm_cml_fit'
nobs(object, type = c("likelihood", "persons", "ratings"), ...)

## S3 method for class 'mfrm_comparison_fit'
nobs(object, type = c("likelihood", "persons", "ratings"), ...)
```

```
## S3 method for class 'mfrm_fit'
logLik(object, ...)

## S3 method for class 'mfrm_comparison_fit'
logLik(object, ...)

## S3 method for class 'mfrm_fit'
AIC(object, ..., k = 2)

## S3 method for class 'mfrm_fit'
BIC(object, ...)

## S3 method for class 'mfrm_cml_fit'
AIC(object, ..., k = 2)

## S3 method for class 'mfrm_cml_fit'
BIC(object, ...)

## S3 method for class 'mfrm_cml_fit'
vcov(object, ...)

## S3 method for class 'mfrm_cml_fit'
logLik(object, ...)
```

Arguments

object	A native result from fit_mfrm(). Imported and extended-model results use their documented summaries and model-specific methods.
type	For coef(), "free" (default) returns the independent fitted coordinates; "locations", "steps", "slopes" or "persons" returns the corresponding saved primary estimates. For nobs(), "likelihood" (default) counts Persons for MML/CML and retained response rows for JML; "persons" and "ratings" request those counts explicitly.
...	Unused for coefficient, covariance, likelihood and count extraction; misspelled arguments are rejected. Multiple-model calls to AIC()/BIC() are refused: use compare_mfrm() for supported comparisons with matched observations, estimation and integration settings.
k	Penalty per free parameter for single-fit AIC(); default 2.

Details

Coordinates. The default coef(fit) is a named numeric vector aligned with both dimensions of vcov(fit). These are free coordinates, not every displayed facet level: anchors and dependent centered coordinates are omitted. MML/JML names such as Rater[1] identify positions within parameter blocks, not literal level names. Steps are free step offsets, slopes are log slopes, and estimated population variance is on the log-variance scale. Population regression coefficients remain on their fitted design scale. CML retains its existing conditional-coordinate names. Use type

= "locations" for all saved facet locations with their level labels, and summary(fit) for tables, readiness, constraints and interpretation. Location signs are the fitted model's signs; this accessor does not reorient them as severity. Default confint() intervals use these same coordinates; GPCM positive-slope intervals require explicit parm = "slopes".

type = "persons" returns saved primary Person estimates: MML EAPs and JML point estimates are different targets. Person posterior SDs are not structural standard errors and do not occur in vcov(fit). CML has no Person estimates. Free-coordinate extraction refuses corrected or penalized JML and any fit whose primary estimates are nonfinite: finite optimizer proxies must not replace primary boundary values. Explicit component extraction preserves the saved infinite or missing primary estimates.

Covariance. MML vcov() reuses the same qualified full joint covariance as mfrm_facet_intervals(), including its model/design restrictions and numerical checks. It may reevaluate information and integration, but does not refit. An ineligible source remains an error. For an otherwise eligible ordinary fixed-population RSM/PCM fit, information failure returns an NA matrix with unavailable_reason, checks and cautions; default confint() retains coefficient estimates and reasons in its diagnostics attribute while bounds are missing. No regularized inverse is substituted. If even one dense NA matrix exceeds mfrmr.max_information_bytes, vcov() errors with that reason; use the smaller target-level mfrm_facet_intervals() result. Other MML routes retain their separate qualification checks and error rules. RSM/PCM singleton-category fits can pass the interval-specific information review while retaining InferenceReady = FALSE; returned cautions and checks explain the small-sample limitation. No count label alone certifies or invalidates this covariance. No JML structural-truth covariance is supplied. CML vcov() reuses saved conditional information. If CML's finite numerical point passed its identification, finite-MLE and stationarity checks but information failed, vcov() returns an NA matrix with unavailable_reason, checks and cautions attributes. coef() and logLik() still expose the retained values; default confint() supplies missing bounds with diagnostic reasons. The warning also limits confidence in point accuracy along weak directions; no regularized covariance or formal inference is substituted. Free-coordinate covariance is not the covariance of a selected component vector or of all displayed levels; use mfrm_facet_intervals() or confint.mfrm_fit() for supported transformations. Comparison-only coef() returns the declared contrast estimates, retaining unavailable rows as NA; its vcov() is available only for native CML conditional comparisons, with the same row/column order.

Likelihood and counts. logLik() returns the retained MML marginal or ordinary JML joint log likelihood, with likelihood_basis and ICReason attributes. Nonunit weights give a weighted objective, not an ordinary sampling likelihood. The df attribute is the free parameter count only when the current information-criterion contract and solution checks permit its use; otherwise it is NA, so AIC() and BIC() are also NA. parameter_count retains the actual dimension separately. JML and two-family GPCM do not gain information-criterion eligibility through these accessors. Corrected JML, penalized objectives and comparison-only fits do not supply a generic logLik() here. For estimated-population/one-family GPCM MML, likelihood qualification may reevaluate solution information.

CML retains its conditional log likelihood and free dimension. Its nobs attribute counts all retained Persons, including extreme totals contributing zero conditional information; informative_persons records nonextreme totals separately. This preserves the existing CML method and is not a new validation of conditional information criteria. Conditional and marginal likelihoods must not be compared. For MML, BIC uses Persons, not rating rows or the sum of rating weights. Counts describe the retained analysis, not omitted input records or an effective sample size under dependence. Use nobs(fit, type = "ratings") to count retained ratings.

Value

coef() returns a named numeric vector; vcov() a matrix in the matching default coordinates. MML covariance retains checks, numerical_checks, information_review and cautions attributes when supplied by the interval route. logLik() returns a logLik object; nobs() an integer count; single-fit AIC()/BIC() a numeric value or NA when unsupported. Existing CML covariance coordinates and numeric values are unchanged. No accessor silently computes new Person scores.

See Also

fit_mfrm(), summary.mfrm_fit(), mfrm_fit_responses, mfrm_facet_intervals(), compare_mfrm(), mfrmr_compatibility_layer

Examples

```
ratings <- load_mfrmr_data("example_core")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 31, maxit = 500)
coef(fit, type = "locations") # Labelled facet locations, including constraints
coef(fit)                     # Independent coordinates, including steps
nobs(fit)                     # Persons contributing to marginal likelihood
nobs(fit, type = "ratings")    # Retained response rows
ll <- logLik(fit)
attr(ll, "ICReason")          # Read before interpreting AIC/BIC
AIC(fit)
# When covariance is qualified, its names match the free coefficients:
# V <- vcov(fit); identical(rownames(V), names(coef(fit)))
```

---

mfrm_fit_responses *Extract fitted rating values and residuals*

Description

Describe the retained ratings at the fitted calibration and saved Person estimates, without fitting, scoring Persons again or computing covariance.

Usage

```
## S3 method for class 'mfrm_fit'
fitted(object, type = c("response", "prob"), ...)

## S3 method for class 'mfrm_fit'
residuals(object, type = c("response", "pearson"), ...)

## S3 method for class 'mfrm_fit'
predict(object, newdata = NULL, type = c("response", "prob"), ...)

## S3 method for class 'mfrm_cml_fit'
fitted(object, type = c("response", "prob"), ...)
```

```
## S3 method for class 'mfrm_comparison_fit'
fitted(object, type = c("response", "prob"), ...)

## S3 method for class 'mfrm_cml_fit'
residuals(object, type = c("response", "pearson"), ...)

## S3 method for class 'mfrm_comparison_fit'
residuals(object, type = c("response", "pearson"), ...)

## S3 method for class 'mfrm_cml_fit'
predict(object, newdata = NULL, type = c("response", "prob"), ...)

## S3 method for class 'mfrm_comparison_fit'
predict(object, newdata = NULL, type = c("response", "prob"), ...)
```

Arguments

`object`	A native ordinary RSM/PCM or one-family GPCM fit from `fit_mfrm()`, estimated by MML or uncorrected JML.
`type`	For `fitted()` and `predict()`, `"response"` returns expected ratings and `"prob"` category probabilities. For `residuals()`, `"response"` returns observed minus expected ratings and `"pearson"` divides that difference by the conditional rating SD.
`...`	Unused; additional arguments are rejected. Prediction intervals, standard errors, new rating designs and missing-score imputation are not supplied by these methods.
`newdata`	Must be `NULL`. These methods describe retained source responses only. For ability scoring from supplied responses use `predict_mfrm_units()` or `score_mfrm_calibration()`; those return Person abilities, not predicted rating values. `score_mfrm_persons()` supplies conditional source-Person scoring for its documented model scope.

Details

Target. Probabilities are evaluated at saved primary Person estimates: MML uses the saved EAP, while JML uses its primary point estimate. Thus `fitted(fit)` is a conditional plug-in expectation. Evaluating probabilities at an EAP is different from averaging probabilities over a posterior. For the latter target, use `mfrm_response_diagnostics()` within its stated model/design scope. These methods do not change `diagnose_mfrm()` or its legacy diagnostic conventions. Legacy diagnostic helpers use finite JML optimizer proxies and omit variances at or below `1e-10`; the primary boundary limits and small positive variances here can therefore give different residuals. Their observation tables retain `PersonMeasureBasis`, `PrimaryPersonMeasure`, `ConditionalVariance` and `ResponseReason` to explain these differences. Do not assume the routes agree at these limits.

For model-score categories k, let p_{ik} be the conditional probability. The mean is $\mu_i = \sum_k k p_{ik}$, the variance is $V_i = \sum_k (k - \mu_i)^2 p_{ik}$, and the Pearson residual is $(y_i - \mu_i)/\sqrt{V_i}$. Rating weights affect estimation but are not multiplied into these per-rating residuals or probabilities. The variance has no artificial positive floor. A zero variance leaves the mean/raw residual available but the Pearson residual missing, with a reason.

A positive/negative infinite JML Person estimate with finite structural calibration has limiting probability one at the maximum/minimum category. These limits use the primary estimate, not the finite optimizer proxy. Other unavailable Person estimates leave the corresponding rows missing. Nonfinite structural estimates, corrected or penalized fits, two-family GPCM, imported fits, CML and comparison-only fits are explicitly refused. CML eliminates Person abilities and does not supply this plug-in target. The supported posterior or corrected-JML routes remain available through `mfrm_response_diagnostics()`.

Rows and scale. Output order is `fit$prep$data` order, including repeated rating events, and excludes records omitted before fitting. Vector names and probability-matrix row names are retained-row positions. The `rows` attribute records these positions and original input-row positions where retained metadata permit recovery; unknown positions are `NA`. No response is silently added or dropped during extraction. The separate `identifiers` attribute retains Person/facet columns in the same order, including user-supplied facet names that coincide with `Row`.

Values use the modeled score scale (`InternalScore` in `score_map`), as in ordinary plug-in diagnostics. With collapsed categories this differs from the original score labels. The saved `score_map` attribute retains that mapping; probability columns name the modeled categories. Inspect the map before comparing fitted values with an original, unrecoded data column.

Interpretation. These same-data summaries hold calibration and Person estimates fixed. They omit their estimation uncertainty and are not held-out forecasts, calibrated goodness-of-fit tests or evidence of rater quality. Point calculation does not establish convergence or inference readiness. Read `source_readiness` and row-specific `diagnostics` before interpreting values, and keep these attributes when saving the result.

Value

A named numeric vector, or for `type = "prob"` a numeric matrix with one column per modeled category. Attributes include `rows`, `identifiers`, `score_map`, `diagnostics` (availability, conditional variance and reason), `settings` (quantity, target, Person basis and limitations) and `source_readiness`. `predict(fit)` and `fitted(fit)` return the same source-rating target.

See Also

mfrm_fit_extractors, `diagnose_mfrm()`, `mfrm_response_diagnostics()`, `predict_mfrm_units()`, mfrmr_compatibility_layer

Examples

```
ratings <- load_mfrmr_data("example_core")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 31, maxit = 500)
expected <- fitted(fit) # Expected ratings at saved EAP abilities
head(data.frame(Observed = fit$prep$data$Score, Expected = as.numeric(expected),
  Residual = as.numeric(residuals(fit))))
head(attr(expected, "diagnostics")) # Check unavailable and zero-variance rows
attr(expected, "settings")         # Read the target and interpretation limits
head(fitted(fit, type = "prob"))    # One column per modeled score category
```

`mfrm_generalizability` *Main-effects generalizability approximation for an MFRM design*

Description

Re-fits the rating data underlying an `mfrm_fit` as a crossed random-effects model `Score ~ 1 + (1 | Person) + (1 | Facet1) + ... + Residual` via `lme4::lmer`, and returns main-effect variance components, residual variance, and G / Phi coefficients under this simplified model. Interactions are not estimated separately. For a G-study including the supported interactions, use `mfrm_g_study()` with one or two common random facets; it also accepts a single score column.

Usage

```
mfrm_generalizability(
  fit,
  data = NULL,
  object_facet = "Person",
  random_facets = NULL,
  reml = TRUE,
  missing = c("error", "omit")
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`data`	Optional data frame. When `NULL`, the rating data stored on `fit$prep$data` is used. Required columns are the selected facets and `Score`. Scores must be numeric or numeric character/factor labels; nonnumeric labels and infinite values are refused. Use `NA` for missing values. Facet labels must be nonblank; `Score` and `Residual` are reserved and cannot be facet names.
`object_facet`	Facet that plays the role of the "object of measurement" – typically `"Person"` (default).
`random_facets`	Character vector of other facets to treat as random conditions of measurement. The default uses `Person` and the configured MFRM facets, excluding `object_facet`. For example, when raters are the objects, persons remain a random condition.
`reml`	Logical, passed to `lme4::lmer()` (default `TRUE`).
`missing`	Either `"error"` (default) or `"omit"`. Missing scores or selected facet values stop the analysis by default. Explicit omission fits only complete rows and records the excluded row positions and missing columns. Missingness in unselected columns does not exclude a row.

Details

The decomposition is on the observed numeric `Score` scale. It does not use the fitted MFRM latent scale or estimate a latent ordinal G/Phi coefficient.

Value

An object of class `mfrm_generalizability` with:

`variance_components` One row per random effect plus residual, with columns `Source`, `Variance`, `ProportionVariance`, and `Identified`. When `Identified` is false, the fitted allocation and its proportion are not uniquely determined by the source design.

`coefficients` One-row data frame with `G` (generalizability coefficient, relative decision) and `Phi` (dependability coefficient, absolute decision), coefficient status labels, and the identification status of the fitted random-effects model. `GReason` and `PhiReason` explain missing coefficients; an empty reason means the point is calculable. `ComponentIdentificationChecked` records the design check.

`design` Description of the crossed-random model. `component_identification` retains the covariance-kernel Gram matrix, rank, scaled eigenvalues, null space and numerical tolerance. `fit_status` retains the separate lme4 boundary/convergence assessment.

`data_usage` Input source, omission policy, named `counts` (`InputRows`, `UsedRows`, `ExcludedRows`), `excluded_rows`, and `missing_cells` (`InputRow`, `Column`). Row positions refer to the supplied data, or stored fitted rows when `data = NULL`. They cannot recover rows previously removed during MFRM fitting. Counts also accompany the coefficient table and D-study projections, including tabular exports; `GStudyDataSource` identifies the scope of those counts.

Interpretation

- `G` is appropriate for **relative** decisions (rank-ordering persons): `G = sigma2(p) / (sigma2(p) + sigma2(Residual))`.
- The reported `Phi` describes dependability for **absolute** decisions: `Phi = sigma2(p) / (sigma2(p) + sigma2(facet ma` before D-study scaling. It does not estimate the probability of correct classification at a particular cut score.
- Use `mfrm_d_study()` to project `G` / `Phi` under planned numbers of raters, items, criteria, or other random measurement facets.
- Values of 0.70 and 0.80 are displayed as familiar planning references, not as universal decision rules. Required dependability depends on the decision, consequences, population, and evidence beyond a single coefficient.
- For ordered categories, `Score` is treated as a numeric observed response in a Gaussian linear mixed model; thresholding is not modeled.

Component identification and boundary estimates

A zero variance estimate can be a valid constrained ML/REML solution. It is distinct from a design that cannot separate two variance components. For example, if every rater identifier always corresponds to the same task identifier, their main-effect covariance kernels coincide. Their sum may be identified while the individual allocations are not. `isSingular()` alone does not test this design property.

The design check uses linear independence of the observation covariance kernels (after projecting out the intercept for REML). Only variance combinations orthogonal to the numerical null space enter coefficients. Thus an identified sum can still supply Phi, while a future plan assigning unequal weights to its confounded components cannot. Individual fitted allocations remain available with

`Identified = FALSE` for diagnosis. A full numerical rank does not establish correct model assumptions, optimizer accuracy or finite-sample precision. This is a linear design check; fitted boundary zeros are not used to resolve otherwise confounded components.

If object variance is zero and the relevant error variance is positive, G or Phi equals zero and is retained with boundary diagnostics. A zero total variance gives an undefined ratio and `zero_total_variance`. The historical `IdentificationStatus` field summarizes design and fit cautions; inspect `design$component_identificati` and `design$fit_status` to distinguish them. This function does not supply coefficient intervals.

Limitations

This helper formulates the random-effects model with main effects only (`Score ~ 1 + (1|Person) + (1|Facet1) + ... + Residual`); no explicit `(1 | Person:Rater)`, `(1 | Person:Criterion)`, or `(1 | Rater:Criterion)` interaction terms are estimated. All interactions are omitted. The residual combines unexplained variation; omitted interactions may also affect the fitted main-effect components. Their separate variances and correct averaging rates are not recovered. This function reports the one-observation-per-cell baseline. `mfrm_d_study()` applies D-study scaling, including residual-scaling sensitivity checks, to the same simplified variance-component decomposition. Because person-by-facet interaction terms are not estimated separately, D-study projections remain practical planning evidence rather than a replacement for a fully specified G-theory design. Counts held constant in a D-study do not turn a random facet into a fixed-facet universe. Variance estimation uncertainty is not propagated to G/Phi. Component values and G/Phi are stored at full precision; rounding is only for display. Recreate older G/D results from the existing MFRM fit before reuse. Boundary or singular `lme4` fits are retained as diagnostic evidence but are not treated as decision-ready G/D-study evidence. New results record component separation explicitly. Earlier saved G-studies without this record can still supply conditional D-study calculations, with `ComponentIdentificationChecked = FALSE`; recreate the G-study from the original ratings to obtain the check. This needs no MFRM refit.

Omission does not correct missing-data bias or identify why ratings are absent. Entirely absent assignments are not reconstructed. Review the rating design and missingness assumptions before interpreting G/D results. Earlier versions silently omitted incomplete rows and unparseable scores. To reproduce complete-row selection, clean invalid labels explicitly and choose `missing = "omit"`. Older saved results remain usable when their calculation version is current, but unavailable row counts are not guessed; rerun the G-study with the original data to obtain row accounting.

References

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. doi:10.18637/jss.v067.i01. Gaussian ML/REML model and optimization. See also https://lme4.github.io/lme4/reference/isSingular.html for the meaning of a singular fitted covariance matrix.

Cronbach, L. J., Gleser, G. C., Nanda, H., & Rajaratnam, N. (1972). *The dependability of behavioral measurements: Theory of generalizability for scores and profiles*. Wiley.

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560. General G/D framework; the main-effects restriction and residual-scaling choices here are package-specific limits.

See Also

mfrm_g_study(), mfrm_d_study(), compute_facet_icc(), diagnose_mfrm()

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
if (requireNamespace("lme4", quietly = TRUE)) {
  gt <- mfrm_generalizability(fit)
  gt$variance_components
  # Look for: a Person variance component well above any single
  #   non-person facet's variance share. Large rater or criterion
  #   variance shares mean those conditions add measurement error
  #   relative to person spread.
  gt$coefficients
  # Compare G and Phi with study-specific requirements; 0.70 and 0.80
  #  are reference guides only. Phi < G means absolute decisions are noisier than relative
  #   decisions; review whether facet main effects need anchoring.
  # Always check IdentificationStatus before using the bands:
  gt$coefficients[, c("G", "Phi", "GStatus", "PhiStatus",
                      "IdentificationStatus")]
  gt$design$identification_note
  # If IdentificationStatus is not "identified", treat G/Phi as
  # design-review evidence rather than decision-ready reliability.
}
```

---

mfrm_g_study *G-study for one or several observed scores*

Description

Investigate how observed assessment scores vary across people, raters and tasks. With several score columns, such as fluency and accuracy, also study how their sources of variation relate. Use mfrm_d_study() next to compare plans with different numbers of raters or tasks.

Usage

```
mfrm_g_study(
  data,
  scores,
  person = "Person",
  rater = "Rater",
  task = "Task",
  method = c("anova", "minque0"),
  missing = c("error", "omit"),
  facets = NULL,
```

```
  nesting = NULL
)

## S3 method for class 'mfrm_multivariate_gstudy'
print(x, digits = NULL, row.names = FALSE, ...)

## S3 method for class 'mfrm_multivariate_gstudy'
summary(object, ...)
```

Arguments

data	A data frame with at most one row per observed combination of person and selected facet identifiers. Do not add rows for unassigned cells or code absent scores as zero.
scores	Names of finite numeric score columns, in the desired order. A single column is allowed as the univariate special case. Scores are neither standardized nor converted from category labels.
person, rater, task	Distinct columns identifying persons and facets. By default both rater and task facets are included. Set rater = NULL for Person-by-Task, or task = NULL for Person-by-Rater. At least one facet is required. Each included factor must have at least two observed levels. Labels may be character, factor, finite numeric, or logical. Blank or infinite labels fail; missing labels follow missing.
method	"anova" (default) requires a complete balanced design. "minque0" estimates the same covariance components from a complete or incomplete or unequal design using identity-working-covariance MINQUE. Neither method constrains covariance estimates to be positive semidefinite.
missing	"error" (default) refuses missing selected scores or factor identifiers. "omit" explicitly excludes any such row from every score's analysis and retains exclusion accounting. It does not impute values or correct missing-data bias. Infinite/nonnumeric scores are always refused.
facets	Optional character vector selecting one or two common random facets instead of rater/task, for example c(Rater = "Assessor", Occasion = "Session"). Values identify data columns; names label covariance components, D-study count columns, and plots. An unnamed vector uses the column names as labels. Labels must be unique and nonblank, contain no :, and not use reserved result names such as Person, Residual, Score, Scenario, or plotting fields. Do not supply rater or task together with facets. person and scores remain explicit. NULL preserves the task/rater interface.
nesting	Optional named character vector specifying one measurement facet nested within the other, for example c(Rater = "Task"). The name is the child facet label and the value is its parent label; with facets, use its labels rather than input column names. Persons remain crossed with these conditions. Child identities are local to each parent: R1 within Task 1 differs from R1 within Task 2. NULL (default) specifies crossed facets. See "Nested measurement facets" below.
x, object	A result from the corresponding G-study or D-study function.

digits	Display precision passed to the printed tables; NULL uses R's current setting. Stored estimates are unchanged.
row.names	Logical; print table row names? Defaults to FALSE.
...	Further table-printing arguments for print(); unused arguments to summary() are rejected.

Details

Estimate observed-score variance-covariance components for fixed score components with one or two common random measurement facets, such as raters, tasks, or occasions. Use balanced ANOVA for a complete design or explicitly select MINQUE(0) for incomplete or unequal observed designs. Conditions have common identities across persons and scores. This is not an MFRM fit.

Every retained cell must have every selected score. Duplicate cells are not supported. The same identifier (parent/child pair for a nested child) must denote the same condition across persons and scores. Equal level counts alone cannot establish that identity or random sampling from the intended universe.

In 0.2.4, mfrm_g_study() replaces the longer public name mfrm_multivariate_gstudy() for both one and several scores. The arguments, default estimator and result class are unchanged. Use mfrm_d_study() for future plans. Existing saved G-study objects remain usable without re-estimation; update constructor calls in old scripts. See mfrmr_compatibility_layer for migration from both former names.

Choose facets from the intended use of scores: common raters for judging performances, common tasks for sampling content, or common occasions for repeat assessments. All selected facets are random and, unless nesting is supplied, fully crossed; method = "minque0" permits incomplete observations of that model. Naming a column Occasion does not model growth, practice, trends, or serial correlation. Its levels must be defensibly treated as exchangeable conditions under the stated random-effects assumptions. Score columns are fixed components of the assessment, not another random facet. Three or more facets, fixed measurement facets, nesting within persons, and partial sharing across score components are not supported.

MINQUE(0) models a common mean for each score and independent, zero-mean random effects with a common covariance matrix for each component. With intercept-removal matrix H and shared-level covariance kernels K_s, it solves S_st = tr(H K_s H K_t) against Q_s = Y' H K_s H Y for every score pair. Group-count calculations avoid an observation-by-observation matrix or a full Cartesian grid. On balanced data it agrees with ANOVA. It does not optimize a likelihood, fit covariate-dependent means, or iteratively estimate working covariance weights.

The diagonally scaled moment system must have all eigenvalues above sqrt(.Machine$double.eps) times its largest eigenvalue. Otherwise the function stops because the components cannot be separated reliably by these equations. estimation retains the rank, scaled eigenvalues, condition number, and per-component replication counts. These are design and numerical diagnostics, not precision estimates or model-fit tests. Graph connectedness alone does not establish component identifiability. For example, four tasks scored by two distinct raters per performance and eight tasks scored once have the same per-person workload. Only the former provides repeated ratings within a Person/Task cell; with one rating per cell, Person-by-Task and residual variation cannot be separated in the seven-component model. Other overlaps are still needed for the remaining components. Uneven workloads across raters can also affect precision even when the moment system has full rank. vcov.mfrm_multivariate_gstudy() supplies normal-model sampling covariance and

SEs for the original component estimates, including incomplete sources. Inspect its separate negative/nonidentifiable sampling-variance diagnostics; it does not provide normal-reference intervals.

Conditioning on the observed assignments must preserve the stated random-effect means and covariances. Outcome-dependent assignment or missingness may violate this assumption; calling missingness MAR does not correct omitted covariates or selection. Review planned assignments separately from recorded scores, for example with `describe_mfrm_data()` and its `expected_design` argument. `design$observed_fraction` uses the cross-product of retained observed levels for crossed facets, or persons times the number of observed parent/child pairs for nested facets. It is not an assignment completion rate and cannot reveal entirely unobserved persons or facets.

With crossed rater and task facets, the seven components are `Person`, `Rater`, `Task`, `Person:Rater`, `Person:Task`, `Rater:Task`, and `Residual`. With one observation per cell, `Residual` combines the three-way interaction and within-cell error; they cannot be separated. With one facet, the three components are `Person`, the facet, and `Residual`; the last combines Person-by-facet interaction and within-cell error. Labels supplied in `facets` replace Rater/Task in these component names. Omitting a facet does not remove its effects from scores or support generalization to new conditions of that facet. No scores are averaged automatically. Each component is a matrix whose diagonal contains variances and whose off-diagonal contains covariances between scores. For ANOVA, mean products replace the mean squares used for a single score.

Raw component estimates, including negative variances and indefinite matrices, are retained without clipping or nearest-PSD repair. The `component_diagnostics` assess eigenvalues after scaling by the observed score SDs, using a relative numerical tolerance. This is a numerical admissibility check, not a significance test or precision assessment. Rank-deficient components are reported. `mfrm_d_study()` retains raw projected variances and evaluates each score/composite and metric separately. A non-PSD component does not automatically withhold coefficients: for example, Rater-by-Task does not enter relative error. D-study tables and plots flag non-PSD components even when a requested projection can be calculated. Inspect these diagnostics before using it. Non-PSD estimates can arise from sampling variation even under a correctly specified model, particularly when a true component is small or zero. They do not by themselves prove bad data, informative missingness, or model misspecification. Automatic clipping or deleting a component would change the estimation procedure and is not performed. Distinguish `estimation$rank`, the number of separable components, from `component_diagnostics$Rank`, the rank of each between-score covariance matrix. A true zero covariance component can have matrix rank zero while the design still separates that component from the others.

The model concerns numeric observed scores. Treating ordered categories as numeric does not estimate latent ordinal or MFRM reliability. The The FACETS G-theory web example gives different coefficients for main-effects and interaction models (about 0.72930 versus 0.55144 in its nested example). Match the specified decomposition before comparing. Its Rater and Task:Rater terms share the same nested incidence and their summed variance corresponds to Rater(Task) here. A matching coefficient does not identify those redundant terms separately. GENOVA 3.1 and urGENOVA 2.1 balanced nested examples agree with the corresponding ANOVA variance components; mGENOVA 2.1 also checks multivariate components and D-study coefficients. urGENOVA uses Henderson's Method 1 for unbalanced designs, whereas this function offers MINQUE0 there; equal results are not required for different estimators. urGENOVA has no D-study output. These comparisons do not establish unbalanced interval coverage or general interchangeability. The mGENOVA Appendix E example checks balanced numerical calculations; MINQUE(0) additionally agrees with direct covariance-kernel calculations for incomplete designs. These checks do not establish population recovery for arbitrary sparse assignments or missingness mechanisms. This function returns

point estimates, not sampling intervals.

Value

An mfrm_multivariate_gstudy list with components (three, five, or seven named covariance matrices), ANOVA mean_products and degrees_of_freedom (NULL for MINQUE), estimation (MINQUE moment matrices and diagnostics, NULL for ANOVA), data_usage (input/used/excluded counts, excluded input row positions and missing cells), component_diagnostics, score_scale (observed SDs used only for matrix diagnostics), design (facets maps facet labels to input columns and count_columns maps those labels to D-study columns; also levels, counts, method, and score convention, nesting, child_counts, completeness, balance, potential cells and observed cell fraction), and data (the retained selected input columns). print() displays the design and diagnostics and returns the same object invisibly. summary() returns the component-diagnostics data frame; use $components for the estimated covariance matrices.

Component uncertainty versus measurement error

For a balanced ANOVA result, confint.mfrm_multivariate_gstudy() adds sampling SEs and pointwise intervals for variance components of one score or a prespecified weighted composite, under an explicit normal-effects assumption. plot(g, intervals = TRUE, assumption = "normal") adds these intervals to a component plot; its returned data retain SEs, raw endpoints and reasons for unavailable intervals. For balanced or identifiable incomplete sources, explicit method = "reml_profile" supplies approximate variance or two-score covariance intervals from a separate normal REML model. Original estimates remain unchanged, with the REML estimate in IntervalCenter. It does not supply a component sampling SE, and numerical availability is not a coverage guarantee. For between-score covariance, pass exactly two score names to confint() and explicitly select method = "gci" or "regression_region" for balanced ANOVA, or experimental "reml_profile" for the joint normal model. These retain signed bounds and distinguish approximate generalized coverage from conservative normal-model enclosures. plot(g, type = "covariance", score = c("V", "W"), intervals = TRUE, method = "gci", assumption = "normal") shows their intervals across sources. Explicit method = "residual_contrast" also accepts retained ANOVA/MINQUE ratings, but supplies only Residual intervals: exact normal-model chi-square for variance, or a conservative covariance enclosure. It removes all nonresidual incidence columns, requires residual degrees of freedom, retains original points and reports a separate IntervalCenter. It does not supply the original component estimator's SE or intervals for G/Phi or other sources. See confint.mfrm_multivariate_gstudy(). score_scale contains observed score SDs used for numerical diagnostics; these are not component SEs. Neither matrix rank nor a nonnegative estimate establishes that a component is precisely known. A D-study's SEM instead describes projected measurement error in score units. The separate mfrm_multivariate_d_compare() provides approximate intervals for **plan differences** in its supported two-facet crossed scope; these are not component intervals or marginal intervals for each plan's coefficient. See mfrmr_methods_reference for the target map.

Fixed tasks as score components

To plan ratings of the same fixed interview, presentation and discussion, put the three task scores in separate columns and use task = NULL. Each row is one Person/Rater pair; the same rater and person identities must apply to every task column. The tasks and prespecified score weights define

the fixed composite. Only raters are sampled measurement conditions. Different task-specific rater teams do not satisfy this representation.

The Person covariance includes stable Person-by-fixed-task differences. For weight vector `w` and `n_r` planned raters, universe variance is `w' P w`, relative error is `w' E w / n_r`, and absolute error is `w' (R + E) w / n_r`. These match a univariate Person-by-Rater analysis of the directly weighted task score. Do not average task-specific reliability coefficients or additionally divide error by the task count. Each planned rater scores every fixed task. Two raters for three fixed tasks require six ratings per person, not two or three.

MINQUE(0) permits identifiable incomplete Person/Rater source designs; every retained row still needs all task scores. Explicit `missing = "omit"` removes a whole incomplete score vector. Do not impute unassigned tasks to manufacture common score identities. The D-study projects a future complete common-rater design; it does not estimate sparse-roster reliability. Adding or replacing tasks, partial rater sharing and an arbitrary mixture of fixed/random facets are not implemented by this representation. Holding the task count constant in a random-task model is a different assumption.

Nested measurement facets

Suppose each task has its own rater team, and each team rates the same persons on both Content and Organization. Use `nesting = c(Rater = "Task")` for Person crossed with Rater-within-Task. The five components are `Person`, `Task`, `Rater(Task)`, `Person:Task`, and `Residual`. The last combines Person-by-Rater-within-Task interaction and within-cell error. There is no separately estimated common Rater or Rater-by-Task component. A rater who actually works across tasks is not an independent nested rater; changing their identifier cannot establish independence.

ANOVA requires all persons at every observed parent/child combination and the same number of children per parent. MINQUE(0) also permits missing cells and unequal child counts when its moment equations separate the five components. Having only one child per parent confounds parent and child components. Neither method corrects informative assignments.

`design$counts["Rater"]` is the total number of distinct task/rater pairs; `design$child_counts` gives the rater count within each task. `design$levels` retains the original labels, interpreted locally for the child facet. A design is `complete` when every person has every observed nested condition, and `balanced` when it also has equal child counts. In a future D-study, `Raters` instead means raters per task: two raters for each of six tasks use twelve distinct raters and twelve ratings per person. Future designs preserve this nesting and have equal child counts. Other two-facet names and the opposite nesting direction follow the same rules; nesting within persons and score-specific child identities do not.

Reviewing incomplete designs

Begin with the planned roster and `data_usage` to distinguish unassigned cells from missing assigned ratings. Then inspect `estimation` for component separation and replication, `component_diagnostics` for covariance admissibility, and the D-study metric-specific status columns (`GStatus`, `PhiStatus`, `RelativeSEMStatus`, `AbsoluteSEMStatus`) before reading the corresponding result. None of these checks estimates sampling precision or identifies the missingness mechanism. No fixed percentage of observed cells, condition-number cutoff, or passed matrix check establishes adequate precision. D-studies project future complete balanced designs. The separate `mfrm_multivariate_d_compare()` offers approximate normal-theory intervals for prespecified differences in two-facet crossed designs; nested-design intervals are not currently available.

For the same rating workload, changing rater overlap or concentrating assignments can change estimation precision. Omitting ratings selected by their scores can introduce bias even when every requested coefficient is calculable. Neither a returned coefficient nor agreement with another program determines whether these assumptions suit the user's assessment. Observed pool sizes concern estimation in the G-study; they differ from the per-person counts in a future D-study scenario. Review which tasks and raters are shared and whether recorded ratings represent the intended population and conditions before interpreting a projected improvement.

References

Brennan, R. L. (2001). *Generalizability theory*. Springer. Chapters 9–11. doi:10.1007/97814757-34560.

Brennan, R. L. (2001). *Manual for mGENOVA, Version 2.1*. Iowa Testing Programs Occasional Papers, No. 50. Pages 7–8 and 19–22; Table 12 (page 32) and Appendix E (pages 74–77). Manual available from the author's GENOVA Suite.

Brennan, R. L. (1992). Generalizability theory. *Educational Measurement: Issues and Practice*, 11(4), 27–34. Equations 13–16 and Table 3.

Rao, C. R. (1971). Estimation of variance and covariance components–MINQUE theory. *Journal of Multivariate Analysis*, 1, 257–275. doi:10.1016/0047259X(71)900017.

See Also

`mfrm_d_study()`, `mfrm_generalizability()`. For a guided data-to-plan-to-saved-result workflow, use `vignette("mfrmr-gtheory", package = "mfrmr")`.

Examples

```
# Common tasks, without a rater facet: Brennan's published synthetic data.
# mGENOVA manual Table 12: 10 persons, 6 common items, two scores V and W.
# The item facet is named Task in the supplied long-format data.
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g_task <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
g_task$components
g_task$component_diagnostics
# To compare future designs, see ?mfrm_d_study.

# Optional planning, incomplete designs, renamed facets and nesting examples.
# example("mfrm_g_study", run.dontrun = TRUE)
## Not run:
d_task <- mfrm_d_study(g_task,
  design_grid = data.frame(Tasks = c(6, 12)), weights = c(V = -1, W = 1))
d_task$coefficients
# At six tasks, W - V has G = 0.30000 and Phi = 0.24116 (Appendix E).

# An illustrative incomplete assignment roster, not a missingness model.
sparse <- tasks[(tasks$Person + tasks$Task) %% 3 != 0, ]
sparse$V[1] <- NA_real_ # One assigned score is additionally unrecorded.
g_sparse <- mfrm_g_study(sparse, c("V", "W"), rater = NULL,
  method = "minque0", missing = "omit")
g_sparse$data_usage
```

```
g_sparse$estimation$component_support
g_sparse$component_diagnostics
# Explicit future COMPLETE designs, not reliability of the sparse roster.
d_sparse <- mfrm_d_study(g_sparse,
  data.frame(Tasks = c(6, 12)), weights = c(V = -1, W = 1))
d_sparse$coefficients # Read metric status and ComponentPSD separately.

# Existing continuous-score example; rename columns to show role mapping.
data("mfrmr_example_gtheory_crossed", package = "mfrmr")
ratings <- mfrmr_example_gtheory_crossed
names(ratings)[names(ratings) == "Rater"] <- "Assessor"
names(ratings)[names(ratings) == "Task"] <- "Session"
head(ratings) # Two score columns for each Person-Assessor-Session cell.
# Here sessions represent exchangeable assessments, without a time trend.
g <- mfrm_g_study(ratings, c("Content", "Organization"),
  facets = c(Rater = "Assessor", Occasion = "Session"))
g$components$Person
g$component_diagnostics
d <- mfrm_d_study(g,
  design_grid = data.frame(Rater = c(2, 4), Occasion = c(3, 3)),
  weights = c(Content = 0.6, Organization = 0.4))
d$coefficients
# Difference-score dependability, when subtraction is meaningful on these scales.
difference <- mfrm_d_study(g,
  weights = c(Content = 1, Organization = -1))
difference$coefficients

# Rater-only planning for one occasion; no generalization across occasions.
one_session <- ratings[ratings$Session == 1, ]
g_rater <- mfrm_g_study(one_session, c("Content", "Organization"),
  rater = "Assessor", task = NULL)
d_rater <- mfrm_d_study(g_rater, data.frame(Raters = c(1, 2, 4)))
plot(d_rater, score = "Content")

# Different rater teams for different tasks; the same Persons and scores.
data("mfrmr_example_gtheory_nested", package = "mfrmr")
nested <- mfrmr_example_gtheory_nested
head(nested) # R1 in Task 1 is a different rater from R1 in Task 2.
g_nested <- mfrm_g_study(nested, c("Content", "Organization"),
  nesting = c(Rater = "Task"))
g_nested$design$child_counts # Three raters per task; eighteen in total.
g_nested$component_diagnostics
d_nested <- mfrm_d_study(g_nested,
  expand.grid(Raters = c(2, 3), Tasks = c(4, 6)),
  weights = c(Content = 0.6, Organization = 0.4))
summary(d_nested)
plot(d_nested, x_var = "Raters") # Raters on the axis means raters PER TASK.
plot(d_nested, x_var = "Tasks", type = "sem")

## End(Not run)
```

`mfrm_misfit_thresholds` *MnSq misfit threshold pair used across mfrmr screening helpers*

Description

Returns the lower / upper bounds that mfrmr screens treat as the heuristic mean-square (Infit / Outfit MnSq) review band when flagging element-level misfit. Defaults use the published 0.5-1.5 interval as a screening convention; both ends can be overridden via R options. The interval is not a universal acceptance rule or an automatic exclusion rule. Low mean squares describe low residual variability rather than poor rater quality. `fit_measures_table()` uses this mean-square screen by default; ZSTD-only evidence is separate unless its combined rule is requested. Compare declared bands against known simulation truth with `mfrm_screening_sensitivity()`.

Usage

```
mfrm_misfit_thresholds(lower = NULL, upper = NULL)
```

Arguments

`lower`	Optional lower bound. When `NULL` (default), the active option / package default is used.
`upper`	Optional upper bound.

Details

Helpers that consume the band include `summary.mfrm_diagnostics()` (`misfit_flagged` block and `key_warnings` auto-flag), `build_misfit_casebook()` (the new `element_fit` source family), the bias / misfit narrative inside `build_apa_outputs()`, and `facet_quality_dashboard()` when `misfit_warn = NULL`. Setting the options once at the top of an analysis script therefore changes every downstream screen at once.

Value

A named numeric vector `c(lower = ..., upper = ...)` with `lower < upper`.

Configuration

Two scalar R options drive the band:

`mfrmr.misfit_lower` Lower review bound. Default `0.5`.

`mfrmr.misfit_upper` Upper review bound. Default `1.5`.

Pass scalar arguments to override the options for a single call, e.g. `mfrm_misfit_thresholds(lower = 0.7, upper = 1.3)` for the tighter Bond & Fox (2015) reporting band.

See Also

summary.mfrm_diagnostics(), build_misfit_casebook(), facet_quality_dashboard()

Examples

```
mfrm_misfit_thresholds()
old <- options(mfrmr.misfit_lower = 0.7, mfrmr.misfit_upper = 1.3)
mfrm_misfit_thresholds()
options(old)
```

---

mfrm_multivariate_d_compare

Compare prespecified multivariate D-study plans

Description

Estimate how much G, Phi or SEM changes from a reference plan, including the dependence between plans estimated from the same G-study. This first interval method requires two common random facets and an explicit normal random-effects assumption.

Usage

```
mfrm_multivariate_d_compare(
  x,
  reference = 1L,
  score = NULL,
  composite = NULL,
  assumption,
  level = 0.95
)

## S3 method for class 'mfrm_multivariate_d_comparison'
summary(object, ...)

## S3 method for class 'mfrm_multivariate_d_comparison'
print(x, digits = NULL, row.names = FALSE, ...)

## S3 method for class 'mfrm_multivariate_d_comparison'
plot(
  x,
  type = c("coefficients", "sem"),
  draw = TRUE,
  preset = "standard",
  view = c("plans", "differences"),
  ...
)
```

Arguments

x	For mfrm_multivariate_d_compare(): A result from mfrm_d_study() with at least two plans. It must retain negative_components = "raw"; this inference does not describe GENOVA-adjusted component estimators. For population plan-difference intervals alongside GENOVA-adjusted points, use the supported reference methods in confint.mfrm_multivariate_d_study(). For plot(): A result from mfrm_multivariate_d_compare().
reference	Row number of the reference plan in x$design_grid.
score	One original score name. With no selection, use the sole composite, or the first original score if there is no composite.
composite	One named composite, or "Composite" for vector weights. Select either score or composite. Several composites require selection.
assumption	Required: "normal" asserts independent, normally distributed random-effect vectors with constant component covariances. This assumption is not tested by the function.
level	Pointwise confidence level between zero and one, default .95.
object	A result from mfrm_multivariate_d_compare().
...	For summary.mfrm_multivariate_d_comparison(): Further table-printing arguments for print(); unused arguments to summary() are rejected. For plot(): Reserved; additional arguments are rejected.
digits	Display precision passed to the printed tables; NULL uses R's current setting. Stored estimates are unchanged.
row.names	Logical; print table row names? Defaults to FALSE.
type	"coefficients" for G/Phi or "sem" for SEM.
draw	Draw the figure; FALSE returns its data only.
preset	Plot style: "standard", "publication", "compact", or "monochrome".
view	"plans" (default) shows point projections on their original scale, with a diamond for the reference. "differences" shows paired changes and their approximate intervals.

Details

Use this comparison when choosing between plans specified before inspecting their estimates. For example, compare two raters and six tasks with three raters and four tasks. A positive G/Phi difference favors the comparison plan; a negative SEM difference favors it. An interval containing zero means the direction is uncertain, not that the plans are equivalent. Equal rating counts do not establish equal examinee burden or cost.

The current scope is two common random facets, complete balanced ANOVA or identifiable incomplete crossed MINQUE(0), and future complete crossed plans. Persons and both facets are sampled from their stated populations. The source assignments are held fixed and must preserve the random-effect distributions. One-facet, nested, fixed-facet, nonnormal-robust and informative-missingness intervals are not provided. Ordinal score labels alone do not justify normal effects. Incomplete source designs can have much weaker information than complete designs with the

same numbers of observed levels. For G/Phi differences from a complete balanced one-facet ANOVA source, use confint.mfrm_multivariate_d_study() with parm = "G" or "Phi" and reference. Those methods map an F or generalized confidence set, respectively; Phi retains approximate coverage. Neither uses this delta approximation. The same generic supports one-facet "RelativeSEM" and "AbsoluteSEM" differences by mapping a common error-SD interval. Relative SEM retains exact positive-variance normal-model coverage; absolute SEM uses approximate MLS. For general balanced crossed/nested two-facet plans, that generic also accepts explicit method = "mean_square_region" with reference for conservative G/Phi/SEM difference enclosures from shared source uncertainty. These are not paired-delta intervals or tight joint difference images; they can be wide. The delta method in this function keeps its crossed scope.

The method uses raw estimated components in the Gaussian covariance of quadratic-form estimates, then the delta method for paired differences and a normal critical value. It uses the gradient of the difference, not a sum of independent marginal variances. For a complete source design this agrees with mean-square variances 2 * MS^2 / df. These are approximate intervals, not exact finite-sample guarantees. Small facet pools, uneven assignments and estimates near boundaries can impair the approximation. In a bounded assessment, a skewed-effect condition reduced nominal 95% coverage to about 92%; this method must not be described as distribution robust.

Negative components are retained and flagged. No components, differences or interval endpoints are clipped. An unavailable point projection, boundary derivative or nonpositive estimated difference variance leaves that interval unavailable with a reason. Identical plans have an exact zero difference when their point projections are available. Component and sampling-covariance diagnostics do not establish model fit or interval accuracy.

Intervals are pointwise for each prespecified comparison. They do not support choosing the largest observed improvement, simultaneous claims over all rows, or comparisons between adaptively selected score weights. SEM is measurement error in score units; SE here is sampling uncertainty in the *difference*.

Point projections are recomputed from the stored G-study using current rules; no G-study is refitted and x is not modified. The G-study must retain its analyzed data. For incomplete sources, covariance computation processes blocks of rows to avoid a full observation-by-observation matrix; its running time is still quadratic in the number of observed ratings.

plot() method: Plot prespecified D-study plans and their differences

Show G/Phi or SEM for every plan, including the reference. Use view = "differences" for changes from the reference with approximate pointwise intervals. In that view, the vertical zero line represents no change; an interval crossing it does not establish equivalence. Positive G/Phi differences or negative SEM differences favor the comparison plan. The method requires normal random effects.

For plot(): The plan view connects supplied plans in their original row order. Both facet counts appear on the horizontal axis; lines are visual guides, not an interpolated response to changing one count. For conventional D-study curves that vary one count while holding the other constant, use plot.mfrm_multivariate_d_study(). The plan view has no sampling intervals: adding the reference estimate to a difference interval would not produce a confidence interval for the individual plan.

Plans and score weights must have been specified before inspecting the results. In the difference view, intervals are not simultaneous over plans or metrics. A point without an interval is retained as an open circle; an asterisk marks rows with unavailable intervals. Missing values are never replaced by zero. Each difference panel has its own horizontal scale. Original score units apply to

SEM differences. plot_data() identifies the score/composite, its weights and the reference counts. Explanatory notes and a suggested title are returned rather than drawn. All plotted plans are future complete crossed plans, even when the source design is incomplete.

Value

An mfrm_multivariate_d_comparison list with comparisons (one row per comparison plan and metric), covariance (joint sampling covariance of those differences), design_grid, reference, selected score/composite and weights, level, method, and component/sampling-covariance diagnostics. source_design and source_rows retain the source method, observed level counts, completeness, score convention and row-use totals, without raw ratings or identifiers. print() displays these separately from the future plans and includes composite weights. Save the full comparison with saveRDS(); a CSV of summary() alone does not retain this context or the confidence level and approximation assumptions. Earlier saved comparisons without source context remain readable and say it is absent. print() returns the same comparison invisibly without recomputation. summary() returns the comparison table. Status describes interval availability; point differences can remain available without an interval.

For plot(): Invisibly, an mfrm_plot_data object. plot_data() extracts the exact comparison table, point-projection series (including the reference), unavailable rows for the selected view, interval_unavailable rows, design_grid, reference, weights, title and labels. Base graphics are supported; automatic ggplot conversion is not provided for this plot.

Calculation and source of the approximation

For one selected score or fixed composite, write its observed values as the vector z. Its seven raw component estimates can be written as $\hat{\theta}_s = z' A_s z$, where each symmetric A_s is fixed by the observed design and removes the common mean: $A_s 1 = 0$. For MINQUE(0), let K_s indicate shared levels for component s, $H = I - 11'/N$, $B_s = HK_sH$, and $S_{st} = \mathrm{tr}(B_sB_t)$. Then $A_s = \sum_t (S^{-1})_{st} B_t$. These are the moment equations described in mfrm_g_study() and the MINQUE framework of Rao (1971).

Under the stated Gaussian model with covariance $V = \sum_s \theta_s K_s$, the quadratic-form identity is

$$C_{st} = \mathrm{Cov}(\hat{\theta}_s, \hat{\theta}_t) = 2\mathrm{tr}(A_s V A_t V).$$

This follows from the product moment in Magnus (1978, p. 209), subtracting the product of the expectations. The mean term vanishes because each A_s removes the intercept. The identity concerns the **true** covariance; this function substitutes raw estimated components into V.

For a metric f and plans a and b, define $h(\theta) = f_b(\theta) - f_a(\theta)$ and $g = \nabla f_b - \nabla f_a$, evaluated at the component estimates. The first-order delta approximation (Oehlert, 1992) gives

$$\widehat{\mathrm{SE}}(\hat{h}) = \sqrt{g' \hat{C} g}.$$

The interval is $\hat{h} \pm z_{1-\alpha/2} \widehat{\mathrm{SE}}(\hat{h})$, where $\alpha = 1 - \mathrm{level}$. Stacking the difference gradients in J gives the returned joint covariance $J\hat{C}J'$.

Thus there are distinct steps: a Gaussian component-covariance identity, substitution of estimates, linearization of the nonlinear metric, and a normal reference interval. Only the first is an exact model identity. The current balanced implementation uses the plug-in 2 * MS^2 / df, not the alternative df + 2 moment correction. The references support the G/D targets and these mathematical steps; none

validates this entire package procedure for every supported design. This is not an implementation of all GENOVA-family interval methods.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

References

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560. G/D targets and sampling variability.

Rao, C. R. (1971). Estimation of variance and covariance components–MINQUE theory. *Journal of Multivariate Analysis, 1*, 257–275. doi:10.1016/0047259X(71)900017. Component moment estimation.

Magnus, J. R. (1978). The moments of products of quadratic forms in normal variables. *Statistica Neerlandica, 32*(4), 201–210. doi:10.1111/j.14679574.1978.tb01399.x. Gaussian product moments; see p. 209. Author's copy.

Oehlert, G. W. (1992). A note on the delta method. *The American Statistician, 46*(1), 27–29. doi:10.1080/00031305.1992.10475842. Taylor/moment approximation framework.

See Also

`mfrm_d_study()`, `plot.mfrm_multivariate_d_comparison()`

`mfrm_multivariate_d_compare()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_multivariate_d_compare", run.dontrun = TRUE)
```

```
## Not run:
# Existing fictional ratings: one row per Person-Rater-Task, two scores.
data("mfrmr_example_gtheory_crossed", package = "mfrmr")
ratings <- mfrmr_example_gtheory_crossed
head(ratings)
g <- mfrm_g_study(ratings, c("Content", "Organization"))
d <- mfrm_d_study(g,
  data.frame(Raters = c(2, 3, 4), Tasks = c(6, 4, 3)),
  weights = c(Content = .5, Organization = .5))
comparison <- mfrm_multivariate_d_compare(d, reference = 1,
  assumption = "normal")
summary(comparison)
plot(comparison) # G/Phi for each plan, including the reference.
plot(comparison, type = "sem")
plot(comparison, view = "differences") # Paired difference intervals.

## End(Not run)
```

`mfrm_network_analysis` *Analyze the MFRM design network*

Description

Analyze the MFRM design network

Usage

```
mfrm_network_analysis(
  fit = NULL,
  diagnostics = NULL,
  top_n_subsets = NULL,
  min_observations = 0,
  include_graph = FALSE,
  data = NULL,
  person = NULL,
  facets = NULL,
  score = NULL,
  basis = c("observed", "planned"),
  weight = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`top_n_subsets`	Optional maximum number of connected-subset rows to retain before constructing the graph.

`min_observations`
: Minimum observations required to keep a subset row.

`include_graph`
: Logical; if `TRUE`, include the underlying `igraph` object in the returned bundle. Defaults to `FALSE` so outputs remain easy to serialize.

`data`
: Optional raw assignment/rating data, supplied instead of `fit`.

`person, facets`
: Names of the person column and one or more facet columns in `data`. Include only Rater in `facets` to inspect person-rater links without a common Criterion node obscuring separate panels.

`score`
: Numeric score column for `basis = "observed"`. Recode study-specific missing markers to `NA` first; no category recoding or estimation is done.

`basis`
: `"observed"` includes rows with complete identifiers and nonmissing finite scores; `"planned"` includes complete assignment rows without scores. Planned networks require `score = NULL` and `weight = NULL`.

`weight`
: Optional numeric weight column for observed raw data. Missing and zero-weight rows do not create edges. Edge weights still count rows, not sampling weights. Negative or infinite weights are refused.

Details

`mfrm_network_analysis()` treats the person/facet-level observation design as an undirected weighted graph. Nodes are person or facet levels; edges connect levels that co-occur in at least one observed rating; edge weights are co-observation counts. The resulting network metrics are design diagnostics, not psychometric measures of person ability or rater quality. This is the package's assignment/co-observation network: unlike the score-relation networks returned by `rater_network_analysis()` and `rater_halo_network_analysis()`, it exposes disconnected co-observation components. Graph connectedness alone does not establish model identification. `SourceSubsets`, `RetainedSubsets` and `ScopeComplete` record whether subset filters retained the full observed design. Connectedness of selected subsets cannot be generalized to omitted subsets. Recreate older design reviews from the existing fit and matching diagnostics to record this coverage. `plot(net, type = "centrality")`, `plot(net, type = "facet_summary")`, and `plot(net, type = "network")` provide immediate visual checks; use `draw = FALSE` to extract reusable plot data.

The most useful review columns are:

- `Components`: more than one component means separate co-observation regions, including known levels without observed links.
- `IsArticulationPoint`: a node whose removal would increase disconnectedness.
- `IsBridge`: an edge whose removal would increase disconnectedness.
- `Betweenness`: a routing-dependence indicator; high values identify levels that carry many shortest paths through the design graph.

In incomplete rater-mediated designs, these graph summaries help identify fragile linking structures before interpreting facet measures or planning additional data collection.

Value

A bundle of class `mfrm_network_analysis` containing:

- summary: graph-level connectedness and vulnerability metrics
- node_metrics: node-level degree, strength, graph-theoretic centrality, and cutpoint flags
- edge_metrics: edge-level weights, betweenness, and bridge flags
- facet_summary: facet-level aggregation of node/bridge indicators
- cut_nodes: articulation-point rows from node_metrics
- bridge_edges: bridge rows from edge_metrics

References

- McEwen, M. R. (2015). *Development of a Software Prototype for Generating and Classifying Incomplete Many-Facet-Rasch Model Rating Designs*. Brigham Young University.
- Csardi, G., Nepusz, T., Traag, V., Horvat, S., Zanini, F., Noom, D., & Muller, K. (2026). *igraph: Network Analysis and Visualization*.

Before fitting

Supply data, person, facets and score instead of fit. Known levels from incomplete/unscored rows remain as nodes, including isolates. A planned assignment roster can be examined with basis = "planned"; absent assignments cannot be reconstructed from an observed-only file. row_usage records each input row's inclusion and first exclusion reason. With a fit, only its retained rows are available here; review original missing assignments before fitting. No prefit result supplies an identification audit or model-readiness decision. settings$is_disjoint describes the source graph, even when subset filters omit components; compare source/retained subset counts. Filters do not establish linking. See vignette("mfrmr-networks-and-reporting") for a panel example.

See Also

subset_connectivity_report(), diagnose_mfrm(), mfrmr_linking_and_dff, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_network_analysis", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_operational")
if (requireNamespace("igraph", quietly = TRUE)) {
  net <- mfrm_network_analysis(data = ratings, person = "Person",
    facets = "Rater", score = "Score")
  net$summary
  plot(net, type = "network", draw = FALSE)
}

## End(Not run)
```

mfrm_pca *Principal components of numeric external features*

Description

Express numeric attributes as principal components: new variables ordered by how much variation they describe. For example, summarize rater experience, training and workload together. Choose `components` when you want to retain fewer variables; the default retains all nonzero components. PCA does not estimate rater quality, ability or groups. Identifiers and omitted rows are retained.

Usage

```
mfrm_pca(
  x,
  components = NULL,
  scale = TRUE,
  weights = NULL,
  missing = c("error", "omit")
)

## S3 method for class 'mfrm_pca'
print(x, ...)

## S3 method for class 'mfrm_pca'
summary(object, ...)

## S3 method for class 'mfrm_pca'
plot(
  x,
  type = c("scree", "scores", "loadings"),
  components = NULL,
  groups = NULL,
  labels = NULL,
  draw = TRUE,
  preset = "standard",
  ...
)
```

Arguments

x	For `mfrm_pca()`: A table reviewed with `mfrm_features()`. For `mfrm_cluster_kmeans()`, a result from `mfrm_pca()` is also accepted. For `plot()`: A result from `mfrm_pca()`.
components	For `mfrm_pca()`: Number of leading principal components to retain. `NULL` retains the numerical rank, using `sqrt(.Machine$double.eps)` relative to the largest singular value. A requested count cannot exceed that rank.

	For `plot()`: For scores, two distinct retained component numbers. For loadings, one retained component number. Defaults to `c(1, 2)` for scores and 1 for loadings. Unavailable axes cause an error.
`scale`	Divide centered numeric features by their sample standard deviations (`TRUE`, default). With `FALSE`, distances retain the original feature units. Numeric coding must have a meaningful Euclidean interpretation; factors, ordered factors, characters and logical features are refused.
`weights`	Optional named, strictly positive finite numeric vector with one weight per selected feature. Defaults to equal weights. Names, not vector order, identify features. Remove a feature to exclude it.
`missing`	Either `"error"` (default) or `"omit"`. The latter excludes incomplete entities explicitly, retaining their IDs, reasons, and missing group membership in the result. No values are imputed.
`...`	For `print()`: Reserved for method compatibility. For `plot()`: Reserved for future use; additional arguments are rejected.
`object`	An object returned by `mfrm_pca()`.
`type`	`"scree"` (default), `"scores"`, or `"loadings"`.
`groups`	Optional `mfrm_cluster_kmeans()`, `mfrm_cluster_pam()` or `mfrm_cluster_hierarchical()` result used to color a scores plot. Its IDs, included entities and shared PCA features must match. Labels only annotate the view; they do not refit PCA or imply separation on every component. Groups use both colour and point shape, including monochrome output. Shapes repeat after six groups; inspect the ID-aligned table for crowded views.
`labels`	Whether to display entity IDs on silhouettes or heatmaps. The default displays them for at most 50 entities. No entities are sampled when labels are hidden. Profile plots always label groups and levels.
`draw`	Draw the plot when `TRUE`; `FALSE` only returns plotted values.
`preset`	Plot style: `"standard"`, `"publication"`, `"compact"`, or `"monochrome"`.

Details

Uses `stats::prcomp()` on centered, optionally standardized features, multiplied by `sqrt(weights / max(weights))`. Thus weights apply to squared Euclidean distances. Weight names identify original features. Constant features, nonfinite transformations and numerically unrepresentable variance are refused. Missingness stops by default; explicit omission retains every excluded ID and its original reason.

PCA finds directions of feature variance, not clusters or latent MFRM abilities. All nonzero components preserve distances in the transformed feature space, within numerical precision. Retaining fewer components changes the clustering objective and can remove informative directions. High explained variance does not establish valid groups. Signs of loadings and scores are arbitrary; tied eigenvalues can also rotate their subspace. Loadings are eigenvector coefficients in the transformed feature space, not correlations in the original units.

Use the same fitted PCA to review scores, loadings and groups. Passing its result to `mfrm_cluster_kmeans()` uses the retained scores without further standardization or whitening. PCA is not a required step for k-means. Group comparison and imputation sensitivity retain their descriptive scope.

plot() method: Plot numeric feature PCA and optional exploratory groups

Review explained variation, original-feature coefficients or entity scores from a saved PCA without refitting it.

For plot(): Scores and loading signs are arbitrary. Loadings are eigenvector coefficients in the centered, weighted and optionally standardized space; they are not original-unit correlations. A two-component scores view omits other directions, so apparent overlap or separation is only a projection. No confidence region, group validity or rater-quality judgment is implied.

Value

An mfrm_pca object with ID-aligned scores (omitted rows are NA), loadings, a variance table (variance, proportion, cumulative proportion and retained status), transformation (centers, divisors and weight factors), original feature_data and settings. summary() returns variance.

For plot(): Invisibly, an mfrm_plot_data object with the exact plotted table, selected components, group colour/shape encoding, excluded IDs and axis meanings. Scree data retain the full variance table, including components not used for clustering. plot_data() extracts these values. as_ggplot() converts all three views using their saved axes and encodings; component = "table" retains the complete view. Excluded IDs and transformation metadata remain attached. Use ggplot2::labs(title = NULL, subtitle = NULL) to hide headings in the returned ggplot. Labels follow the saved labels setting; no entities are sampled when labels are hidden. Physical text and point sizes can differ between base graphics and ggplot. Converted scores use equal axis units and equal displayed spans to avoid a narrow panel when the selected components explain very different amounts of variation.

Session plot defaults

Set options(mfrmr.plot_preset = "publication") to choose a session default for plotting functions that expose the common preset argument. The supported values are "standard", "publication", "compact" and "monochrome". Precedence is an explicit call argument, then the session option, then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of plot() for report bundles use the same option through .... Plots without a common preset argument, including extended-model plots with their own palette controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with as_ggplot() uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply preset explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with old <- options(mfrmr.plot_preset = "monochrome") followed by options(old). Use options(mfrmr.plot_preset = NULL) to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

mfrm_cluster_kmeans(), mfrm_cluster_compare(), plot.mfrm_pca(), stats::prcomp()

mfrm_pca(), mfrm_cluster_kmeans()

Examples

```
raters <- data.frame(Rater = paste0("R", 1:8),
  ExperienceYears = c(1, 2, 3, 4, 10, 12, 13, 15),
  WorkshopHours = c(4, 12, 8, 20, 12, 28, 16, 32),
  AnnualRatings = c(80, 120, 90, 160, 280, 350, 300, 400))
features <- mfrm_features(raters, "Rater", names(raters)[-1])
pca <- mfrm_pca(features, components = 2)
summary(pca)
pca$loadings
groups <- mfrm_cluster_kmeans(pca, k = 2, seed = 17, silhouette = FALSE)
plot(pca, type = "scores", groups = groups)
plot(groups, type = "profile", feature = "ExperienceYears")
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_pca", run.dontrun = TRUE)
## Not run:
attributes <- data.frame(ID = letters[1:6],
  ExperienceYears = c(1, 2, 4, 8, 10, 12),
  WorkshopHours = c(8, 16, 12, 24, 16, 32))
result <- mfrm_pca(mfrm_features(attributes, "ID", names(attributes)[-1]))
plot(result)
plot(result, type = "scores")
plot(result, type = "loadings", components = 1)
plot_data(plot(result, type = "scores", draw = FALSE))$table

## End(Not run)
```

---

mfrm_population_intervals

Pointwise intervals for a conditional-normal population model

---

Description

Report coefficients, prespecified linear contrasts and residual variance from an estimated RSM/PCM MML population model. Coefficient intervals use a normal approximation; residual variance intervals exponentiate a normal interval for log residual variance. These experimental intervals have unqualified repeated-sample coverage. Method qualification is incomplete even when numerical checks pass. Saved cautions are also retained as an `interpretation_notes` table in mfrm_results(), reports and table exports.

Usage

```
mfrm_population_intervals(
  object,
```

```
  level = 0.95,
  contrasts = NULL,
  anchor_covariance = NULL,
  independent_anchors = FALSE
)

## S3 method for class 'mfrm_population_intervals'
plot(
  x,
  type = c("coefficients", "contrasts"),
  draw = TRUE,
  preset = "standard",
  reference = 0,
  ...
)

## S3 method for class 'mfrm_population_intervals'
print(x, ...)

## S3 method for class 'mfrm_population_intervals'
summary(object, ...)

## S3 method for class 'mfrm_population_intervals'
vcov(object, ...)
```

Arguments

`object`	A native RSM/PCM MML fit with an active population model, or saved native-location `mfrm_facet_intervals()` output from that fit. The latter reuses its numerically checked full joint covariance without differentiation. For `summary()` and `vcov()`, supply the population-interval result.
`level`	Pointwise confidence level, strictly between zero and one.
`contrasts`	Optional numeric matrix of prespecified linear combinations of population coefficients. Give each row a distinct nonempty target name and name every column with a fitted coefficient name, including zero-weight columns, as in `names(fit$population$coefficients)`. Columns are matched by name. Each row must be finite and nonzero; dependent rows are allowed for pointwise intervals. `NULL` retains the coefficient/variance-only output.
`anchor_covariance`	Optional full sampling covariance of the supplied anchor estimates, including their correlations, in native ability units squared. Row and column names must match every directly anchored level on the one noncentered facet. Requires the fixed-grid direct RSM/PCM anchor-origin route with an intercept-only population and fixed SD. `NULL` retains the existing conditional-on-anchors calculation.
`independent_anchors`	Logical, default `FALSE`. Set `TRUE` with `anchor_covariance` only when source calibration and receiving responses are independent. Shared respondents or de-

	pendent clusters require cross-sample covariance and are unsupported. This is an explicit assumption, not a check inferred from different Person identifiers.
x	For `print.mfrm_population_intervals()`: An interval result. For `plot()`: A saved result from `mfrm_population_intervals()`.
type	Show `"coefficients"` (default, including the intercept) or the prespecified `"contrasts"` stored in `x`. Residual variance has different units and remains in `summary(x)` rather than sharing this axis.
draw	Logical; `FALSE` returns the plot data without drawing.
preset	A package plot preset, including `"publication"` and `"monochrome"`. The session default is used when omitted.
reference	Optional finite vertical reference, default zero; `NULL` omits it. This is not a practical-importance or automatic decision threshold.
...	For `print.mfrm_population_intervals()`: Unused. For `plot()`: Unused; unsupported arguments are rejected.

Details

For the bounded fixed-SD anchor-origin route, `anchor_covariance` with `independent_anchors = TRUE` propagates an independently estimated anchor covariance into the population coefficients and contrasts. Alternatively, pass a saved facet-interval result already containing source propagation and omit both arguments; its sensitivity and conditional information are reused once. See `mfrm_facet_intervals()` for the formula, required independence, supported scope and complete bundled-data example. Point estimates are unchanged; conditional SE/bounds remain in separate `Conditional*` columns. `vcov()` then returns source-propagated coefficient covariance. The fixed population variance still has no SE or interval. Drift, source-estimate bias, dependent samples and uncertainty in the supplied SD are not corrected.

The fitted model is $\theta_i \mid X_i \sim N(X_i\beta, \sigma_e^2)$. Residual variance is conditional on the fitted predictors; it is not the marginal variance of ability. Predictor transformations and coefficient coding are those retained by the fit. The covariance is taken from the inverse of the full information, including all structural nuisance terms. A contrast matrix L gives estimates $L\widehat{\beta}$ and covariance $LV_\beta L^\mathsf{T}$, using the coefficient block of that inverse. Contrast rows are added to the interval table. These describe conditional population means or their linear combinations, not differences between two individual abilities or causal effects. For example, in `~ X`, weights `(0, 2)` compare the means at `X = 1` and `X = -1`. Interactions, factor reference levels and transformations must use the fitted coding. The helper accepts weights in coefficient coordinates; it does not infer raw covariate profiles or rebuild a transformation from a new sample.

Use direct MML with fixed or adaptive integration, unit observation weights, full-rank population coding, centered additive facets and centered steps, without constraint groups or fitted interactions. The fixed-SD anchor-origin extension described below also admits supplied anchors. Source consistency, category support, local solution, full joint information and finer-integration checks must pass as in `mfrm_facet_intervals()`. The procedure can return eligible intervals while the source fit retains `InferenceReady = FALSE`; inspect the interval's `CIEligible`, `Status`, checks and cautions. This does not change overall fit readiness or qualify other inference procedures. A recorded identification or boundary failure prevents intervals. Failed numerical checks retain point estimates and reasons with unavailable uncertainty. This is an interior, model-based approximation: it

does not handle a zero-variance boundary, misspecified populations, dependent Persons or informative assignment. Passing numerical checks does not prove global optimality or coverage. When residual variance is estimated, a zero-residual-variance candidate at the retained nuisance estimates is also compared with the fitted likelihood at both integration orders. A higher boundary likelihood beyond the numerical allowance prevents intervals. This inexpensive comparison does not optimize the boundary or rule out other better boundary candidates. Requests using saved facet inference also perform this likelihood comparison, without refitting or differentiating again. A fit with explicit `population_sd` conditions on that positive fixed SD; zero variance is not a feasible candidate. Its residual-variance row retains the supplied value with `Status = "fixed"`, missing SE/bounds and `CIEligible = FALSE`. Coefficients and their contrasts still have model-based intervals when numerical checks pass. With `population_formula = ~1`, direct anchors on the selected `noncenter_facet` can identify the population mean and free facet locations. By default, these intervals condition on the supplied anchors and SD. The optional `anchor_covariance` extension adds independent source-anchor uncertainty while still conditioning on the SD.

`SE` is on the reported estimate scale (delta-method SE for variance). `WorkingEstimate` and `WorkingSE` use coefficient/log-variance coordinates. `WorkingLower` and `WorkingUpper` retain the interval on that scale. Exponentiation can give displayed variance bounds of zero or infinity outside floating-point range; `NaturalScaleStatus` identifies that representation limit while the finite log-scale interval is retained. `vcov()` returns their covariance, with explicit coordinate names; its final coordinate is log residual variance, not natural variance, when variance is estimated. A fixed variance has no free covariance coordinate. It remains the coefficient/log-variance covariance when contrasts are supplied; `contrast_covariance` separately contains the covariance of the requested rows. No multiplicity adjustment or automatic significance decision is made.

Save the result directly, or attach it through `mfrm_results(fit, intervals = list(population = ci), compute = "never")`. Tables, reports and exports reuse saved intervals and numerical checks. Supplied contrast weights and their covariance are also exported. These tables prefix coordinate columns with `beta:/contrast:` to distinguish them from the `Target` identifier column. The saved result retains the unchanged fit, including its population formula, trained terms, factor levels and coding; save the raw reference profiles separately if you used them to construct the weights. The ordinary population summary remains a point-estimate table.

`plot()` method: Plot saved latent-population coefficients or contrasts

Read point estimates and their experimental pointwise intervals from an existing population analysis, without fitting or recalculating uncertainty.

For `plot()`: Rows retain their saved order. Filled circles and lines show numerically available intervals. Crosses with a `CI unavailable` label retain finite points without eligible finite bounds. A missing point keeps its labeled row without a marker; it is never plotted at zero. Titles and explanatory notes are returned, not drawn. No significance colors or automatic classifications are added. Plotting does not requalify a saved numerical result.

Coefficients describe the conditional population mean on the fitted native ability scale. A continuous coefficient is per unit of its fitted predictor; the intercept depends on predictor zero values and factor coding. Read interactions, transformations and categorical comparisons using the saved coding, not an assumed reference group. Coefficients on different predictor scales are not directly comparable measures of importance. Higher ability predicts higher coded ratings in these RSM/PCM models; whether that is better performance depends on the rubric. Coefficients are not standardized effects.

Contrasts use the stored linear weights and may describe a conditional mean or a difference between means. They are not individual ability differences, causal effects, simultaneous intervals

or prediction intervals. Numerical availability does not establish nominal coverage: the source's experimental status, failed checks and qualifications remain in `notes` and the tables. A wide interval remains visible. Missing bounds are not infinite endpoints. When source-anchor uncertainty was requested, the saved intervals include that contribution. The returned subtitle, settings and `anchor_uncertainty` distinguish it from intervals conditional on exact supplied anchors; conditional bounds remain in the returned table.

For an executable bundled-data example, run the example on `mfrm_population_intervals()`, which creates `ci` and both views. Then use `p <- plot(ci, type = "contrasts", draw = FALSE)`, `plot_data(p, "contrast_weights")` and `plot_data(p, "notes")` to read it. `as_ggplot(ci, type = "contrasts")` converts the same saved result.

Value

An `mfrm_population_intervals` object with the interval table, working-coordinate covariance, full parameter covariance, numerical checks, source identity and unchanged source fit. With `contrasts`, also contains the column-aligned `contrasts` matrix and `contrast_covariance`; targets in the table have type `coefficient_contrast` and prefix `contrast:`.

For `plot()`: Invisibly, an `mfrm_plot_data` object. `table` preserves selected source rows and adds `PlotStatus`, `Label` and `Y`; `intervals` retains the complete original table. `population_spec` preserves the formula, trained terms, factor levels and coding; `population_coding` is a readable coding table. `contrast_weights`, source identity, settings, numerical checks and interpretation notes accompany the figure. Use `plot_data()` to inspect these and `as_ggplot()` for an editable figure with the same saved values.

Reading the result

Start with `summary(ci)`, then inspect `ci$checks`, `ci$numerical_checks` and `ci$cautions`. Keep these with the interval table when saving or sharing. plot(ci) displays saved coefficients; `plot(ci, type = "contrasts")` displays the supplied linear contrasts. Unavailable intervals remain visible, and `as_ggplot()` reuses the same data.

- `Status = "available"` and `CIEligible = TRUE` mean that the numerical checks permit the experimental interval. They do not certify nominal coverage. `InferenceReview` records this distinction in each row.
- `Status = "fixed"` retains a supplied residual variance without an inferential interval; missing uncertainty is not an estimated zero SE.
- `Status = "numerical_review_failed"` retains the point estimate and failed-check reason, with unavailable SEs and bounds. `nonpositive_variance` or `nonfinite_bounds` instead identifies an unusable target-specific uncertainty calculation. Missing bounds are not infinite endpoints.
- `NaturalScaleStatus = "outside_floating_point_range"` means the finite log-variance interval cannot be fully represented after exponentiation. Inspect `WorkingLower` and `WorkingUpper`; a displayed zero is not an interval qualified for a zero-variance boundary.

Coverage evidence is specific to the target, design and complete estimation procedure. In the retained RSM checks, mean-at-$X = 1$ and log-variance intervals at 20 Persons remain inconclusive under the prespecified practical criteria. The independent 240-Person slope/mean-difference confirmation supports those targets in its tested design; it does not establish a minimum sample size or

qualify arbitrary regressions, PCM designs or all contrasts. All outputs from this helper therefore retain their experimental status.

These intervals do not add calibration uncertainty to individual scoring. predict_mfrm_units() has separate source and scoring checks; its conditional posterior intervals hold the fitted calibration and population parameters fixed. Availability here does not authorize an otherwise refused scoring result. See mfrmr_interval_guide() for the different uncertainty targets.

Session plot defaults

Set options(mfrmr.plot_preset = "publication") to choose a session default for plotting functions that expose the common preset argument. The supported values are "standard", "publication", "compact" and "monochrome". Precedence is an explicit call argument, then the session option, then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of plot() for report bundles use the same option through .... Plots without a common preset argument, including extended-model plots with their own palette controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with as_ggplot() uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply preset explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with old <- options(mfrmr.plot_preset = "monochrome") followed by options(old). Use options(mfrmr.plot_preset = NULL) to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

mfrm_facet_intervals(), predict_mfrm_units(), mfrm_results(), plot.mfrm_population_intervals(), apa_table(), export_mfrm_results()

mfrm_population_intervals(), mfrmr_interval_guide(), plot.mfrm_facet_intervals()

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrm_population_intervals", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
persons <- data.frame(Person = unique(ratings$Person))
# Fictional predictor for this example; use your observed covariates.
persons$X <- seq(-1, 1, length.out = nrow(persons))
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  model = "RSM", method = "MML", population_formula = ~ X,
  person_data = persons, quad_points = 61)
L <- matrix(c(0, 2), nrow = 1,
  dimnames = list("X = 1 minus X = -1", c("(Intercept)", "X")))
```

```
ci <- mfrm_population_intervals(fit, contrasts = L)
subset(summary(ci), Type == "coefficient_contrast")
plot(ci)
p <- plot(ci, type = "contrasts")
plot_data(p, "contrast_weights")
plot_data(p, "notes")
results <- mfrm_results(fit, intervals = list(population = ci),
  include = "fit", compute = "never")
apa_table(results$population_intervals$population)

## End(Not run)
```

`mfrm_random_rater_intervals` *Bootstrap prediction intervals for observed random raters*

Description

Regenerate persons, shared raters and scores on the observed assignment, refit the same RSM, and retain prediction errors for every planned replicate.

Usage

```
mfrm_random_rater_intervals(object, nsim = 499L, seed, level = 0.95)

## S3 method for class 'mfrm_random_rater_intervals'
plot(
  x,
  level = x$settings$level,
  method = c("studentized", "error"),
  comparison = TRUE,
  draw = TRUE,
  sort = c("input", "estimate", "uncertainty"),
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  reference = 0,
  text_scale = 1,
  point_size = 2.5,
  ...
)

## S3 method for class 'mfrm_random_rater_intervals'
confint(
  object,
```

```
  parm = NULL,
  level = 0.95,
  method = c("studentized", "error"),
  ...
)

## S3 method for class 'mfrm_random_rater_intervals'
summary(object, ...)

## S3 method for class 'mfrm_random_rater_intervals'
print(x, ...)
```

Arguments

object	A numerically ready fit_mfrm_random_rater() result with positive information, positive population SD and finite positive rater prediction SEs. Estimated zero-variance sources are not supported.
nsim	Number of planned bootstrap datasets; default 499, minimum 2. Each requires a full refit. Small values are useful for checking a workflow, not for stable tail quantiles.
seed	Required nonnegative integer simulation seed. The caller's random number state is restored. Per-replicate seeds and RNG kind are retained.
level	For mfrm_random_rater_intervals(): Pointwise nominal coverage; default 0.95. For plot(): Nominal pointwise coverage; defaults to the saved level.
x	For mfrm_random_rater_intervals(): A bootstrap result. For plot(): A result from mfrm_random_rater_intervals().
method	For mfrm_random_rater_intervals(): "studentized" (default) uses generated-minus-estimated rater effects divided by each refit's calibration-adjusted PredictionSE. "error" uses unscaled prediction errors as a comparison. Changing method or level through confint() reuses saved draws without new fitting. For plot(): "studentized" or "error"; see mfrm_random_rater_intervals().
comparison	Show ordinary normal intervals from the same source fit.
draw	FALSE returns plotted data without opening a device.
sort	Row order for interval plots: input order, increasing estimate, or increasing interval width. Missing values come last; ties retain input order. Sorting is descriptive, not a test of differences.
palette	"accessible" uses blue and dark orange with filled/open symbols; "mono" uses dark grey with the same symbol distinctions.
title, caption	NULL uses the default text; a string replaces it; "" removes it. Interpretation notes remain in the saved plot data.
show_title, show_notes	Show the title and caption, respectively. FALSE hides text without deleting its metadata.

show_labels	Show IDs on interval and precision plots. Distribution plots summarize estimates and do not label individual IDs. For crowded precision plots use FALSE and consult the retained table.
reference	Vertical reference line in logits; NULL omits it. This is an orientation aid, not a quality threshold.
text_scale	Positive multiplier for text size.
point_size	Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).
...	Unused.
parm	Rater IDs for confint(); NULL returns all fitted raters.

Details

Every generated dataset draws one ability per person and one severity per rater, shared across their observed rows. Generating fixed facets, steps and both population SDs are the source estimates. Each Person ability is drawn from the fitted normal population. The refit re-estimates calibration and each population SD unless originally fixed. Earlier saved fits retain their known N(0,1) ability population. Per-trial records include the refitted ability SD and its zero-boundary status. The target is a realized effect relative to the population mean, not a centered effect, new-rater score or fixed rater coefficient. Coverage is marginal over new persons, rater effects and scores at this assignment; it is not conditional coverage for each fixed true severity.

Studentized endpoints are the source estimate plus its PredictionSE times the empirical tail quantiles (type 1) of bootstrap errors divided by refitted PredictionSE. Unscaled endpoints add error quantiles directly. All planned replicates remain in the denominator. Failed fits and unavailable studentizers are unresolved, including studentizers at estimated variance boundaries. Lower-tail calculations place unresolved roots at minus infinity; upper-tail calculations place them at plus infinity. The resulting limits enclose empirical bootstrap limits for any completion of those roots. They may be unbounded; failures are never silently removed or replaced. A numerically ready boundary refit still supplies an unscaled prediction error.

This is a model-based bootstrap candidate, not a general finite-sample coverage guarantee. Linear mixed-model bootstrap theory motivates the construction but does not establish its accuracy for this crossed ordinal RSM, few raters, variance boundaries or Laplace approximation. Basic error intervals and studentized intervals need separate studies of their coverage. The fit's normal-population and assignment assumptions remain essential. Omitted scores stay omitted; this conditions on analyzed rows and does not simulate a missingness mechanism or impute assigned scores. Rater contrasts, familywise intervals and simultaneous rater classification are not provided.

plot() method: Compare bootstrap and ordinary random-rater intervals

For plot(): Arrows identify unbounded bootstrap endpoints. Their displayed ends are plot limits, not finite interval limits. Ordinary intervals never replace an unbounded bootstrap interval. These are pointwise prediction intervals for realized rater effects, not a rater-quality classification.

Value

An mfrm_random_rater_intervals object containing the source rater table, intervals, aligned error/studentized matrices, generated effects, refitted estimates/SEs, per-trial checks, warnings/errors,

seeds, model settings and analysis data. `confint()` returns a two-column matrix with availability, method and Monte Carlo resolution attributes. No native pointers are stored; save with `saveRDS()`.

For `plot()`: Invisibly, an `mfrm_plot_data` object with exact bounds, availability, settings and the source estimates. Use `plot_data()` for custom graphics.

Review unresolved refits

Inspect `$trials` before interpreting the intervals. `FitReady` combines numerical and information checks; it does not certify interval coverage. New results also retain `OptimizerCode`, `NumericalReady`, `InformationPositive`, `PersonQuadratureStable`, `QuadraturePoints`, `CheckPoints`, `LogLikDifference`, `GradientDifference`, `EstimatedVarianceBoundary` and `PersonVarianceUpperBoundary` from each refit. Together with the recorded gradient and Person-variance boundary, these distinguish unresolved numerical calculations from variance boundaries and missing regular studentizers. A failed refit has missing additional check values and its recorded error; missing checks are not passes. Older saved trial tables lack these additional fields and cannot acquire them without rerunning the corresponding refits. Failed checks must not be removed, selectively retried until successful or relabeled as adequate interval coverage. Increasing Person quadrature can address integration precision but does not qualify the rater Laplace approximation, the interval method or the population assumptions.

Increasing `nsim` alone does not resolve unavailable prediction errors. For example, with 499 planned refits at 95%, 13 unresolved studentized errors for one rater make both of that rater's limits infinite under the type-1 completion rule. Twelve unresolved errors leave finite empirical limits when all other errors and the source estimate/SE are finite. This describes the calculation, not a threshold establishing accurate coverage. If the unresolved fraction remains above the nominal tail probability, a larger run still has unbounded limits. Inspect causes in `$trials` before committing to more refits; do not discard unresolved draws.

References

Chatterjee, S., Lahiri, P. and Li, H. (2008). Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models. *Annals of Statistics*, 36, 1221–1245. doi:10.1214/07AOS512.

See Also

`fit_mfrm_random_rater()`, `plot.mfrm_random_rater_intervals()`

`plot.mfrm_testlet_scores()` for display controls and accessible output. Bootstrap comparisons retain the interval view.

Examples

```
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$random_rater$fit
intervals <- example$random_rater$intervals
# To regenerate (requires RTMB >= 2.0 and repeated model fitting):
# intervals <- mfrm_random_rater_intervals(fit, nsim = 19, seed = 923701)
# These 19 saved trials illustrate mechanics, not accurate 2.5% tails.
# Every trial is retained; unresolved trials can give unbounded intervals.
summary(intervals)
```

```
confint(intervals, method = "error")
# Changing the level reuses the same draws, with no further fitting:
confint(intervals, level = .90)
```

`mfrm_report` *Build reports from measurement, G/D-study or residual results*

Description

`mfrm_report()` connects existing measurement results, G/D-study results and residual exploration to tables, Markdown and HTML. It does not refit, recompute diagnostics or create new inferential support. Pass G/D-study and `mfrm_residual_network()` objects directly. Rater/halo networks and `fit_rater_network()` results also connect directly, retaining definitions, unavailable pairs, network fit, cluster-bootstrap contrasts and ERGM checks. Attach previously calculated G/D intervals explicitly. See the dedicated section below and the editable `templates/gtheory-report.Rmd`.

For an `mfrm_results()` object, it turns the comprehensive first-screen result into a first-screen table, section plan, claim-readiness table, report-gap table, report-index table, template-index table, fit-criteria table, result-specific fit evidence summaries, fit-reporting wording templates, precision/separation reporting templates, bias/DFF reporting templates, misfit/pathway reporting templates, linking/anchor reporting templates, ZSTD-convention table, evidence-boundary table, next-action table, and optional Markdown or HTML report. For `mfrm_results()` inputs, except for the individual rater sheet, the object and its table list retain the exact source-fit `fit_readiness*` tables from `mfrm_results()`; report synthesis does not reinterpret or upgrade them. The `decision` table translates that same record into interpretation, formal-inference, reason, and next-action text.

Usage

```
mfrm_report(
  x,
  style = c("qc", "apa", "validation", "reviewer", "technical", "rater"),
  output = c("object", "markdown", "html", "tables"),
  facet = NULL,
  rater = NULL,
  audience = c("rater", "researcher"),
  label = NULL,
  max_cases = 5,
  interval = NULL,
  intervals = NULL,
  study_context = NULL
)
```

Arguments

x	An `mfrm_results()` object, a supported G/D-study or network result, or output from `estimate_bias()`, `estimate_all_bias()`, `review_mfrm_anchors()`, `build_linking_review()` or `build_model_choice_review()`.

`style` Report emphasis. `"qc"` is the default first-screen report. `"apa"` emphasizes manuscript wording, `"validation"` emphasizes the validity-argument boundary, `"reviewer"` emphasizes reviewer response preparation, and `"technical"` emphasizes appendix/reproducibility routes. `"rater"` creates a standalone individual feedback sheet from saved native additive RSM/PCM or two-family GPCM MML results; it requires `facet` and `rater`.

`output` Return format: `"object"` for an `mfrm_report` object (`mfrm_rater_feedback` when `style = "rater"`), `"markdown"` for a character scalar, `"html"` for a temporary HTML file, or `"tables"` for the report's named data-frame list.

`facet` For `style = "rater"`, the fitted non-Person facet representing raters, for example `"Rater"`. It is not inferred from the column name.

`rater` For `style = "rater"`, one character level of `facet`.

`audience` For a rater sheet, `"rater"` gives plain-language guidance; `"researcher"` adds the model, estimator and diagnostic basis. Both use the same saved numerical values and omit source identifiers.

`label` Optional recipient-facing label for a rater sheet. The default is `"Selected rater"`; even the selected source identifier is not copied. Any identifying information explicitly supplied here will be displayed.

`max_cases` Maximum number of saved unexpected ratings in a rater sheet, ordered by absolute standardized residual. Default 5; use 0 to omit individual cases. This is not a misfit threshold.

`interval` Optional name of a saved fixed-facet interval attachment in `x$facet_intervals` for an additive-model rater sheet, or a saved component slope interval in `x$gpcm_inference` for two-family GPCM. The attachment must contain the selected individual coefficient, not just a difference involving it. A single matching attachment is used automatically. Multiple matching attachments require an explicit choice. No interval is calculated here.

`intervals` For a G/D-study only, saved `confint(x)` output or a list of such results. Source, design and target context are checked. `NULL` reports point estimates without intervals; logical values do not trigger computation. Old component intervals lacking source context must be recreated with the current `confint(g)`. This differs from singular `interval`, which selects an attachment for an individual rater sheet.

`study_context` Optional named character vector of study declarations, for example `c(question = "...", rating_unit = "...", score_direction = "Higher means easier to understand", rubric = "1 = very difficult; 9 = very easy", transformation = "None", anchor_source = "...")`. Names must be unique; values must be nonmissing and nonblank. The same declaration can accompany separate measurement, G/D and feedback reports when their meanings match. It is retained in `$study_context`, `$tables$study_context`, Markdown, HTML and report appendix export. These are user declarations, not model-verified facts: they never recode data, reverse parameter signs, validate anchors or change eligibility. For a recipient-facing rater sheet, supply only context intended for that recipient; text is copied verbatim.

Details

The intended workflow is:

1. Create `res <- mfrm_results(fit, include = ...)`.
2. Inspect `summary(res)$triage` and `summary(res)$next_actions`.
3. Create `report <- mfrm_report(res, style = "qc")`.
4. Read `summary(report)` and `report$first_screen` before opening detailed report tables.
5. Use `report$report_index` to choose the next `PrimaryTable`, `TemplateTable`, plot route, or export route.
6. Use `report$template_index` before copying APA/QC/validation wording.
7. Use `style = "apa"`, `"validation"`, `"reviewer"`, or `"technical"` only when that reporting question is needed.

Report rows deliberately distinguish evidence from claims. The testlet and random-rater route is a smaller stored-result report: all supported styles (excluding `"rater"`) retain numerical checks, data usage, interval meanings and supplied predictions/intervals. When saved scores include a scoring roster, the `scoring_roster` table preserves its rows, identifiers and omitted scores; testlet `scoring_blocks` separately summarizes observed blocks. Older results without a roster do not reconstruct one from calibration rows. It does not supply ordinary residual diagnostics or fit/APA wording templates; `template_index` is empty. See the model-specific section in `mfrm_results()` for supported tables and plots. The `Response residuals` row distinguishes an unsupplied saved `mfrm_response_diagnostics()` result (`not_computed`) from a supplied result with no available observed-row results (`unavailable`) or results requiring review. If needed, compute these diagnostics separately for a numerically ready fit and attach them through `mfrm_results(fit, response_diagnostics = diagnostics)`. Available summaries remain descriptive (`caveat`): ordinary-model cutoffs do not apply, and they do not establish model adequacy. Report creation never computes the missing diagnostics. Older result bundles can be passed to `mfrm_report()` again to update the report; previously exported files retain their original text.

For ordinary models, the `first_screen` table is the compact entry point: it gives an overall row and one row per major evidence area with status, readiness, main issue, next action, and primary route. The `summary.mfrm_report` method summarizes that first screen into immediate actions, optional not-requested sections, claim-readiness counts, report gaps, and template-boundary rows without introducing a new pass/fail decision. The default print method follows the same short reading order and does not print every detailed evidence table. HTML output places the same reader guidance and report-summary tables before the full Markdown text so the browser view starts from the first-screen route. The `report_index` table is the detailed evidence-route index: it lists the major report areas, evidence status, readiness label, review-signal count, and the primary/template tables, evidence routes, template routes, plot routes, export route, and `mfrm_results(include = ...)` preset to inspect next. In ordinary use, open detailed tables through the `PrimaryTable` and `TemplateTable` columns rather than scanning every element of `report$tables`. The `template_index` table then stacks all fit, precision, bias, misfit, and linking wording templates into a single boundary/claim-strength index before users drill into the area-specific template tables. The `claim_readiness` table marks which report claims are ready, caveated, unavailable, or require additional requested sections. The `report_gaps` table turns those statuses into follow-up actions. The fit-specific tables keep multiple MnSq threshold profiles, observed fit-status counts, and engine-vs-FACETS-style ZSTD conventions visible, including the small-df/capping boundary used for FACETS-style ZSTD review.

They summarize the stored fit_measures component from mfrm_results(); mfrm_report() itself does not recompute diagnostics. The fit_reporting_templates table turns those counts into cautious reporting language while keeping MnSq, ZSTD standardization, df sensitivity, and separation/reliability in separate sentences. All reporting-template tables share EvidenceTable, EvidenceRoute, BoundaryType, ClaimStrength, and RecommendedUse columns so each template can be traced back to its evidence and claim boundary. The default RecommendedUse is "report_with_context"; more restrictive rows request evidence, identify a methods or appendix caveat, or require targeted follow-up. template_index stacks those columns across all template areas so report authors can review unsupported or caveated wording before opening the full template text. The precision_reporting_templates table does the same for separation, reliability, and strata using the stored precision review and diagnostics$reliability. precision_by_facet preserves every stored facet row, original values, missing-value reasons and inference flags. Templates report model and fit-adjusted indices separately for each facet; they are not confidence bounds. Cross-facet ranges in the compatibility summary are not an overall reliability coefficient. A Person separation index computed with EAP/posterior SD is not posterior-variance EAP reliability; this report does not compute that different index. fit_calculation_basis identifies the recorded target of the MnSq tables, including EAP plug-in and imported source definitions. Missing metadata stays unrecorded. Rebuild reports from saved results to obtain these explanations; no re-estimation is needed. The bias_reporting_templates table is available when the source result was built with include = "bias" and keeps facet-level screens, interaction-bias contrasts, DFF follow-up, and fairness conclusions in separate sections. The misfit_reporting_templates table is available when the source result was built with include = "misfit_review" and keeps unexpected responses, displacement, pathway-map evidence, and case-review actions separate. The linking_reporting_templates table is available when the source result was built with include = "linking" and keeps anchor readiness, drift review, equating-chain review, and GPCM support boundaries separate. For example, fit and separation are not collapsed into a single pass/fail statement; bias screens are not treated as final fairness conclusions; pathway/misfit rows are case-review prompts; and drift/equating claims require multiple fitted forms or waves.

Value

Depending on output, an mfrm_report object, a Markdown character scalar, an mfrm_report_html object, or a named list of data frames. For style = "rater", object output has class mfrm_rater_feedback and contains title, label, audience, review, guidance, notes, tables and markdown. It does not contain the original result object.

Anchor, bias and model-choice modules

Use mfrm_report(review_mfrm_anchors(...)) before anchored fitting, mfrm_report(build_linking_review(...)) for saved link evidence, mfrm_report(estimate_bias(...)) for a prespecified interaction, and mfrm_report(estimate_all_bias(...)) for a collection of exploratory screens. Collection reports keep every requested interaction in tables$overview, all retained cells in tables$bias_cells with an Interaction key, failed pairs in tables$errors, and calculation settings/history and facet directions. Failed pairs remain visible even with keep_errors = FALSE; unavailable screens are not reported as negative results. No collection-wide multiplicity adjustment is added. Use mfrm_report(build_model_choice_review(...)) for candidate-model roles and comparability. Reporting does not fit a new model, select new interaction cells, calculate a missing LRT or test anchor invariance. Full bias cells and anchor values remain alongside compact summaries; source objects and unavailable states are preserved. style = "rater" continues to require indi-

vidual feedback from mfrm_results(). Supply study_context to keep rubric direction, transformations, anchor provenance and the question with the report. A shared declaration does not establish equivalence between a raw G-study score and a latent MFRM measure. See the editable templates/assessment-review.Rmd for separate modules using bundled data and a saved report archive.

Planning and numerical review

Pass recommend_mfrm_design() results directly to preserve the planning target, threshold sources, all candidate designs and long criterion outcomes. Facet calibration reliability is not inter-rater agreement. Recommendations apply to the simulated conditions; no universal sample size is inferred. Main-effects mfrm_d_study() results also enter directly. These reports preserve residual-scaling assumptions and unavailable coefficients separately from full interaction-component G-studies. Newly created projections retain source components and observed counts; missing historical metadata are labelled not_recorded without refitting. Intervals are unsupported for this approximation and for simulation recommendations. Native measurement reports include numerical_review: separate evidence and next actions for optimization, curvature, starts, integration and replay. This table neither reruns diagnostics nor changes inferential readiness.

G/D-study and residual-exploration reports

Saved Stan/brms summaries read by read_mfrm_posterior() also enter this route directly. Reports retain their credible-interval level, fixed values, priors, model-data identity and sampling review. They do not calculate new intervals, native MML diagnostics, prior sensitivity or posterior predictions.

Pass mfrm_g_study(), mfrm_d_study(), mfrm_multivariate_d_compare() or mfrm_residual_network() output directly, without mfrm_results(). The non-rater styles all preserve the same complete domain-specific tables and interpretation notes. Report creation never refits, clusters or computes intervals. The overall review status asks for substantive study context; it is not a numerical convergence or formal-inference assessment.

G/D tables retain observed counts separately from future plans, fixed score/composite weights, signed covariance components, component policy, G/Phi, universe/error variances, measurement SEM and unavailable states. Saved component or plan/contrast intervals retain their methods, assumptions and any separate interval center. Without supplied intervals the report explicitly states not_requested. Paired-delta comparisons have their own report, preserving their shared-source covariance and approximation. These are observed-score studies, not latent-trait fits or MFRM separation reliability. Brennan (2003) explains why coefficients should be accompanied by error variances; Jiang et al. (2022) emphasize uncertainty and studied design conditions. Their findings do not validate every interval method supplied here.

Use build_summary_table_bundle(report) for all numerical and contextual tables, apa_table(report, which = "coefficients") for a selected D-study table, and export_summary_appendix() for CSV/HTML. summary(report) keeps the full evidence tables in evidence_tables; top_n does not truncate them. Save the complete report/source with saveRDS() for replay. Source archives can contain participant identifiers and require sharing review. The installed editable template is located by system.file("templates", "gtheory-report.Rmd", package = "mfrmr").

Residual reports retain all nodes and pairs, overlap, threshold settings and cluster availability. They support exploratory review; they do not establish dimensions, rater quality, stable classes or causal

relationships. See `mfrm_residual_network()` for unsupervised learning, validation and the distinct role of supervised random forests.

Stored agreement in ordinary-model reports

When diagnostics are present, the narrative and `tables$interrater_interpretation` distinguish observed agreement from the fitted-model expected baseline and from separation reliability. `tables$interrater_summary` retains the stored numerical summary unchanged. The denominator counts matched-context opportunities across rater pairs, not independent people. Unavailable values, zero matched opportunities and an absent stored summary are described explicitly; no missing agreement statistic is computed during reporting. Reports without diagnostics retain their existing component-status explanation rather than inventing values. These summaries share their interpretation with `build_apa_outputs()`. See `interrater_agreement_table()` for calculations and matching conventions. Markdown and HTML contain the same explanation; export with `export_mfrm_results(res, include = "report")` to retain the summary and interpretation as CSV files as well. Existing saved APA drafts retain their original wording until regenerated with `build_apa_outputs()` and the same saved fit/diagnostics. Regeneration does not refit the model.

Two-family GPCM reports

Reports retain all saved non-Person fitted locations, category steps and component slopes. Locations identify their centering constraint; steps identify their owner and level and are centered offsets, not standalone thresholds. Separately computed experimental location/contrast intervals from `mfrm_facet_intervals()` may be attached; step intervals remain unavailable. The Markdown view shows up to 20 rows per table with a notice; complete tables remain in `$tables` and CSV exports. Report styles do not change inferential support.

Corrected JML reports

Results from an explicit `jml_correction_order` have an estimator-specific report of saved estimates, numerical status and RootSE interpretation. The report does not contain ordinary fit tests or structural confidence intervals. The `"rater"` style is unavailable for this estimator. Saved `mfrm_response_diagnostics()` output adds conditional response means, probabilities and descriptive residual summaries, with unavailable indices and their reasons retained. These do not classify rater quality. See `mfrm_results()` and the **Corrected JML** section of `fit_mfrm()`.

Individual rater sheets

Use `mfrm_report(res, style = "rater", facet = "Rater", rater = "R01", output = "html")` to create a temporary HTML sheet. Open its `$path`, review it, and copy that file to a permanent location for distribution. HTML is self-contained, includes print styling and category-use bars with numerical tables, and does not load external resources. Page count depends on content and browser print settings; this is not a PDF export API. After reviewing the sheet, use `file.copy(sheet$path, recipient_file)` to keep it at a chosen HTML path. To prepare sheets in a later session, save the complete `res` with `base::saveRDS()` and reload it with `base::readRDS()`. The RDS file retains fitted data and identifiers for the analyst; it is not the recipient's sheet. Reloading and reporting reuse the saved analysis without updating it for new ratings. `output = "tables"` includes an `interpretation_notes` table with the overall review, audience-specific guidance and section explanations. Keep this table with the numerical tables when saving CSV files or sharing them. These

explanations are the sheet's fixed, recipient-safe text; free-form notes from the source analysis are not copied. The same table is available in the sheet object's `$tables`. HTML and Markdown show the explanations beside their sections, without duplicating them as another table. The `uncertainty` table retains `Available` and a recipient-safe `Reason`, distinguishing constraint-fixed values, insufficient independent-cluster information, unusable variance/bounds and failed numerical/profile checks where the saved diagnostics identify the cause. If the interval is unavailable, both bounds are missing in the sheet, even when the analyst's saved result retains one computed endpoint. This does not mean zero uncertainty or an unbounded interval. The original saved analysis is unchanged; unidentified reasons direct the recipient to the analyst. Printing the sheet also shows interval availability and its explanation. Recreate sheets from saved `mfrm_results` objects to obtain these fields; reading an older sheet does not update its contents.

The additive RSM/PCM sheet includes scoring tendency (severity), exposure, available saved fixed-facet uncertainty, ordinary Infit/Outfit, category use and selected unexpected ratings. Severity is oriented so that positive values mean lower expected scores. Its zero is the fitted model reference, not necessarily the average of the other raters; custom centering and anchors matter. Model scores and expected scores stay on the fitted category coding. The category table also shows the original numeric scores when a mapping exists. Exposure and category percentages count retained rows without weights; weight sums are reported separately. Neither is planned-design completion.

Supply matching `diagnose_mfrm()` output to `mfrm_results()` to include fit and unexpected ratings. Attach `mfrm_facet_intervals()` output through `intervals = list(raters = ci)` for supported MML fixed-facet intervals. JML sheets can show saved descriptive diagnostics but do not gain formal fixed-facet intervals. Inspect the complete saved interval result and its numerical cautions before sharing; free-form cautions are not copied into the recipient's sheet. Missing sections explain the missing input. Ineligible source fits retain a prominent review notice. Severity is not rater quality, and no automatic misfit cutoff, exclusion decision or diagnostic accuracy claim is added. These sheets do not support one-family GPCM, fitted interactions, imported fits, testlet or shared-random-rater models; use their specific results and reports because their effects and diagnostics differ.

For two-family GPCM MML, the same entry creates an experimental descriptive sheet. Select the modeled rater facet explicitly; either slope owner is supported. It separates the fitted location from the component slope, retains their distinct centering references, and reports category use, retained exposure and how many levels of the other facet were observed. Category-step offsets are shown only when this facet owns the steps. They are centered offsets, not standalone thresholds. This individual sheet does not display location or step intervals; use the analyst report for separately requested location intervals. A larger slope is not evidence of competence or accuracy.

Attach saved `confint.mfrm_fit()` output from `parm = "slopes"` through `intervals` in `mfrm_results()` to include a matching experimental component-slope interval. `interval` chooses among named attachments; pointwise/Bonferroni meaning, profile/log-Wald method and unavailable bounds are retained. No location interval or test of rater differences is implied. Attach saved `mfrm_response_diagnostics()` output through `response_diagnostics` for same-data posterior residual summaries and cases. The sheet counts saved, available, unresolved and not-included rows separately; a summary average is unavailable if any saved row for that recipient is unresolved. These residuals have no expectation-one reference or calibrated cutoff, exclude calibration uncertainty and do not predict independent future ratings. Creating or reopening a sheet does not fit, integrate, compute intervals or diagnose new responses. Inspect the complete analyst results before sharing a sheet; numerical convergence does not qualify feedback decisions.

All four output formats use only selected numeric summaries and fixed explanatory text. They

omit the source fit, Person identifiers, other rater identifiers, original row numbers, task labels and free-form source notes. Case numbers refer only to the displayed ordering. This prevents copying those identifier fields; it is not a guarantee against recognition from rating patterns, small groups or an explicitly supplied `label`. Review the content before sharing, and use `max_cases = 0` when cases are unnecessary. Distribute the standalone sheet, not the comprehensive results/export bundle, which retains the original analysis. Rater-specific arguments are rejected with other report styles, even when explicitly supplied as `NULL`.

References

Brennan, R. L. (2003). *Coefficients and indices in generalizability theory*. CASMA Research Report No. 1, University of Iowa. https://education.uiowa.edu/sites/education.uiowa.edu/files/2026-04/casma-research-report-1-archived.pdf.

Jiang, Z., Raymond, M., DiStefano, C., Shi, D., Liu, R., & Sun, J. (2022). A Monte Carlo study of confidence interval methods for generalizability coefficient. *Educational and Psychological Measurement, 82*, 705–718. doi:10.1177/00131644211033899.

See Also

`mfrm_results()`, `export_mfrm_results()`, `build_apa_outputs()`, `reporting_checklist()`, `mfrmr_output_guide()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrm_report", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Fit these ratings; replace the data-loading line for your own analysis
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

# Build results, then turn them into a report
res <- mfrm_results(fit)
report <- mfrm_report(res)

# Read the report and the issues to address
summary(report, view = "reader")
report$first_screen[, c("Area", "Status", "MainIssue", "NextAction")]
report$fit_calculation_basis
report$precision_by_facet[, c("Facet", "Levels", "Reliability",
```

```
                              "RealReliability", "InputUncertaintyBasis")]
contextual_report <- mfrm_report(res, study_context = c(
  rating_unit = "One Person by Rater by Criterion rating",
  score_direction = "Higher coded scores indicate more favorable performance",
  rubric = "Synthetic example: permitted scores 1 to 4", transformation = "None"
))
contextual_report$tables$study_context

# Select a recipient explicitly. Missing intervals are explained in the sheet.
recipient <- as.character(fit$prep$levels$Rater[1])
sheet <- mfrm_report(res, style = "rater", facet = "Rater",
                     rater = recipient, output = "html", max_cases = 0)
sheet$path
# Review in a browser, then choose a permanent path for continuing work.
recipient_file <- tempfile(fileext = ".html")
stopifnot(file.copy(sheet$path, recipient_file, overwrite = FALSE))

## End(Not run)
```

`mfrm_residual_network` *Explore residual associations and groups of facet levels*

Description

After reviewing a fitted measurement model, locate facet levels whose standardized residuals move together across persons. For example, explore criteria that share unexplained variation, or raters with similar residual profiles. This descriptive, signed correlation network complements `q3_statistic()`, residual PCA and case review.

Usage

```
mfrm_residual_network(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  min_pairs = 5L,
  missing = c("pairwise", "complete"),
  min_abs = 0,
  clusters = NULL,
  linkage = c("average", "complete")
)

## S3 method for class 'mfrm_residual_network'
summary(object, ...)

## S3 method for class 'mfrm_residual_network'
print(x, ...)
```

```
## S3 method for class 'mfrm_residual_network'
plot(
  x,
  type = c("network", "matrix", "dendrogram"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ...
)
```

Arguments

fit	A supported fit_mfrm() result, with ordinary observation-level standardized residuals available from diagnose_mfrm().
diagnostics	Optional matching diagnostics; computed once if omitted.
facet	A single fitted facet. Its levels, not persons, are the nodes and the objects clustered. Defaults to "Rater".
min_pairs	Minimum shared persons for a correlation, at least three.
missing	"pairwise" uses the available persons for each pair. "complete" uses only persons with finite aggregated residuals at every selected facet level. No residual is imputed. Clustering requires "complete" so all distances describe the same sample.
min_abs	Absolute correlation threshold for displayed edges, in [0, 1]; default zero retains every nonzero available association. This is a display choice, not a significance threshold. It never changes the correlations or the clustering distance.
clusters	Optional integer number of groups from 2 to the number of levels. NULL skips clustering. The number is supplied, not estimated.
linkage	"average" (default) or "complete" agglomerative linkage.
object	An mfrm_residual_network.
...	Reserved; unused arguments are rejected.
x	For print.mfrm_residual_network(): An mfrm_residual_network. For plot(): A result from mfrm_residual_network().
type	"network", "matrix" or "dendrogram".
preset	Visual preset; see mfrmr_visual_diagnostics.
draw	Whether to draw with base graphics.

Details

plot() method: Plot saved residual associations and exploratory groups

Compare the full matrix with its thresholded network, then inspect the clustering tree when requested. Network positions are a deterministic circle: distance on the page is not a statistical distance. Positive edges are solid red and negative edges dashed blue; thickness represents |r|. Gray matrix cells denote unavailable correlations. Node colors do not encode quality. Dendrogram labels append the chosen group number in square brackets; these numbers are labels, not ranks. Titles and cautions are retained in plot_data(), not printed into the figure.

Value

An mfrm_residual_network with nodes, all pairs, displayed edges, the correlation matrix, aggregated residuals, cell_counts, person_usage, settings, clustering (status, distance, tree and membership), source model metadata and interpretation notes. plot() returns saved figure data; mfrm_report() and build_summary_table_bundle() connect the result to reporting. Complete RDS archives include person identifiers: review sharing permissions.

For plot(): Invisibly, an mfrm_plot_data containing the selected view, saved matrix, nodes, pairs, edges, clustering, settings and interpretation notes. Plotting does not recompute residuals or clustering. Save the analysis object to redraw every view. The matrix supports as_ggplot().

Statistical definition

Finite StdResidual values are averaged without weighting within each Person-by-facet-level cell. Pearson correlations across persons define the network. Thus repeated ratings are not treated as independent persons. cell_counts records differing amounts of evidence behind those means. Missing overlap and zero variance produce unavailable (NA) associations, with reasons in pairs; they never become zero edges. All nodes, including isolates, remain in the output. Positive edges indicate co-movement and negative edges counter-movement after the fitted model. These are marginal residual correlations, not conditional/partial correlations or raw Yen Q3.

Unequal assignments, aggregation, estimated Person effects and finite samples can induce residual structure, including negative correlations. Inspect overlap and the rating design before explaining an edge as a substantive relationship. Threshold sensitivity is not a calibrated test. No edge p-values, uncertainty intervals or stability estimates are supplied.

Machine learning and interpretation

Hierarchical agglomerative clustering is unsupervised learning: it merges similar nodes without known outcome labels. With complete-person Pearson correlations, sqrt(2 * (1 - r)) is Euclidean distance between centered, unit-length residual profiles. Average linkage uses the mean between-group distance; complete linkage uses the maximum. Strong negative correlation means dissimilar profiles, not similarity based on absolute correlation. A supplied clusters cuts the resulting stats::hclust() tree with stats::cutree(). Changing min_abs cannot change these groups.

Clustering remains unavailable if any level lacks sufficient common-person variation; the graph and failure reason are still returned. Pairwise deletion can yield an indefinite correlation matrix, so it is not used to manufacture clustering distances. Common-person deletion can select a different population and greatly reduce the sample; inspect person_usage.

Groups are exploratory residual-profile similarities, not latent classes, demonstrated rubric dimensions, rater competence or automatic grounds for excluding a rater. This procedure is not exploratory graph analysis (EGA): Golino and Epskamp (2017) used a graphical-LASSO partial-correlation network and community detection. It is also not a jointly estimated residual-network measurement model. Compare substantive explanations, inspect cases and check sensitivity to sample, linkage and group count. For validation, repeat preprocessing, measurement fitting and clustering within independent or appropriately resampled data; resampling saved residuals alone omits calibration uncertainty. The appropriate resampling unit follows the rating design, including dependencies across persons, tasks and raters.

Where random forests belong

Wesolowski (2019) first used k-means on differential-rater-functioning indices to construct rater types, then random forests to predict those labels. That study motivates a supervised follow-up, not random forests as the clustering algorithm implemented here. Its features and targets differ from these residual correlations. In this supervised use, a forest combines predictions from bootstrap samples and random candidate feature subsets; it requires a declared prediction target and out-of-sample assessment.

A classifier predicting exploratory groups can reproduce the grouping rule without establishing external validity. Fit feature transformations, measurement models and cluster discovery within training data, keep a separate test set and define how new units receive reference labels. Split at the intended generalization unit (for example new raters, persons or tasks); row-wise splitting of repeated ratings can leak information. Out-of-bag error alone does not validate labels constructed on all data. Report class balance, confusion matrices, discrimination/calibration where appropriate and independent substantive outcomes. Feature importance is predictive, not causal. This function does not fit a random forest; use established ML packages for a separately designed prediction study.

References

Wesolowski, B. C. (2019). Predicting operational rater-type classifications using Rasch measurement theory and random forests: A music performance assessment perspective. *Journal of Educational Measurement, 56*, 610–625. doi:10.1111/jedm.12227.

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLOS ONE, 12*, e0174035. doi:10.1371/journal.pone.0174035.

See Also

mfrm_cluster_kmeans(), mfrm_cluster_hierarchical(), mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_residual_network", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
net <- mfrm_residual_network(fit, facet = "Criterion", missing = "complete",
                             clusters = 2)
net$pairs                # association, shared-person count, unavailability
net$clustering$status    # check before interpreting groups
plot(net, type = "matrix", draw = FALSE)
report <- mfrm_report(net)
build_summary_table_bundle(report)$table_index

## End(Not run)
```

`mfrm_response_diagnostics` *Describe fitted response probabilities and residuals*

Description

Compute response means, variances and descriptive Infit/Outfit summaries. Ordinary and extended RSMs and two-family GPCMs integrate uncertain latent effects; corrected GPCM JML conditions on saved calibration and reprofiled Person estimates.

Usage

```
mfrm_response_diagnostics(
  fit,
  rows = NULL,
  group_by = NULL,
 quad_points = fit$settings$quad_points %||% fit$config$estimation_control$quad_points
)

## S3 method for class 'mfrm_response_diagnostics'
plot(
  x,
  facet = NULL,
  style = c("paired", "scatter"),
  draw = TRUE,
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  text_scale = 1,
  point_size = 2.5,
  ...
)

## S3 method for class 'mfrm_response_diagnostics'
summary(object, ...)

## S3 method for class 'mfrm_response_diagnostics'
print(x, ...)
```

Arguments

`fit`	A numerically ready native RSM MML `fit_mfrm()`, `fit_mfrm_testlet()` or `fit_mfrm_random_rater()` result, a converged two-family GPCM fixed-grid

EM or adaptive direct MML fit, or a shared-owner GPCM fit with an explicit `jml_correction_order` and an available point solution.

`rows` Distinct original input row numbers to return, or `NULL` for all assigned rows. This selects outputs only: every observed source rating still informs the joint posterior or saved JML Person profile. Groups summarize the selected rows; selection does not refit the model.

`group_by` Fitted identifier columns to summarize separately. `NULL` uses all Person, testlet/rater and fixed-facet columns. No grouping by score. Corrected JML lists the fixed facets first, then Person, so the default plot shows the first declared facet.

`quad_points` Normal quadrature order, 7 to 121 for testlets or 7 to 241 for ordinary RSMs, two-family GPCMs or shared raters. Defaults to the saved fit's order. Results are checked at `2 * quad_points + 1`; each category probability must change by no more than `1e-7`. Omit this argument for corrected JML, which performs no integration.

`x` Saved `mfrm_response_diagnostics()` output.

Also for `summary()`: A saved response-diagnostics result.

`facet` One recorded grouping column; default is the first.

`style` `"paired"` connects each group's Infit and Outfit on a common axis. `"scatter"` plots Infit against Outfit; labels identify groups.

`draw` `FALSE` returns plot data without opening a device.

`palette` `"accessible"` uses blue circles and orange triangles; `"mono"` retains distinct symbols in black. No red/green warning scale.

`title, caption` `NULL` uses the default text; a string replaces it; `""` removes it. Interpretation notes remain in the saved plot data.

`show_title, show_notes` Show the title and caption, respectively. `FALSE` hides text without deleting its metadata.

`show_labels` Show group labels. Unavailable groups keep their labels in the paired view; both views retain all rows in `plot_data()`.

`text_scale` Positive multiplier for text size.

`point_size` Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

`...` For `summary.mfrm_response_diagnostics()`: Unused.

For `plot()`: Unused; ordinary-model threshold arguments are not accepted.

`object` A saved response-diagnostics result.

Details

Calibration is held fixed. For a posterior replicate rating sharing the original row's latent effects, the target is

$$p_{ik} = E[P(Y_i^{rep} = k \mid b, \widehat{\psi}) \mid Y_{obs}, \widehat{\psi}],$$

where b includes ability, and for extensions the local effect or jointly uncertain shared-rater vector. This uses the same observed data for estimation and checking; it is not held-out prediction. Define

$$\mu_i = \sum_k k p_{ik}, \qquad V_i = \sum_k (k - \mu_i)^2 p_{ik}.$$

For these posterior calculations, the mixture variance includes conditional rating variance and variance of conditional means. Substituting an ability/rater estimate or averaging conditional variances alone gives a different quantity.

For selected observed unit-weight rows in a group, descriptive Outfit is the mean of $(y_i - \mu_i)^2/V_i$; descriptive Infit is $\sum_i (y_i - \mu_i)^2 / \sum_i V_i$. These summaries have no established expectation of one. There are no default cutoffs, ZSTD, p-values, automatic flags or rater-quality classifications. Ordinary plug-in Infit/Outfit values do not have the same probability definition and must not be compared directly. No response covariance or calibration uncertainty is supplied, so these results do not calibrate a formal goodness-of-fit test.

Ordinary RSMs integrate normal ability by quadrature, retaining every observed rating of the relevant Person. Supported ordinary fits have unit weights, additive unanchored severity facets, no interactions or shrinkage, Person as the noncentered facet, consecutive integer source categories, and either known N(0,1) or `population_formula = ~1`. The estimated population mean and variance are both retained. This separate route does not alter the plug-in indices from `diagnose_mfrm()`. Old fits that omitted scores without retaining original row positions need to be refitted from the complete assigned-score roster.

Testlets use nested normal quadrature. Shared raters use category-specific Laplace integrals of the complete likelihood times the replicate category probability, after integrating abilities by quadrature. Each numerator reoptimizes the joint latent rater mode; calibration is never refitted. The positive numerator integrals are normalized across categories. `NormalizationError` records the unnormalized probability sum minus one. This defect and agreement between quadrature orders are numerical diagnostics, not bounds on the Laplace approximation error. In particular, small defects do not certify accuracy with few raters or sparse ratings. Shared-rater work grows with requested rows and categories; use `rows` to inspect a subset while retaining the complete conditioning data.

For these posterior routes, missing scores remain `missing_score`; they are not imputed. Nonfinite or zero predictive variances and unresolved integration remain unavailable with a reason. A group containing any unavailable observed row has no summary, rather than silently dropping that row. Entirely missing groups have no summary. A fitted zero latent variance defines the corresponding submodel's predictions; it does not establish absence of heterogeneity.

`plot()` method: Plot descriptive response residual summaries

Compare residual summaries across selected groups without reference cutoffs.

For `plot()`: The response probabilities integrate the latent effects conditional on the same observed responses and fixed calibration. These are descriptive summaries, not ordinary plug-in Infit/Outfit. Neither an expectation of one nor a universal acceptable range is established. No interval, warning region or statistical test is drawn. Hiding annotations changes only the display; `plot_data()` retains the probability definition and limitations. For corrected JML, probabilities instead condition on corrected calibration and reprofiled Person point estimates; no latent effects are integrated. The paired view displays each available index independently, so an available Infit is retained when zero conditional variance makes Outfit undefined. The scatter view requires both indices. The caption reports partial availability, and the saved table retains its reasons.

Value

An mfrm_response_diagnostics object with rows, category probabilities, grouped measures, settings and exact source metadata. Save with saveRDS(); attach via mfrm_results(fit, response_diagnostics = result, compute = "never"). The resulting mfrm_results object contains tables$response_overview, distinguishing available standardized residuals, unresolved calculations, missing scores, zero-variance rows and source rows not selected. Partial results require review; availability does not establish model adequacy. No probabilities or residuals are recomputed for this table. Older saved mfrm_results objects without this table recover the overview and status from retained diagnostic rows when summarized, reported, exported or opened in a supported results viewer. The original object is not modified; newly exported results include the recovered overview and status. Supply two saved outputs to compare_mfrm() for aligned ordinary versus extended RSM comparisons, using the same selected events and group_by. Corrected-JML and two-family GPCM comparisons are not supported by that route. Collecting, plotting and exporting saved results never recomputes integrals.

For plot(): Invisibly, mfrm_plot_data with selected measures, display settings, notes and alternative text. as_ggplot() converts either view.

Two-family GPCM posterior residuals

Use this route to describe how observed scores differ from the fitted model's posterior predictions. For example, summarize by a Judge column with mfrm_response_diagnostics(fit, group_by = "Judge"). Both slope families multiply the complete adjacent-category predictor, as in fitting. Ability is integrated under the fitted fixed N(0,1) population and all of the Person's observed ratings; probabilities are not evaluated at the EAP. Only fitted observed rows are returned, preserving the original identifiers and incomplete assignment pattern. No unassigned cell is filled.

The saved solution and point calibration must agree with their fitting specification. A slope covariance or slope interval is not required. Fixed-grid EM and adaptive direct MML retain their respective integration methods. Adaptive grids use the complete Person posterior's mode and curvature, preserving the original N(0,1) prior through density-ratio and Jacobian weights. Source likelihood and engine-specific gradient checks are recomputed before prediction. Stored convergence flags alone do not establish a usable source. This has the same scope as the two-family fit: two fixed slope facets, unit weights, no anchors or covariates, and observed zero-based categories. quad_points checks response integration at the retained parameters; it does not refit the calibration. If integration is unresolved, increase it explicitly. The returned probabilities use 2 * quad_points + 1 points; the lower order supplies the row-wise check. Selecting fewer output rows does not shorten the posterior's conditioning record or change its grid. Settings retain the integration method through plots, reports and exports. Separately use mml_quadrature_sensitivity() to examine whether the fitted calibration changes across quadrature grids.

Inspect rows for unavailable predictions before reading measures. A larger descriptive Infit or Outfit is not a calibrated misfit decision; there is no expectation-one reference. These same-data summaries neither establish rater quality nor test unidimensionality or local independence. diagnose_mfrm(), Q3/PCA, Wright/Pathway maps and model ranking retain their separate support restrictions. Attach the saved result with mfrm_results(fit, include = c("fit", "plots"), compute = "never", response_diagnostics = result) for plots, reports and exports. Collection and replay do not refit or reintegrate the saved probabilities.

Corrected JML conditional residuals

For an explicit profile-score-adjusted GPCM fit, probabilities are evaluated at the saved corrected facet/step/slope estimates and the reprofiled Person abilities: $p_{ik} = P(Y_i = k \mid \widehat{\theta}_i, \widehat{\beta}_{adj})$. These are conditional plug-in probabilities, not posterior predictions or held-out predictions. There is no normal ability prior or quadrature. Calibration and Person estimation uncertainty are excluded. Residual bias may remain. Unavailable RootSE does not prevent these point calculations. The response formula is the same shared-owner GPCM used for fitting.

Only observed fitted rows are available; no unassigned cells are filled or missing scores imputed. Original row numbers, repeated-event multiplicity, category labels and facet names are retained. `rows` selects displayed and summarized events without changing their Person profiles. Save the result and attach it through `mfrm_results(fit, response_diagnostics = result)`; the saved calibration, slopes, Person profiles and source rows must match.

An extreme Person profile has a limiting probability of one at the minimum or maximum score. Its expected score, zero conditional variance and raw residual remain available. `StandardizedResidual` is missing with `Status = "zero_variance"`: the ratio at the saved probabilities is undefined. No automatic convention replacing `0/0` with zero is applied. `ProbabilityAvailable` distinguishes this case from a failed probability calculation. If all selected raw residuals and variances are available and their total variance is positive, Infit remains available, including those zero-variance rows. Outfit requires every selected standardized residual; otherwise it is missing, and the group is `partially_available` with a reason. The grouped `Available` column counts rows with finite standardized residuals; Infit can use all `Observed` rows despite that smaller count. No row is silently discarded. An all-zero-variance group has neither index. Explicitly selecting other rows changes the summarized set and must be reported as such.

These same-data residual summaries have no calibrated expectation-one reference, uncertainty intervals, p-values, automatic flags or rater-quality classifications. They do not establish correction of structural bias or replace `diagnose_mfrm()`, which is unavailable for this estimator. `plot()` offers paired and scatter views, including `as_ggplot()` conversion. A paired view retains available Infit when Outfit is missing; a scatter view needs both. Reports and exports preserve the probability definition and unavailable outcomes. Corrected-JML model comparison remains unavailable; portable scoring has its own calibration checks and conditional EAP target.

References

Tierney, L., & Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. Journal of the American Statistical Association, 81(393), 82–86. doi:10.1080/01621459.1986.10478240. This supports the ratio-of-integrals approximation, not universal accuracy for this rating design.

See Also

`plot.mfrm_response_diagnostics()`, `mfrm_results()`, `score_mfrm_random_rater()`, `predict.mfrm_testlet()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("mfrm_response_diagnostics", run.dontrun = TRUE)
## Not run:
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$testlet$fit
```

```
residuals <- example$testlet$diagnostics
# To recompute the posterior predictive residuals:
# residuals <- mfrm_response_diagnostics(fit, group_by = "Rater")
residuals$measures
plot(residuals)
mfrm_results(fit, response_diagnostics = residuals, compute = "never")

## End(Not run)
```

`mfrm_response_imputations` *Review multiple imputations of scores on assigned ratings*

Description

Check several completed versions of a rating table before analyzing them. Supply these completed data from an imputation model; this function checks that observed scores, rating assignments and identifiers are preserved. No person-by-facet grid is constructed. This function validates supplied imputations; it does not choose or fit an imputation model. Start with `review_mfrm_imputations()`, then use `fit_mfrm_imputed()` and `pool_mfrm_imputed()` for eligible fixed-facet analyses. The older name `mfrm_response_imputations()` is retained with its original `impute` argument.

Usage

```
review_mfrm_imputations(
  data,
  completed,
  person,
  facets,
  score,
  event_id,
  impute_ids,
  categories,
  assigned = NULL,
  imputation_model = NULL,
  missing = c("error", "omit")
)

mfrm_response_imputations(
  data,
  completed,
  person,
  facets,
  score,
  event_id,
  impute,
  categories,
```

```
  assigned = NULL,
  imputation_model = NULL,
  missing = c("error", "omit")
)

## S3 method for class 'mfrm_response_imputations'
print(x, ...)

## S3 method for class 'mfrm_response_imputations'
summary(object, ...)
```

Arguments

data	Original long-format rating roster, including missing scores.
completed	A list of at least two completed data frames, or a mids object from mice::mice() fitted to the long-format roster. Each completion must contain all original columns and rows. Row order may differ; rows are matched by event_id.
person, score, event_id	Column names. event_id uniquely identifies a rating event, including repeated ratings of the same person and facets.
facets	Nonempty character vector of facet column names.
impute_ids	Character vector of event IDs explicitly selecting missing scores on assigned ratings. Observed scores cannot be selected. These are values from the event_id column, not a column name or a logical switch. For example, c("E2", "E7") selects those two rating events.
categories	The full intended contiguous integer category vector, for example 0:4. It is preserved across all completed analyses.
assigned	Optional name of a complete logical column: TRUE denotes an assigned rating and FALSE an unassigned combination. Without this column every supplied row is declared assigned. Unassigned rows must have missing scores in the original and every completion.
imputation_model	For a list of completions, the saved model or a nonempty list containing its specification, settings and diagnostics. Required so that the provenance is retained. For a mids input, that object is retained automatically; omit this argument.
missing	How to handle assigned missing scores not selected by impute_ids: "error" (default) or explicit "omit". Omitted events remain in the roster and every completion, with missing scores, and are excluded from each analysis. This choice is not a correction for nonresponse.
impute	Compatibility name for impute_ids, used only by mfrm_response_imputations(). Supply one selection, not both argument names.
x, object	An object returned by review_mfrm_imputations() or its compatibility wrapper mfrm_response_imputations().
...	Unused for print and summary methods.

Details

The score must contain numeric integer category labels (numeric vectors, or character/factor labels such as `"0"`, `"1"`). Recoded sentinel missing values must already be `NA`. Identifiers, assignment indicators and observed values must not change. Only selected scores may be filled; missing auxiliary predictors may also be completed. For a `mids` input, its original data and score `where` selection must agree with this review. Numeric identifiers must remain distinct when converted to text. Numeric observed predictors are compared exactly, without rounding to text, and must remain numeric in completions; integer/double storage may differ.

A sparse assignment and a missing assigned score are different events. Neither absent roster rows nor explicit unassigned rows are imputed. An entirely imputed person or facet level is visible in `support`; its analysis depends on the imputation model, not on observed ratings for that level. Imputed links do not establish empirical connectedness.

Proper multiple imputation must include uncertainty about missing values and imputation parameters, reflect the ordinal score support and the person/facet dependence, and be compatible with the intended analysis. Include relevant observed predictors of nonresponse. An MAR analysis requires an adequate conditional model; informative missingness beyond observed predictors requires sensitivity analysis. Passing these software checks does not establish MAR, model adequacy or interval coverage.

For a wide-format or multilevel imputation model, reshape each completion back to the original long roster using event identities and supply the resulting list with the saved model. Do not average completed scores. See `vignette("mfrmr-response-imputation", package = "mfrmr")` for a joint RSM example with supplied posterior predictive completions, shared Person draws, sampling diagnostics, direct observed-score inference and a separate lower-score sensitivity analysis. The accompanying R/Stan script regenerates that example; it is not a general imputation engine. Under the same score model and ignorable missingness, observed-score MML can directly estimate the fixed-facet target without completing scores. MI does not create additional observed information.

Value

An `mfrm_response_imputations` object containing the original `data`, aligned `completed` data sets, `imputation_model`, an `events` table with assignment/observation/imputation/omission status, a `support` table counting original observed and imputed scores by person/facet level, and `settings`. No imputations or failed analyses are silently discarded.

See Also

`fit_mfrm_imputed()`, `pool_mfrm_imputed()`, `mfrm_cluster_imputed()`

Examples

```
# Small supplied completions to illustrate input checks, not a fitted imputer.
ratings <- data.frame(Event = paste0("E", 1:4), Person = c("P1", "P1", "P2", "P2"),
  Rater = c("A", "B", "A", "B"), Score = c(0, NA, 1, 2))
first <- second <- ratings
first$Score[2] <- 1
second$Score[2] <- 2
reviewed <- review_mfrm_imputations(ratings, list(first, second),
  person = "Person", facets = "Rater", score = "Score", event_id = "Event",
```

```
  impute_ids = "E2", categories = 0:2,
  imputation_model = list(method = "Illustrative supplied completions"))
reviewed$events  # Only E2 is imputed; the three observed scores are retained.
summary(reviewed)  # Original observed support and imputation counts per level.
```

`mfrm_results` *Build comprehensive first-screen MFRM results*

Description

Build comprehensive first-screen MFRM results

Usage

```
mfrm_results(
  fit,
  include = "standard",
  response_time = NULL,
  response_time_data = NULL,
  response_time_facets = NULL,
  response_time_score = NULL,
  output = c("object", "summary", "tables", "html"),
  diagnostics = NULL,
  compute = c("auto", "never"),
  predictions = NULL,
  intervals = NULL,
  scores = NULL,
  comparison = NULL,
  response_diagnostics = NULL,
  calibration_intervals = c("none", "normal"),
  calibration_level = 0.95
)
```

Arguments

`fit`	Output from `fit_mfrm()`, `run_mfrm_facets()`, `fit_mfrm_testlet()` or `fit_mfrm_random_rater()`. A standard long-format `data.frame` is also accepted when person and score columns can be inferred unambiguously from common names such as `Person` and `Score`; remaining measurement columns must use recognizable facet-role names. Ambiguous extra columns are rejected rather than guessed as facets. Native CML and comparison-only fits are excluded; use their `summary()` and the CML interval object's `plot()`, `apa_table()` and RDS saving. Imported source-package fits use their own `summary()` and point-only Wright map; comprehensive reports require a native mfrmr fit.
`include`	Result sections or purpose presets to include. Purpose presets are `"standard"`, `"publication"`, `"validation"`, `"facets"`, `"bias"`, `"misfit_review"`, `"linking"`, `"network"`, `"gpcm_review"`, and `"all"`. Section names include `"fit"`, `"diagnostics"`,

"tables", "precision", "reporting", "categories", "plots", "facets_fit", "bias", "misfit", "linking", "network", and "apa". When omitted, NULL or empty for a two-family GPCM fit, this selects c("fit", "plots"). Explicit presets and sections remain scope-checked.

response_time
: Optional response-time column name. When NULL and include contains "response_time", conservative column names such as ResponseTime, response_time, or RT are detected when available.

response_time_data
: Optional original long-format data containing the timing column. Required for already fitted objects unless the timing column is still present in fit$prep$data.

response_time_facets
: Optional facet columns for response-time summaries. Defaults to the fitted model's source facet columns when available.

response_time_score
: Optional score column for response-time summaries. Defaults to the fitted model's source score column when available.

output
: Return format: "object" for an mfrm_results object, "summary" for its compact summary, "tables" for a named list of available data frames, or "html" for a temporary HTML report.

diagnostics
: Optional matching output from diagnose_mfrm(). An mfrm_response_diagnostics() object is also accepted as an alias for response_diagnostics; do not supply it twice.

compute
: Diagnostic computation policy. "auto" preserves the standard behavior; "never" collects only sections that can be built without computing diagnostics and marks every requested dependent section as "not_computed". Matching supplied or stored diagnostics are still reused under "never". The "fit" and "pathway" result plots reuse these diagnostics too. Under "never", if none are available, their expected-score curves omit fit-statistic tables. To include those tables, supply matching diagnostics or rebuild the result with compute = "auto"; explicitly requesting include_fit_measures = TRUE without diagnostics under "never" gives an error. For two-family GPCM fits, both settings collect saved results without fitting, integration or diagnostic calculation; ordinary diagnostics remain "not_available".

predictions
: Optional saved predictions for a testlet or random-rater fit: the result of predict(fit, ...). Matching calibration, column roles, levels, settings and prediction-source metadata are required. Older predictions without this metadata must be regenerated from the saved fit before attachment; the fit itself does not need to be re-estimated.

intervals
: Optional saved mfrm_random_rater_intervals() result from the exact supplied random-rater fit. For a native GPCM fit, accepts saved slope intervals, curve intervals, a GPCM bootstrap result, or a named list of these. Two-family GPCM also accepts mfrm_facet_intervals() for locations and within-facet contrasts. All must match the exact fitted data, parameters, population and integration settings. For native RSM/PCM fits, accepts a saved mfrm_facet_intervals() or mfrm_population_intervals() result, or a named list combining these. Population intervals are delivered in tables, with their saved cautions in population_<name>_interpreta
No bootstrap or interval calculation is run by this function.

scores
: Optional saved score_mfrm_persons() result for an extension, or score_mfrm_random_rater() output, with matching source calibration. Native one- and two-family GPCM accept saved predict_mfrm_units() output with matching calibration, population and source identity. Two-family GPCM also accepts a format-6 score_mfrm_calibration() result. One-family portable scores keep their standalone reporting/export methods. For testlets this is an alias for predictions; supply it once. Shared raters can also attach response predictions and rater intervals. No Person scoring is run here. Complete source-roster identity is required for the additional model-aware map routes; see mfrmr_model_maps.

comparison
: Optional descriptive extended-model result from compare_mfrm(). Calibration, settings and observed/omitted events must match this fit. Retained tables and plot(..., type = "comparison") compare centered facet summaries without fitting or automatic ranking. When the comparison includes saved response diagnostics, its comparison_response_* tables and type = "response_comparison" plot retain aligned predictive quantities; select metric for other indices.

response_diagnostics
: Optional saved mfrm_response_diagnostics() output matching the fit's calibration and exact source roster. It is identity-checked and reused without integration. Ordinary RSM MML and testlet/shared-rater fits and two-family GPCM MML support these posterior summaries; corrected JML instead supplies conditional plug-in summaries. None has reference cutoffs. Two-family results default to include = c("fit", "plots") and only collect saved quantities. Use compute = "never" to also avoid computing ordinary plug-in diagnostics. Saved posterior summaries appear in response_* tables and plot(..., type = "response_diagnostics"). tables$response_overview counts selected, available, unresolved, missing-score, zero-variance and unselected rows; partial output requires review. Older saved results without this table recover the overview and matching status from their retained diagnostic rows during summary, reporting, export or supported viewing, without re-estimation. The original object is unchanged; newly exported results include the recovered metadata.

calibration_intervals
: For testlet and shared-rater fits, "none" (default) retains calibration estimates and approximate SEs with missing bounds. "normal" explicitly requests pointwise observed-information normal intervals for fixed facets and steps; finite-sample coverage is not established. Numerical, information and estimated-boundary restrictions remain. Saved result tables, testlet calibration plots and reports retain the selection. This does not select individual-rater or Person intervals.

calibration_level
: Nominal level for those calibration intervals, between zero and one; default 0.95. Nondefault calibration options are unavailable for ordinary MFRM fits.

Details

mfrm_results() is a high-level result object. It does not introduce a new estimator or a new validity rule. It fits only when fit is a data frame, computes diagnostics automatically when needed, and collects output from functions such as diagnose_mfrm(), fit_measures_table(), precision_review_report(), and reporting_checklist(). Sections that are unsupported for

a particular fit are retained in the status table as not_available rather than stopping the whole results workflow. The additive readiness table keeps Numerical, Data, Design, Stability, Diagnostics, Reporting, and Plot interpretation states separate. Plot routes can therefore remain available for diagnosis while InterpretationStatus marks them as review-only. The returned object also carries next_actions and input$reproducible_code so users can move from the comprehensive first screen to explicit reporting or replay code. The examples call this object res to distinguish it from results <- summary(fit) in the quick start. Use mfrm_report(res) and export_mfrm_results(res) for reporting and export; those functions need the comprehensive results object. If diagnostics were already computed, pass them with diagnostics = diagnostics to reuse them.

Value

Depending on output, an mfrm_results object, a summary.mfrm_results object, a named table list, or an mfrm_results_html object.

Include presets

- "standard": fit, diagnostics, tables, precision, reporting, categories, and plot routes
- "publication": standard sections plus APA output assembly
- "validation": standard sections plus FACETS-fit/df-sensitivity review
- "facets": fit, diagnostics, tables, categories, plots, and FACETS-fit review for FACETS-facing migration work
- "bias" / "bias_review": standard sections plus facet-level bias-screen guidance; interaction bias still requires explicit facet-pair selection
- "misfit" / "misfit_review": standard sections plus unexpected-response, displacement, and pathway-map case-review surfaces
- "linking" / "anchors": standard sections plus anchor-readiness and operational linking-review surfaces from the fitted object's stored anchor review; drift and screened-chain review still require multiple fitted forms or waves
- "network": standard sections plus network/connectivity review
- "response_time": descriptive response-time QC review when timing metadata are supplied through response_time / response_time_data
- "gpcm_review": standard sections with GPCM caveats retained in the collected summaries and reports
- "all": standard sections plus FACETS-fit, network, APA, and response-time sections

Response-time metadata

Response-time review is opt-in and descriptive. It does not change fitted MFRM estimates, fit a joint speed-accuracy model, or create automatic exclusion rules. Use include = "response_time" together with response_time = "ResponseTime". When fit is an already fitted object, also supply response_time_data = original_data because fitted objects keep only the measurement columns needed for estimation.

What to inspect first

Start with `summary(res)`. The most useful fields are:

- `overview`: input mode, model, method, table count, and plot-route count
- `decision`: plain-language interpretation, formal-inference, reason, and next-action text derived from source-fit readiness and the saved diagnostic precision profile. Missing precision remains unreviewed; a positive precision assessment cannot override source-fit restrictions. Reading this summary does not compute new diagnostics or establish sampling coverage.
- `readiness`: separate analysis and plot-interpretation checks
- `fit_readiness`, `fit_readiness_components`, and `fit_readiness_parameters`: the exact source-fit readiness record retained separately from the workflow-level `readiness` table
- `triage`: first-screen signals ordered by unavailable/review/info/ok
- `status`: which sections were available, skipped, or unsupported
- `plot_map`: supported plot routes, availability, and interpretation status
- `next_actions`: recommended follow-up calls
- `reproducible_code`: replay script for the first-screen route

Data-frame input

Direct data-frame input is intentionally narrow. It accepts unambiguous `Person` / `Score` columns and familiar facet-role names such as `Rater`, `Item`, `Task`, or `Criterion`, and fits the `RSM` / `MML` route. It stops when other columns could be metadata, grouping variables, or background variables rather than silently treating them as measurement facets. For research scripts, use `fit_mfrm()` explicitly so column roles, model, method, anchors, and missing-data rules are recorded. Use `mfrm_results_interactive()` only for opt-in column selection at the console.

Visualization and HTML

For ordinary models, `plot(res)` routes to the primary native Wright map when the fitted object contains compatible person and facet locations. This default retains available mfrmr facet uncertainty. Use `plot(res, type = "fit")` when the explicit three-plot Wright/pathway/category bundle is wanted. The compact native default discloses any omitted facet locations in its subtitle and `data$retention`; use `plot(res, top_n = Inf)` for a complete final map. Other routes include `plot(res, type = "wright")`, `"pathway"`, `"fit_pathway"`, `"qc"`, `"category"`, `"anchors"`, `"response_time"`, and `"tables"`. The Wright map is the required first fitted-scale figure; `"fit_pathway"` is a follow-up with Infit or Outfit on the horizontal axis and measure on the vertical axis. `output = "html"` writes a lightweight temporary HTML file; use `launch_mfrmr_viewer()` when you want an optional local Shiny reader for an already-created `mfrm_results` object. Use `export_mfrm_results()` for a compact analysis archive of the comprehensive results object, or `export_mfrm_bundle()` when a fit-centered durable analysis archive is needed. Neither route deidentifies its contents.

Typical workflow

1. Fit explicitly with `fit_mfrm()` in scripts and manuscripts.
2. Call `res <- mfrm_results(fit)`.

3. Read `summary(res, view = "brief")` and its `readiness` table, then create the required `plot(res, type = "wright", show_ci = TRUE, top_n = Inf)` figure.
4. Read `summary(res)$triage`, `summary(res)$status`, `summary(res)$plot_map`, and `summary(res)$next_actions`.
5. Call `report <- mfrm_report(res)` when a report-ready surface is needed.
6. Use `export_mfrm_results(res, preset = "starter")` to write the Wright map, CSV, report, RDS, replay, and manifest files for controlled review. Treat the folder as potentially identifying unless it has been separately transformed and reviewed under the applicable data-handling policy.
7. Use `plot(res, type = "fit_pathway", include_person = TRUE, top_n_person = 12, person_labels = "none", facet_labels = "flagged")` or `plot(res, type = "qc")` for focused visual follow-up.
8. Optionally inspect the same result with `launch_mfrmr_viewer()` in an interactive session.
9. Use `build_summary_table_bundle()` or the helper named in `summary(res)$next_actions` for report-specific follow-up.

Reproduce attached inference

When additional inference or posterior diagnostics are attached, the code shown by `summary(res)` and in HTML saves and reloads the complete `res` object. Choose the file path before running it. This preserves attached methods, levels and unavailable results without fitting or recalculating. Exported replay scripts only reload the RDS already written by the export. The starter export index links the saved RSM/PCM and GPCM inference figures alongside the ordinary maps, whose uncertainty and fit meanings are separate.

Saved RSM/PCM fixed-facet intervals

For supported MML fits, use `intervals = list(raters = ci)` with output from `mfrm_facet_intervals()`. Native CML and comparison-only fits are not inputs to this collector. For CML, use `summary(fit)` and save the interval result separately with `saveRDS(ci, file)`; `summary(ci)`, `plot(ci)` and `apa_table()` reuse that result without an ordinary report bundle. The fit must match the saved data, parameters, constraints, population and integration settings. Named results give `facet_` plot routes, for example `plot(res, type = "facet_raters")`; a single result gives `facet_inference`. Include `"plots"` to enable those routes. `as_ggplot()` supports customization. Tables, reports and exports preserve the selected method, confidence level, contrasts, cluster mapping and fixed/unavailable targets. Export replay reloads saved results without refitting or changing the selected intervals. These are pointwise fixed-facet intervals, not simultaneous rater decisions or uncertainty for replacement raters. Ordinary map and fit displays keep their own meanings and are not recalculated with the attached covariance.

Corrected JML results

For an explicit correction fitted by `fit_mfrm()`, this helper collects saved estimates, numerical status and the RootSE interpretation without computing ordinary diagnostics. Unavailable requested sections are labelled in `status`. `plot()` offers `"slopes"`, `"locations"` and `"steps"`, with point or cumulative-distribution displays and `as_ggplot()` conversion. `mfrm_report()` and `export_mfrm_results()` retain the estimator, correction order, unavailable inference and covariance-failure reasons. Saved replay reloads the result without refitting. Attach saved `mfrm_response_diagnostics()`

output through response_diagnostics to include conditional probabilities, expected scores and descriptive Infit/Outfit. The source and probability definition are checked; collection never recomputes probabilities. Use plot(res, type = "response_diagnostics") for paired or scatter views, retaining available Infit when Outfit is undefined. Ordinary diagnostic, comparison, prediction and interval attachments and individual rater sheets are unavailable; a numerical solution is not a rater-quality judgment.

Provisional two-family results

For a two-slope-family GPCM fit, mfrm_results(fit) collects its saved estimates and numerical status, with include = c("fit", "plots") when omitted. Neither compute setting fits, integrates or calculates diagnostics. The gpcm_fitted_slopes, gpcm_fitted_locations and gpcm_fitted_steps tables retain all saved non-Person estimates when "fit" is included. gpcm_interpretation_notes retains the incomplete method qualification even when no intervals are attached; it accompanies the result/report CSV tables. Locations retain their centering reference; steps identify their owner and level and are centered offsets within that level. Separately computed experimental mfrm_facet_intervals() results may be attached through intervals = list(locations = ci); step intervals remain unavailable. Reports and CSV exports retain these values; the report shows at most 20 rows of each table with an explicit notice. Explicit unsupported sections or attachments are rejected. Attach saved mfrm_curve_intervals() results through intervals; these contain provisional fitted values without intervals. Summary, tables, named curve plots, reports and saved exports retain that limitation. Separately requested confint(fit, parm = "slopes") results can be attached as intervals = list(slopes = ci, curves = curves). Their experimental component-slope approximation, owner identities, numerical checks and unavailable outcomes follow the saved tables, plots and report. If exactly one saved curve, slope/location-interval or response-diagnostic plot is available, plot(res) displays it. With multiple saved plots, select type from summary(res)$plot_map; no new curves or diagnostics are calculated. Replay code saves and reloads the complete result, including the fit's owner identities, estimation settings and unresolved numerical status. Attach saved conditional new-Person output as scores = scores, from predict_mfrm_units() or the format-6 portable calibration workflow. The source identity must match this fit's calibration, data, category/owner coding, integration settings and recorded status. Collection does not repeat source checks or scoring. New source identifiers exclude the writing R version and attribute insertion order. Older scores without an identity or with a version-dependent identity still support their standalone methods; regenerate them if attachment is refused (also re-extract an older portable calibration), without re-estimating the fit. person_scores and scoring_* tables retain unrounded estimates, conditional interval meanings, source/batch checks, omissions and review labels in reports, CSV exports and saved replay. Native scoring records omitted row counts but does not enumerate a not-scored Person roster; portable scoring retains it. This attachment adds tables and reports; it does not add a score plot route. The intervals exclude calibration uncertainty; coverage and population transport remain unqualified. extract_mfrm_calibration() provides the separately checked portable route without retaining training responses. Ordinary diagnostics, Wright/Pathway maps and model comparison are unavailable. The starter export uses the available saved curves instead of requesting an unsupported Wright map.

Saved one-family GPCM Person scores

First compute scores <- predict_mfrm_units(fit, new_ratings) under the intended scoring prior, then attach them with res <- mfrm_results(fit, scores = scores, include = "fit", compute = "never"). This accepts the existing ordinary MML/JML scoring routes and their recorded review

restrictions; it does not extend scoring to new designs or estimators. Collection checks the fitted calibration, data, population coding and numerical state, plus the saved prior, interval and integration records. It does not refit, rescore or repeat source numerical checks. Older scores without compatible fitted-analysis identity or check records must be recreated from the saved fit before attachment; no model refit is needed.

`res$tables$person_scores` retains EAP (`Estimate`), posterior `SD`, continuous equal-tail posterior bounds (`Lower`, `Upper`), level and numerical status. Posterior SD is neither the SD of EAPs across Persons nor the population SD or a calibration-parameter SE. These bounds differ from the diagnostic EAP +/- normal-multiplier times posterior-SD bands elsewhere in the report. Neither route includes calibration/population estimation uncertainty or establishes frequentist coverage for fixed abilities.

`scoring_priors` retains each Person's fitted and scoring priors, including explicit prior overrides and covariate-dependent means. The scalar `PriorMean`/`PriorSD` in `scoring_settings` are NA when they differ by Person; use `scoring_priors` instead. `scoring_integration`, `scoring_row_review` and any `scoring_population_review` retain checks and omissions. Native scoring does not enumerate a complete not-scored Person roster: `NotScored` is NA. Review-only scores stay review-only even when calibration-parameter inference has a different readiness state. JML-derived EAP scores are post hoc posterior summaries, not the original JML Person estimates.

`mfrm_report()` and `export_mfrm_results()` retain all attached tables and complete saved scoring objects. Replay reloads them without recomputing. This adds no score plot route; ordinary diagnostic maps keep their existing targets and are not replaced by the attached scores.

Saved GPCM inference

Attach explicitly selected results, for example `intervals = list(slopes = confint(fit, parm = "slopes", scale = "standardized"))`. Tables retain the target, method, confidence level and multiplicity choice; `mfrm_report()` and `export_mfrm_results()` preserve them. Names become lowercase plot routes, e.g. `plot(res, type = "gpcm_slopes")`, when `"plots"` is included. A single result uses `"gpcm_inference"`. Passing a raw slope bootstrap object selects its default relative 95% pointwise intervals; pass `confint(bootstrap, ...)` to select a different target or level. Replay reloads saved results, without refitting or changing interval methods. Older intervals lacking source metadata must be recomputed from the saved fit. These results supplement ordinary location and fit displays; discrimination is not rater quality and does not select scoring weights.

Testlet and random-rater results

`fit_mfrm_testlet()` and `fit_mfrm_random_rater()` results use a separate reporting route within the same `mfrm_results` class. It always retains calibration estimates, numerical checks, settings, data usage and interval meanings. Requested ordinary-model sections that are unsupported are marked `not_available`. Neither `compute` setting fits, scores, resamples or computes diagnostics for these models. Supply separately computed `predictions`, `scores` or `intervals` explicitly; their absence is recorded. Rebuild results from an older saved fit to apply current calibration-bound defaults without refitting. Previously saved result bundles keep their original tables. Explicit normal bounds do not improve their coverage.

Use `mfrm_report()` for a static report and `export_mfrm_results()` for CSV/HTML/RDS and figures. Numerical convergence does not establish model adequacy or general interval coverage. The ordinary Wright map, residual diagnostics, Shiny viewer, response-MI pooling and portable-calibration workflow do not support these classes. Plot types are `"calibration"` (first fixed facet

by default; select with `facet`) and `"scores"` for testlets, or `"raters"`, `"intervals"` and `"scores"` for random-rater results. Both support `"comparison"` and, with corresponding saved quantities, `"response_comparison"` and `"response_diagnostics"`. These models also support `"wright"` and `"fit_pathway"` through the separately defined conditional-location and posterior-residual routes in mfrmr_model_maps, with matching saved source evidence. All these routes require `"plots"` in `include`; saved predictions/intervals are required for their corresponding plots. Report styles use the same stored evidence and boundaries; they do not add ordinary-model reporting templates.

See Also

`fit_mfrm()`, `run_mfrm_facets()`, `diagnose_mfrm()`, `reporting_checklist()`, `build_summary_table_bundle()`, `export_mfrm_results()`, `launch_mfrmr_viewer()`, `mfrmr_output_guide()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mfrm_results", run.dontrun = TRUE)
## Not run:
# Load the package and example ratings
library(mfrmr)
ratings <- load_mfrmr_data("example_operational")

# Fit these ratings; replace the data-loading line for your own analysis
fit <- fit_mfrm(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  rating_min = 1, rating_max = 4, category_policy = "preserve",
  method = "MML",
  model = "RSM"
)

# Build the fuller results object, including diagnostics
res <- mfrm_results(fit)
review <- summary(res)
review$decision
review$next_actions

# Draw the Wright map
plot(res)

## End(Not run)
```

`mfrm_results_interactive` *Interactively choose data-frame columns before calling* `mfrm_results()`

Description

Interactively choose data-frame columns before calling `mfrm_results()`

Usage

```
mfrm_results_interactive(
  data,
  include = "standard",
  output = c("object", "summary", "tables", "html")
)
```

Arguments

`data`	A long-format data frame.
`include`	Passed to `mfrm_results()`.
`output`	Passed to `mfrm_results()`.

Details

This helper is deliberately opt-in and stops in non-interactive sessions. It asks the user to choose the person, score, optional weight, and facet columns, prints reproducible code for the selected roles, then fits the documented RSM/MML starting route before calling `mfrm_results()`. Use explicit `fit_mfrm()` calls in scripts, Quarto documents, batch workflows, and reproducible analyses.

Value

The selected `mfrm_results()` output.

Why this helper is opt-in

Interactive prompts are useful at the console but are unsafe defaults for reproducible analysis, unattended workflows, and manuscripts. The helper therefore prints replay code and leaves the scripted route explicit.

See Also

`mfrm_results()`, `fit_mfrm()`, `run_mfrm_facets()`

Examples

```
if (interactive()) {
  toy <- load_mfrmr_data("example_core")
  res <- mfrm_results_interactive(toy)
}
```

`mfrm_screening_performance` *Evaluate screening outcomes against a planned simulation roster*

Description

In a simulation where the true problem status is known, summarize how often a warning rule detects affected raters or flags unaffected raters. Supply the planned trials and observed flags; this function does not generate or fit simulated ratings. Unknown truth in real ratings cannot supply these rates. Unavailable screens remain unavailable, and correlated raters are not counted as independent simulation trials.

Usage

```
mfrm_screening_performance(roster, results, rule, level = 0.95)

## S3 method for class 'mfrm_screening_performance'
summary(object, ...)

## S3 method for class 'mfrm_screening_performance'
print(x, ...)

## S3 method for class 'mfrm_screening_performance'
plot(
  x,
  scope = c("family", "target"),
  metric = c("false_flags", "detection"),
  condition = NULL,
  draw = TRUE,
  ...
)
```

Arguments

`roster`	Data frame with `Condition`, `Replicate`, `Target` identifiers and logical `Affected`: `TRUE` denotes the prespecified departure that the screen is intended to detect. Include every planned target and replication, including failed runs. Within a condition, use the same targets and truth labels in every replication. Put different designs or methods in separate conditions. Identifiers are matched exactly as character labels.
`results`	Data frame with the same three identifier columns and logical `Flag` (`TRUE`, `FALSE` or `NA`). Omitted result rows remain unavailable. Extra or duplicate keys are refused. Additional columns are retained in the source results, for example numerical status and failure messages.
`rule`	Nonempty description of the prespecified screen, including its thresholds, comparison family and any selection/refitting procedure.

level	Confidence level for exact binomial Monte Carlo intervals; default 0.95. These describe simulation uncertainty, not an interval for rater severity or a guarantee of screening accuracy.
object	An object returned by mfrm_screening_performance().
...	For summary.mfrm_screening_performance(): Unused by print and summary methods. For plot(): Unused.
x	Output from mfrm_screening_performance(). Also for summary(): An object returned by mfrm_screening_performance().
scope	"family" (default) shows any-flag events once per replication; "target" shows each individual target separately.
metric	"false_flags" (default) or "detection".
condition	Optional condition labels to include, matched exactly.
draw	Logical; FALSE returns exact plot data without opening a device.

Details

A target's rate is sensitivity when Affected = TRUE and a false flag rate when Affected = FALSE. The family summaries count whether any unaffected target was flagged, and whether any affected target was flagged, once per replication. They do not pool all raters into a binomial sample. An observed positive makes an any-flag event known even if another target is unavailable. Without a positive, any unavailable member leaves the event unknown; only a complete negative family establishes no flag.

Replications must be independent and identically generated within each condition for the binomial uncertainty calculation. Pairing conditions does not invalidate their separate intervals, but a comparison between paired conditions needs its own paired analysis. Zero observed false flags does not imply a zero population probability: the exact upper bound stays positive. Conditional rates may be biased for the intended all-trial rate if failures depend on outcomes. Inspect availability and all-trial bounds together.

This helper neither creates flags nor establishes their validity. A severity difference alone is not rater misfit. Known simulation truth is required; labeling unknown real-world rater quality does not create a validation study. Define computable screening output separately from formal inference eligibility. The recorded rule cannot prove that it was selected before seeing outcomes.

plot() method: Plot screening performance and unresolved outcomes

For plot(): Blue points and whiskers show conditional rates and their exact binomial Monte Carlo intervals. Gray segments bound the realized all-trial proportion by assigning unresolved outcomes negative or positive; these are not confidence intervals. A row with no known outcomes has no point estimate. Right-hand counts show available/planned events. No model is fitted and no threshold is selected by this plot.

Value

An mfrm_screening_performance object with by_target, by_family, aligned outcomes, replication-level family_outcomes, the original roster and results, and settings. Tables retain full precision. Planned, Available, Unavailable and Positive give denominators and counts. Rate

and MCSE condition on available outcomes; MCLower and MCUpper are exact binomial bounds. AllTrialsLower and AllTrialsUpper bound the realized proportion across all planned trials by assigning every unknown outcome negative or positive. These latter bounds are not confidence intervals. A family with no relevant targets has zero planned trials and undefined rates. CompleteScreens counts fully observed families.

For plot(): Invisibly, an mfrm_plot_data object. Use plot_data() for custom graphics; automatic as_ggplot() conversion is unavailable.

References

Morris, T. P., White, I. R. and Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38, 2074–2102. doi:10.1002/sim.8086.

See Also

mfrm_screening_sensitivity() for several threshold bands, evaluate_mfrm_signal_detection(), facet_quality_dashboard()

Examples

```
roster <- expand.grid(Condition = "Null", Replicate = 1:4,
                      Target = c("R1", "R2"), stringsAsFactors = FALSE)
roster$Affected <- FALSE
results <- roster[c("Condition", "Replicate", "Target")]
results$Flag <- c(FALSE, TRUE, FALSE, NA, FALSE, NA, FALSE, NA)
performance <- mfrm_screening_performance(roster, results,
  rule = "Infit or Outfit outside [0.5, 1.5]; both raters; no exclusion/refit")
summary(performance)
plot(performance)
```

---

mfrm_screening_sensitivity

Compare fit-screening thresholds against known simulation truth

---

Description

See how detection and false-flag rates change when you change fit-screening thresholds in a simulation with known truth. Apply specified mean-square bands to saved Infit/Outfit values without refitting models. Keep underfit, overfit and their union separate. Here sensitivity analysis means comparing thresholds; the reported Sensitivity rate means detecting an affected target.

Usage

```
mfrm_screening_sensitivity(
  roster,
  measures,
  thresholds,
  rule,
```

```
  statistic = c("either", "infit", "outfit"),
  zstd_cut = NULL,
  level = 0.95
)

## S3 method for class 'mfrm_screening_sensitivity'
summary(object, ...)

## S3 method for class 'mfrm_screening_sensitivity'
print(x, ...)

## S3 method for class 'mfrm_screening_sensitivity'
plot(
  x,
  style = c("tiles", "curves"),
  direction = "either",
  metric = c("false_flags", "detection"),
  condition = NULL,
  quantity = c("rate", "unavailable"),
  draw = TRUE,
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  text_scale = 1,
  point_size = 2.5,
  ...
)
```

Arguments

`roster`	Data frame with `Condition`, `Replicate`, `Target` identifiers and logical `Affected`: `TRUE` denotes the prespecified departure that the screen is intended to detect. Include every planned target and replication, including failed runs. Within a condition, use the same targets and truth labels in every replication. Put different designs or methods in separate conditions. Identifiers are matched exactly as character labels.
`measures`	Data frame with `Condition`, `Replicate`, `Target`, `Infit` and `Outfit`. Include `InfitZSTD` and `OutfitZSTD` when requesting the combined ZSTD rule. The selected statistic's columns are required; nonfinite values and omitted rows remain unavailable. Additional source columns, such as numerical readiness, are retained without filtering.
`thresholds`	Data frame with unique nonempty `Profile` labels and numeric `Lower`, `Upper` bounds satisfying `0 < Lower < Upper`. Profiles are compared in supplied order, without selecting an optimum.

rule	Nonempty description of the prespecified screen, including its thresholds, comparison family and any selection/refitting procedure.
statistic	"either" (default) flags Infit OR Outfit; "infit" or "outfit" uses only the specified index.
zstd_cut	NULL (default) uses only mean squares. A positive finite number adds directional ZSTD flags with inclusive boundaries. This is an explicit alternative rule, not a significance calibration. Record the chosen df convention and residual definition in rule.
level	Confidence level for exact binomial Monte Carlo intervals; default 0.95. These describe simulation uncertainty, not an interval for rater severity or a guarantee of screening accuracy.
object	An mfrm_screening_sensitivity object.
...	For summary.mfrm_screening_sensitivity(): Unused by print and summary. For plot(): Unused.
x	Output from mfrm_screening_sensitivity(). Also for summary(): An mfrm_screening_sensitivity object.
style	"tiles" shows a labelled rate matrix. "curves" shows profile comparisons with pointwise Monte Carlo intervals by condition.
direction	One or more of "either", "underfit", "overfit".
metric	"false_flags" or "detection" chooses the family event.
condition	Optional condition labels to display.
quantity	"rate" (default) or "unavailable", the proportion of planned family events whose outcome is unresolved.
draw	For plot(): FALSE returns plot data without opening a device.
palette	"accessible" uses a sequential blue scale for tiles and blue/orange/purple lines with different symbols. "mono" uses greys.
title, caption	NULL uses the default text; a string replaces it; "" removes it. Interpretation notes remain in the saved plot data.
show_title, show_notes	Show the title and caption, respectively. FALSE hides text without deleting its metadata.
show_labels	Show percentages and known/planned counts in tiles.
text_scale	Positive multiplier for text size.
point_size	Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

Details

Mean-square boundaries are strict: equality is inside the band. Missing values use three-valued logic: TRUE OR NA is TRUE, but FALSE OR NA is NA. The family event counts any affected or any unaffected target once per independent replication. Correlated raters are not independent trials. The source roster fixes truth and planned denominators for every profile. Affected denotes

the departure of interest; when comparing directional screens, an affected-target flag can have the wrong direction for that departure. Inspect directions together, not just the union's sensitivity.

Thresholds are review heuristics, not universal error-controlled tests. A low mean square describes low residual variability, not poor rater quality or proof of misconduct. ZSTD is sensitive to sample size and its df convention. Selecting a threshold on these results and reporting its same- sample performance is optimistic; use a separately designed validation. Pointwise Monte Carlo intervals are not simultaneous across profiles or conditions. Their overlap is not a paired test of rules applied to the same replications. No automatic exclusion or refitting is performed.

This helper does not qualify the supplied statistics. In particular, ordinary Rasch bands must not be transferred to extended-model posterior- predictive residual summaries that lack the same reference distribution.

`plot()` method: Plot directional screening rates across threshold profiles

For `plot()`: Tile shading is supplemented by numeric labels; missing estimates are labelled NA. Profile order is supplied order, not an optimized ranking. Curves connect discrete profile choices, not a continuous threshold scale. Tile plots do not display uncertainty intervals: consult the retained table or curve view. Exact intervals are pointwise and conditional on known outcomes; all-trial bounds are retained, not confidence intervals.

Value

An `mfrm_screening_sensitivity` object with `by_target`, `by_family`, aligned `outcomes`, replication-level `family_outcomes`, source `roster` and `measures`, `thresholds` and `settings`. Each result table identifies `Profile`, `Lower`, `Upper` and `Direction`; the remaining columns have the meanings documented by `mfrm_screening_performance()`.

For `plot()`: Invisibly, `mfrm_plot_data` with exact selected tables, settings, notes and alternative text. `as_ggplot()` supports either style.

References

Linacre, J. M. (2003). Size vs. significance: standardized chi-square fit statistic. *Rasch Measurement Transactions*, 17(1), 918. https://www.rasch.org/rmt/rmt171n.htm.

See Also

`fit_measures_table()`, `mfrm_screening_performance()`

Examples

```
roster <- expand.grid(Condition = "Null", Replicate = 1:4,
                      Target = c("R1", "R2"), stringsAsFactors = FALSE)
roster$Affected <- FALSE
measures <- roster[c("Condition", "Replicate", "Target")]
measures$Infit <- c(.45, .8, 1.1, NA, .9, 1.4, 1.2, NA)
measures$Outfit <- measures$Infit
bands <- data.frame(Profile = c("Broad", "Narrow"),
                    Lower = c(.5, .7), Upper = c(1.5, 1.3))
sensitivity <- mfrm_screening_sensitivity(roster, measures, bands,
  rule = "Illustrative fixed-rater residual screen; no exclusion/refit")
plot(sensitivity, direction = "overfit")
```

mfrm_threshold_profiles *List literature-based warning threshold profiles*

Description

List literature-based warning threshold profiles

Usage

```
mfrm_threshold_profiles()

## S3 method for class 'mfrm_threshold_profiles'
summary(object, digits = 3, ...)
```

Arguments

object	Output from mfrm_threshold_profiles().
digits	Number of digits used for numeric summaries.
...	Reserved for generic compatibility.

Details

Use this function to inspect available profile presets before calling build_visual_summaries().

profiles contains thresholds used by warning logic (sample size, fit ratios, PCA cutoffs, etc.). pca_reference_bands contains uncalibrated descriptive references used in summary text; these are not dimensionality-test critical values.

summary() method: Summarize threshold-profile presets for visual warning logic

For summary(): Summarizes available warning presets and their PCA reference bands used by build_visual_summaries().

Value

An object of class mfrm_threshold_profiles with profiles (strict, standard, lenient) and pca_reference_bands.

For summary(): An object of class summary.mfrm_threshold_profiles.

Interpreting output

- profiles: numeric threshold presets (strict, standard, lenient).
- pca_reference_bands: narrative reference bands for PCA interpretation.

For summary():

- thresholds: raw preset values by profile (strict, standard, lenient).
- threshold_ranges: per-threshold span across profiles (sensitivity to profile choice).

- pca_reference: literature bands used for PCA narrative labeling.

Larger Span in threshold_ranges indicates settings that most change warning behavior between strict and lenient modes.

Typical workflow

1. Review presets with mfrm_threshold_profiles().
2. Pick a default profile for project policy.
3. Override only selected fields in build_visual_summaries() when needed.

For summary():

1. Inspect summary(mfrm_threshold_profiles()).
2. Choose profile (strict / standard / lenient) for project policy.
3. Override selected thresholds in build_visual_summaries() only when justified.

See Also

build_visual_summaries()

mfrm_threshold_profiles(), build_visual_summaries()

Examples

```
profiles <- mfrm_threshold_profiles()
s_profiles <- summary(profiles)
s_profiles$overview
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_threshold_profiles", run.dontrun = TRUE)
## Not run:
profiles <- mfrm_threshold_profiles()
summary(profiles)

## End(Not run)
```

---

mml_quadrature_sensitivity

Review MML sensitivity to the quadrature grid

Description

Refit one ordered-response MML model to the same supplied response data at two or more Gauss–Hermite quadrature counts. The original fit is reused at its own quadrature count; all other fits reuse its stored model, identification, anchor, interaction, optimizer, and population settings.

Usage

```
mml_quadrature_sensitivity(
  fit,
  data,
  quad_points = c(31L, 41L),
  theta_range = c(-4, 4),
  theta_points = 161L,
  adaptive_quad_points = NULL
)

gpcm_mml_quadrature_sensitivity(
  fit,
  data,
  quad_points = c(31L, 41L),
  theta_range = c(-4, 4),
  theta_points = 161L,
  adaptive_quad_points = NULL
)
```

Arguments

`fit`	An RSM, PCM, or GPCM MML `mfrm_fit` returned by `fit_mfrm()`. `gpcm_mml_quadrature_sensitivi` accepts only GPCM fits.
`data`	The original response data.frame used to create `fit`. Prepared response rows are compared semantically after refitting; row order may differ, but changed observations fail closed.
`quad_points`	At least two distinct positive integers, including the quadrature count stored on `fit`. The default `c(31, 41)` is a comparison starting point, not a claim that either grid is adequate for the data. Refits retain the source fit's fixed or adaptive integration mode.
`theta_range`	Two finite values defining the common ability grid used for fitted category-probability comparison.
`theta_points`	Number of common-grid ability points; at least 21.
`adaptive_quad_points`	Optional vector of at least two distinct integer orders >= 3, for example `c(31, 61)`. Adds a per-Person `quadrature_review` at each fit's fixed parameters, comparing a fixed-prior grid with mode/curvature-adapted grids. This separates integration error from parameter changes during refitting. Inspect changes between adaptive orders too; neither grid is certified exact. Also available for two slope families as a numerical diagnostic; it does not add Person scoring or change the fitted integration method.

Details

This is an explicit refit diagnostic: neither `summary.mfrm_fit()` nor `diagnose_mfrm()` invokes it automatically. It reports continuous changes rather than classifying a fit as quadrature-stable or

unstable. In particular, it does not select a cutoff for acceptable changes, make unavailable standard errors valid, or resolve other warnings about the fit.

The probability comparison evaluates every observed combination of non-Person facet levels on the same theta grid. It includes fitted two-way facet interactions and, for GPCM, the complete-predictor slope action. Same-Person EAP and posterior-SD changes use each fit's corresponding quadrature count. A one-point grid has no public scoring route, so those two changes are `NA` when it is the reference.

Explicit intercept-only and covariate population models reuse the retained person table, factor coding and formula, with design equality checked by Person. Older fits without that data cannot be replayed.

Raw slope and population-SD standard errors are computed from each local observed-information Hessian for diagnostic comparison only. The public parameter-level `SEEligible` state remains unchanged.

Value

An object of class `mfrm_quadrature_sensitivity` containing:

- `summary`: one comparison row per quadrature grid relative to the original fit, including likelihood, measurement-coordinate, probability, EAP, and posterior-SD changes;
- `quadrature_review`: when requested, unrounded per-Person fixed/adaptive log-marginal, EAP and posterior-SD differences, adaptive-order changes, mode/curvature and computation status. Unavailable rows retain their reason. `summary()` also supplies a compact `quadrature_overview`;
- `runs`: likelihood, gradient, curvature, population-scale, and readiness details for each fit;
- `slopes`: component-slope estimates, raw diagnostic SEs and freshly checked 95% model intervals. The summary reports endpoint changes among jointly eligible levels and counts changes in interval availability. One-family slopes have geometric mean one; two-family scale references differ;
- `intervals`: for two families, saved `mfrm_slope_intervals` objects named by quadrature count, with their numerical checks and cautions;
- `conditions`: warnings and messages from refitting and result extraction;
- `fits`: the reference and refitted `mfrm_fit` objects;
- `settings` and `notes`: the fixed comparison contract and interpretation boundary.

The object supports `print()`, `summary()`, `as.data.frame()`, and the existing `apa_table()` list route (for example `apa_table(out)`).

Two slope families

For a two-family GPCM, refits preserve the ordered slope owners, fixed N(0,1) population, score ladder, engine, integration method and stopping controls (`em_score_tol` for fixed-grid EM or `reltol` for adaptive direct MML). Each refit reconstructs the source initialization policy from the observed data. Fixed-grid EM uses the neutral vector; adaptive fits retain either neutral-only or neutral-plus-EM initialization. Older adaptive fits without this setting retain the earlier neutral-only procedure. The reference estimate is not used as a new starting vector. Component slopes retain `SlopeOwner` and `ScaleReference`: the first family has geometric mean one; the second is free on the fixed ability scale. Category-probability comparisons use the product of both slopes.

For both two-family engines, `intervals` contains the separately checked experimental `confint()` result for each grid, including failed checks, cautions and missing bounds. For example, after comparing `quad_points = c(31, 61)`, use `plot(out$intervals$q61)` and `apa_table(out$intervals$q61, which = "numerical_checks")`. These checks also evaluate q versus 2q-1 at that fit's saved parameters without another refit. Passing these checks does not establish sampling coverage or resolve other warnings about the fitted model. Adaptive fits use their moving-node objective for both the information and the higher-order comparison. Person-score comparisons in `summary` remain unavailable for two families: EAP and posterior-SD changes there are `NA`, not zero. With `adaptive_quad_points`, the separate `quadrature_review` table evaluates posterior moments and log marginal likelihoods at each unchanged calibration. These diagnostic integrals do not enable `predict_mfrm_units()` or supply Person intervals. The effective slope is the product of both facet slopes; moving the integration grid retains the original N(0,1) density and its Jacobian. This does not provide adaptive EM, update estimates or repair their intervals.

More quadrature points improve the numerical approximation, not the amount of observed information. In a sparse design, a small group of candidates rated many times may need a finer grid than candidates with few ratings. Inspect interval availability and measured changes; no fixed point count is guaranteed to be adequate for every design. See the GPCM scope vignette for a same-data example and its sampling limitations.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `apa_table()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("mml_quadrature_sensitivity", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", model = "RSM",
  quad_points = 31
)
sensitivity <- mml_quadrature_sensitivity(
  fit, toy, quad_points = c(31, 41), adaptive_quad_points = c(31, 61)
)
summary(sensitivity)
# Preserve small numerical differences when preparing the sensitivity table.
apa_table(sensitivity, digits = 5)

## End(Not run)
```

`normalize_conquest_overlap_exports` *Normalize the CSV exports generated by the ConQuest overlap template*

Description

Normalize the CSV exports generated by the ConQuest overlap template

Usage

```
normalize_conquest_overlap_exports(
  bundle,
  parameter_file,
  regression_file,
  covariance_file,
  case_file,
  delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  conquest_version = NULL,
  conquest_edition = NULL,
  run_date = NULL,
  report_file = NULL
)
```

Arguments

bundle	Output from build_conquest_overlap_bundle().
parameter_file	Path to the ConQuest export parameters CSV.
regression_file	Path to the ConQuest export reg_coefficients CSV.
covariance_file	Path to the ConQuest export covariance CSV.
case_file	Path to the ConQuest show cases EAP CSV.
delimiter	Delimiter used by all four files. The default, "auto", detects comma-, tab-, or semicolon-delimited files.
conquest_version	Optional ConQuest version recorded for provenance.
conquest_edition	Optional edition label, such as "demo/free", recorded for provenance.
run_date	Optional external-run date coercible to Date. Supply the date of the ConQuest run, not the normalization date.
report_file	Optional generated English show parameters review text. The estimation summary supplies stopping status and checks the person count, item model, covariate and ITEMS constraint. Missing reports leave external convergence unknown. This is not a general report parser.

Details

This helper is intentionally limited to the native CSV files requested by the command template from build_conquest_overlap_bundle(). It maps the exported regression rows to the bundle's intercept and covariate, maps the unidimensional covariance row to sigma2, trims padded person identifiers, and reconstructs the final item location under ConQuest's default sum-to-zero item constraint. It does not parse arbitrary ConQuest reports.

After normalization, pass the returned object directly to review_conquest_overlap(). Parameter and regressor indices must be unique integers in the generated model's coordinate range; row order may differ. Present dimension columns must identify dimension 1. Population variance must be finite and nonnegative; a zero boundary value is retained for descriptive review, not accepted as evidence of regular inference. Malformed exports are rejected before labels are assigned. This validation does not establish successful estimation. Present item labels must agree with the item map. IDs are read as text, including leading zeros, long integers and the literal ID NA. New bundle aliases are restored through person_map; unknown aliases are rejected. Keep all files from the same run: checks cannot establish that response cells or covariate values were unchanged. criterion_met describes the reported stopping rule, not integration accuracy or inferential readiness.

Value

An object of class mfrm_conquest_overlap_tables with normalized population, item, and case-EAP tables.

See Also

normalize_conquest_overlap_files(), normalize_conquest_overlap_tables(), review_conquest_overlap()

Examples

```
## Not run:
bundle <- build_conquest_overlap_bundle(output_dir = "conquest-overlap")
# Run bundle$conquest_command in ConQuest, then:
cq <- normalize_conquest_overlap_exports(
  bundle,
  "conquest-overlap/conquest_overlap_conquest_parameters.csv",
  "conquest-overlap/conquest_overlap_conquest_reg_coefficients.csv",
  "conquest-overlap/conquest_overlap_conquest_covariance.csv",
  "conquest-overlap/conquest_overlap_conquest_cases_eap.csv",
  report_file = "conquest-overlap/conquest_overlap_conquest_parameters_review.txt"
)
review_conquest_overlap(bundle, cq)

## End(Not run)
```

---

normalize_conquest_overlap_files

Normalize extracted ConQuest overlap files to the mfrmr *review contract*

Description

Normalize extracted ConQuest overlap files to the mfrmr review contract

Usage

```
normalize_conquest_overlap_files(
  population_file,
  item_file,
  case_file,
  population_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  item_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  case_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  conquest_population_term = "auto",
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
  conquest_item_estimate = "auto",
  conquest_case_person = "auto",
  conquest_case_estimate = "auto",
  keep_extra_columns = TRUE
)
```

Arguments

`population_file`
: Path to an extracted ConQuest population-parameter table in CSV/TSV/TXT form.

`item_file`
: Path to an extracted ConQuest item-estimate table in CSV/TSV/TXT form.

`case_file`
: Path to an extracted ConQuest case-level EAP table in CSV/TSV/TXT form.

`population_delimiter`
: Delimiter for `population_file`. `"auto"` chooses comma, tab, or semicolon from the file extension/header line.

`item_delimiter`
: Delimiter for `item_file`. `"auto"` chooses from the file extension/header line.

`case_delimiter`
: Delimiter for `case_file`. `"auto"` chooses from the file extension/header line.

`conquest_population_term`
: Column in `population_file` that stores parameter names. `"auto"` tries conservative aliases such as `Parameter` and `Term`.

`conquest_population_estimate`
: Column in `population_file` that stores parameter estimates. `"auto"` tries aliases such as `Estimate` and `Est`.

`conquest_item_id`
: Column in `item_file` that stores the item identifier as extracted by the user. `"auto"` tries aliases such as `ResponseVar`, `ItemID`, `Item`, and `Label`.

`conquest_item_estimate`
: Column in `item_file` that stores item estimates. `"auto"` tries `Estimate` and `Est`. `Facility` is not an automatic alias: establish whether it is a difficulty, easiness, or raw proportion and convert it to the required logit difficulty before review.

`conquest_case_person`
: Column in `case_file` that stores person IDs. `"auto"` tries conservative aliases such as `Person`, `PID`, and `Sequence ID`.

conquest_case_estimate
: Column in case_file that stores case EAP estimates. "auto" tries conservative aliases such as Estimate, EAP_1, and EAP.

keep_extra_columns
: If TRUE, keep all remaining columns after the standardized identifier and estimate columns.

Details

This helper is a thin file-wrapper around normalize_conquest_overlap_tables(). It is intentionally limited to already extracted tabular files and does not parse raw ConQuest report text.

The recommended workflow is:

1. export a bundle for the documented comparison scope with build_conquest_overlap_bundle();
2. extract the relevant ConQuest tables to CSV/TSV/TXT files;
3. call normalize_conquest_overlap_files() on those files;
4. pass the result to review_conquest_overlap().

Read summary(normalized)$normalization_scope before review to confirm that the files were treated as extracted tables, not raw ConQuest report text, and to check duplicate-ID / non-numeric-estimate pre-review flags.

Value

A named list with class mfrm_conquest_overlap_tables.

See Also

normalize_conquest_overlap_tables(), review_conquest_overlap()

Examples

```
# Reuse the synthetic fit; normalize the extracted tables below.
# Regeneration recipe: examples/measurement-fits.R in the package.
fit <- readRDS(system.file("examples", "measurement-fits.rds",
  package = "mfrmr", mustWork = TRUE))$conquest_quick
bundle <- build_conquest_overlap_bundle(fit = fit)
tmp_dir <- tempdir()
pop_path <- file.path(tmp_dir, "cq_pop.csv")
item_path <- file.path(tmp_dir, "cq_item.tsv")
case_path <- file.path(tmp_dir, "cq_case.csv")
utils::write.csv(
  data.frame(
    Term = bundle$mfrmr_population$Parameter,
    Est = bundle$mfrmr_population$Estimate
  ),
  pop_path,
  row.names = FALSE
)
utils::write.table(
```

```
  data.frame(
    Item = bundle$mfrmr_item_estimates$ResponseVar,
    Est = bundle$mfrmr_item_estimates$Estimate
  ),
  item_path,
  sep = "\t",
  row.names = FALSE
)
utils::write.csv(
  data.frame(
    PID = bundle$mfrmr_case_eap$Person,
    EAP = bundle$mfrmr_case_eap$Estimate
  ),
  case_path,
  row.names = FALSE
)
normalized <- normalize_conquest_overlap_files(
  population_file = pop_path,
  item_file = item_path,
  case_file = case_path,
  conquest_population_term = "Term",
  conquest_population_estimate = "Est",
  conquest_item_id = "Item",
  conquest_item_estimate = "Est",
  conquest_case_person = "PID",
  conquest_case_estimate = "EAP"
)
summary(normalized)$normalization_scope
review <- review_conquest_overlap(bundle, normalized)
summary(review)$summary
```

`normalize_conquest_overlap_tables` *Normalize extracted ConQuest overlap tables to the* `mfrmr` *review contract*

Description

Normalize extracted ConQuest overlap tables to the `mfrmr` review contract

Usage

```
normalize_conquest_overlap_tables(
  conquest_population,
  conquest_item_estimates,
  conquest_case_eap,
  conquest_population_term = "auto",
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
```

```
  conquest_item_estimate = "auto",
  conquest_case_person = "auto",
  conquest_case_estimate = "auto",
  keep_extra_columns = TRUE
)
```

Arguments

`conquest_population`
: Extracted ConQuest population-parameter table as a data.frame.

`conquest_item_estimates`
: Extracted ConQuest item-estimate table as a data.frame.

`conquest_case_eap`
: Extracted ConQuest case-level EAP table as a data.frame.

`conquest_population_term`
: Column in `conquest_population` that stores parameter names. `"auto"` tries conservative aliases such as `Parameter` and `Term`.

`conquest_population_estimate`
: Column in `conquest_population` that stores parameter estimates. `"auto"` tries aliases such as `Estimate` and `Est`.

`conquest_item_id`
: Column in `conquest_item_estimates` that stores the item identifier as exported or extracted by the user. `"auto"` tries aliases such as `ResponseVar`, `ItemID`, `Item`, and `Label`.

`conquest_item_estimate`
: Column in `conquest_item_estimates` that stores item estimates. `"auto"` tries `Estimate` and `Est`. Other columns must be specified explicitly and converted to difficulty logits before review.

`conquest_case_person`
: Column in `conquest_case_eap` that stores person IDs. `"auto"` tries conservative aliases such as `Person`, `PID`, and `Sequence ID`.

`conquest_case_estimate`
: Column in `conquest_case_eap` that stores case EAP estimates. `"auto"` tries conservative aliases such as `Estimate`, `EAP_1`, and `EAP`.

`keep_extra_columns`
: If `TRUE`, keep all remaining columns after the standardized identifier and estimate columns.

Details

This helper does not parse raw ConQuest text output. It standardizes already extracted tables to the contract used by `review_conquest_overlap()`:

- population parameters become columns `Parameter`, `Estimate`, and `EstimateNonNumeric`;
- item estimates become columns `ItemID`, `Estimate`, and `EstimateNonNumeric`;
- case summaries become columns `Person`, `Estimate`, and `EstimateNonNumeric`.

The resulting object is intentionally conservative. It does not infer whether item IDs correspond to exported response variables or original item levels; that matching step remains part of `review_conquest_overlap()`, where the standardized ConQuest tables are compared against a concrete overlap bundle.

Value

A named list with class `mfrm_conquest_overlap_tables`.

Output

The returned object has class `mfrm_conquest_overlap_tables` and includes:

- `summary`: one-row normalization summary
- `conquest_population`: standardized population table
- `conquest_item_estimates`: standardized item table
- `conquest_case_eap`: standardized case table
- `settings`: source-column metadata
- `notes`: interpretation notes

Read `summary(normalized)$normalization_scope` before review to confirm that the object contains extracted tabular inputs, not parsed raw ConQuest report text, and to check duplicate-ID / non-numeric-estimate pre-review flags.

See Also

`build_conquest_overlap_bundle()`, `review_conquest_overlap()`

Examples

```
normalized <- normalize_conquest_overlap_tables(
  conquest_population = data.frame(
    Term = c("(Intercept)", "GroupB", "sigma2"),
    Est = c(0, 0.2, 1)
  ),
  conquest_item_estimates = data.frame(
    Item = c("I1", "I2"),
    Est = c(-0.2, 0.2)
  ),
  conquest_case_eap = data.frame(
    PID = c("P001", "P002"),
    EAP = c(-0.1, 0.1)
  ),
  conquest_population_term = "Term",
  conquest_population_estimate = "Est",
  conquest_item_id = "Item",
  conquest_item_estimate = "Est",
  conquest_case_person = "PID",
  conquest_case_estimate = "EAP"
)
summary(normalized)$normalization_scope
```

`ordinal_logistic_table` *Compare ordered responses using cumulative logistic models*

Description

Compare two groups at an explicitly supplied numeric matching variable, using three nested proportional-odds models. This independent-Person sensitivity analysis does not estimate ability or fit an MFRM.

Usage

```
ordinal_logistic_table(
  data,
  person,
  target,
  score,
  group,
  matching,
  groups,
  categories,
  p_adjust = "holm"
)
```

Arguments

`data`	Long-format data with at most one row per Person and target.
`person`	Name of the independent Person identifier column.
`target`	Name of the item or other target identifier column.
`score`	Name of the ordered response column. Numeric, character or factor scores are accepted, with order declared through `categories`.
`group`	Name of the group column. Each Person belongs to one group.
`matching`	Name of one finite numeric matching column. Values may differ across targets. Missing scores or matching values are excluded together from all models for the affected target.
`groups`	Two distinct labels in reference, focal order.
`categories`	Distinct numeric or character category labels, in increasing response order. Factor labels are treated as character. Category spacing has no effect. Every declared category must occur in each target's complete data; otherwise that target is unavailable. Categories are never collapsed.
`p_adjust`	A `stats::p.adjust()` method, default `"holm"`. The DIF comparison family comprises all three tests for every observed target, including unavailable tests. Proportional-odds diagnostics form a separate family of one test per target.

Details

Requires the optional package **ordinal**. Maximum likelihood is computed by `ordinal::clm()` with a logit link and flexible thresholds. For each target, matching is centered at its pooled midrange and divided by its half-range: `z = (matching - Center) / Scale`. Group is coded 0 for reference and 1 for focal. `M1` uses `z`, `M2` uses `z + g`, and `M3` uses `z * g`. All three use identical complete responses. The model is `logit Pr(Y <= k) = threshold[k] - linear_predictor`.

`Uniform` compares M2 to M1 (1 df), `Nonuniform` compares M3 to M2 (1 df), and `Joint` compares M3 to M1 (2 df). These are asymptotic likelihood-ratio tests. A uniform comparison assumes no interaction; its coefficient must not be interpreted as a constant group difference when an interaction matters. No automatic selection between these models or DIF/bias flag is made. `NagelkerkeR2` is `(1-exp(2*(LL0-LL)/N))/(1-exp(2*LL0/N))`, where LL0 is the intercept-only multinomial log likelihood. `DeltaNagelkerkeR2` subtracts the reduced model's value from the full model's value. This is not explained score variance or McKelvey–Zavoina R-squared; no universal effect-size cut-off is applied.

`contrasts` gives M3's reference/focal upper-tail odds ratio at the lower endpoint, midpoint and upper endpoint of the groups' overlapping matching ranges. Its log is `-(beta_g + beta_z:g * z)`; a value above 1 favors higher reference scores. This applies to each category cut under proportional odds. These are point estimates, not simultaneous intervals. Fits use all complete responses, including those outside that overlap. Range overlap alone does not establish comparable distributions or adequate local support. Each group must have varying matching values and their ranges must overlap with positive width. The `matching` table retains the ranges and transformation.

For three or more categories, `proportional_odds` compares M3 with a model in which all three slopes may differ across cumulative category cuts (`clm(y ~ 1, nominal = ~ z * g)`). This global asymptotic LR diagnostic has `3 * (K - 2)` df. It concerns equality across category cuts, whereas the nonuniform comparison concerns differences across matching values. They are different questions. The diagnostic is not applicable to binary responses. An unavailable or nonsignificant diagnostic does not certify proportional odds. Its p-value does not determine availability or change the primary comparisons.

`models` retains convergence codes, warnings, gradient, Hessian and Newton step diagnostics. A fit is unavailable if the backend reports nonconvergence or nonunique parameters, any coefficient is aliased/nonfinite, the Hessian's minimum eigenvalue is at most `sqrt(.Machine$double.eps) * max(1, max_eigen)`, the maximum absolute gradient exceeds `1e-6`, or the maximum absolute Newton step exceeds `1e-4` on the scaled parameterization. Backend warnings also make the fit unavailable. These conservative numerical checks detect common separation failures; they are not a general proof of finite ML existence. Likelihood decreases beyond `64 * .Machine$double.eps * max(1, abs(LL))` invalidate a comparison; smaller rounding errors are set to zero. No penalized estimator, collapsed categories or alternative inference is substituted.

Inference requires independent Persons within each target, a suitable linear matching relationship, finite identifiable parameters and adequate data for chi-squared approximations. Sparse categories and small samples can invalidate those approximations even when numerical checks pass. Duplicate Person-target rows, including missing-score duplicates, are rejected before exclusions. Renaming repeated ratings as Persons does not establish independence. Dependence across targets is compatible with Holm adjustment of valid individual p-values, but no dependence within targets is handled here.

Interpreting conditional group association as DIF additionally requires defensible matching, invariant anchors and comparable rater/task exposure. Use a justified common rating plan; this interface

does not adjust for plan differences, matching error, clustered ratings or informative missingness. Total and rest scores need not define the same comparison. RSM/PCM's adjacent category logits do not generally imply cumulative proportional odds. Therefore agreement with `analyze_dff()` or `mantel_haenszel_table()` does not by itself establish robustness, and disagreement does not by itself establish DIF.

Value

A plain list containing `table` (three comparisons per target), `models` (three PO fits and, for ordinal responses, a diagnostic fit), `coefficients` (available models only), `contrasts`, `proportional_odds`, `matching`, `cases`, `settings` and `notes`. `cases$Included` denotes complete rows supplied to eligible models, not successful estimation or valid DIF inference. Precluded targets retain their counts and reasons. Save the full result with `base::saveRDS()` and individual tables with `utils::write.csv()`. No backend model objects or executable environments are retained.

References

Rogers, H. J., & Swaminathan, H. (1993). A comparison of logistic regression and Mantel-Haenszel procedures for detecting differential item functioning. *Applied Psychological Measurement, 17*, 105–116. doi:10.1177/014662169301700201.

Zumbo, B. D. (1999). *A handbook on the theory and methods of differential item functioning (DIF): Logistic regression modeling as a unitary framework for binary and Likert-type (ordinal) item scores*. Department of National Defense, Ottawa, pp. 23–32. Pseudo-R-squared definitions and thresholds in that handbook are not interchangeable.

See Also

`analyze_dff()`, `mantel_haenszel_table()`, mfrmr_linking_and_dff

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("ordinal_logistic_table", run.dontrun = TRUE)
## Not run:
if (requireNamespace("ordinal", quietly = TRUE)) {
  d <- expand.grid(Match = c(-1, 0, 1), Group = c("R", "F"),
                   Score = 0:2, Replicate = 1:5)
  d$Person <- paste0("P", seq_len(nrow(d)))
  d$Item <- "A"
  ol <- ordinal_logistic_table(d, "Person", "Item", "Score", "Group",
                               "Match", c("R", "F"), 0:2)
  ol$table
  ol$proportional_odds
}

## End(Not run)
```

`pairwise_icc_table` *Compute observed-score ICCs for each rater pair*

Description

Compare each pair on its common, complete units using a two-way ANOVA ICC. No MFRM fit, mixed-model estimation or extra package is required.

Usage

```
pairwise_icc_table(
  data,
  rater,
  unit,
  score,
  model = c("mixed", "random"),
  definition = c("agreement", "consistency"),
  type = c("single", "average")
)
```

Arguments

`data`	Long-format data with at most one row per rater and unit.
`rater`	Name of the rater identifier column.
`unit`	Names of columns jointly identifying a rated unit, for example `c("Person", "Task", "Criterion")`. These columns match observations; they are not separately estimated variance components.
`score`	Name of a numeric score column. Scores must share a meaningful interval scale. Numeric ordinal category codes do not establish that assumption. Factors/characters are not silently converted to scores.
`model`	`"mixed"` (default) describes the selected raters. `"random"` assumes raters were sampled from a population to which results may generalize. Two-way point formulas are identical; their interpretation differs. Choosing `"random"` does not establish random sampling.
`definition`	`"agreement"` (default) includes systematic rater offsets; `"consistency"` removes their contribution.
`type`	`"single"` (default) concerns one rating; `"average"` concerns the unweighted mean of the two ratings, not a mean over the full panel.

Details

All observed rater identifiers are paired. Each pair uses only units with two nonmissing scores. Other raters' missing scores do not exclude a unit for that pair. Missing-score rows and absent assignments are distinguished in `cases`. Duplicate rater-unit cells (including missing-score duplicates) are rejected rather than averaged. Identifiers must be nonmissing and nonblank. There is no

imputation or observation weighting. Pairs with fewer than two complete units remain in `table` with an unavailable ICC.

Let R, C and E be the unit, rater and residual ANOVA mean squares on a complete n-by-2 matrix. Following Koo and Li (2016, Table 3), the numerator is R-E. The single-rating denominator is R+E for consistency, or R+E+2(C-E)/n for absolute agreement. The average-rating denominator is R for consistency, or R+(C-E)/n for agreement. `Form` records ICC(C,1), ICC(A,1), ICC(C,2) or ICC(A,2); interpret these together with `Model`. ICC(A,2) denotes an average of two ratings, not Shrout–Fleiss ICC(2,1). One-way ICCs are not computed.

Negative estimates are retained. A nonpositive denominator makes `ICC` unavailable and sets `Status = "nonpositive_denominator"`; `RawICC` retains the algebraic ratio when finite so it cannot masquerade as a valid reliability estimate. An entirely constant matrix is undefined, not 1. Internally scores are shifted and scaled to avoid overflow/underflow. `anova` records scaled mean squares, the offset/scale, numerator and denominator, so the calculation can be reconstructed. These scaled mean squares are not variances in the original score units.

Results describe each pair's retained unit mix. Different pairs can have different units and score ranges; their ICCs cannot automatically rank rater quality and should not be averaged into a panel ICC. Restricted unit variability can lower ICC without larger absolute disagreement. Inference requires a suitable sampling design and assumptions about errors and dependence. Treating several tasks from one person as separate units does not make them independent. No confidence intervals, hypothesis tests, quality bands or rater-training decisions are supplied. This is distinct from the component variance shares in `compute_facet_icc()`, MFRM separation reliability and the fitted expected agreement in `interrater_agreement_table()`.

Value

A plain list containing `table` (one row per rater pair), `anova` (scaled ANOVA ingredients), `cases` (one row per pair and union of assigned units, with input-row references and inclusion flags), `units` (the unique input unit keys in input-column order), `settings` and `notes`. `Unit` in `cases` is a row index into `units`. Units assigned to neither member of a pair are outside that pair's audit. `Missing1`/`Missing2` indicate explicit missing-score rows; `Assigned1`/`Assigned2` also distinguish absent assignments. Save with `base::saveRDS()` and export data frames with `utils::write.csv()`. Reading saved output performs no calculation.

References

Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine, 15*(2), 155-163. doi:10.1016/j.jcm.2016.02.012. Table 3; see also the 2017 erratum, doi:10.1016/j.jcm.2017.10.001, correcting the one-way ICC(1,1) formula.

See Also

`interrater_agreement_table()`, `compute_facet_icc()`, `krippendorff_alpha_table()`, `score_comparison_table()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("pairwise_icc_table", run.dontrun = TRUE)
## Not run:
```

```
data("mfrmr_example_agreement", package = "mfrmr")
ratings <- mfrmr_example_agreement
absolute <- pairwise_icc_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", definition = "agreement")
consistent <- pairwise_icc_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", definition = "consistency")
absolute$table
consistent$table
# R1/R2 have perfect consistency, but R2's one-category offset reduces agreement.
# These ICCs treat 0--3 as numeric scores; justify equal score intervals.

## End(Not run)
```

`pairwise_kappa_table` *Compute pairwise weighted Cohen's kappa*

Description

Compare observed scores on matched units for every pair of raters, using quadratic (QWK), linear or unweighted agreement. No model is fitted.

Usage

```
pairwise_kappa_table(
  data,
  rater,
  unit,
  score,
  categories,
  weights = c("quadratic", "linear", "unweighted")
)
```

Arguments

`data`	Long-format ratings with at most one row per rater and unit.
`rater`	Name of the rater identifier column.
`unit`	Names of columns jointly identifying a rated unit, such as `c("Person", "Task")`.
`score`	Name of the score column. Values may be numeric, character, factor or logical, with `NA` for missing scores. Factors use their labels.
`categories`	Required distinct permitted category values, including unused categories, in their substantive order. At least two are required. They must have the same numeric/character/logical representation as scores. With linear/quadratic weights, numeric values use their actual spacing; character/logical values use equally spaced positions in this vector. Factor levels never silently determine the order. To treat numeric codes as equally spaced ranks, explicitly convert scores and categories to character. Choose spacing on substantive grounds.
`weights`	One of `"quadratic"` (default), `"linear"` or `"unweighted"`. Agreement weights are 1 minus squared normalized distance, 1 minus normalized absolute distance, or the identity matrix, respectively.

Details

For each pair, both observed and expected agreement use only common complete units. Expected cell proportions are products of the two rater-specific margins on this same sample. Kappa is `(observed - expected)/(1 - expected)`. This complete-case convention differs from the all-available rater margins in Gwet's incomplete-data Example 4.1. Different pairs may retain different cohorts; comparing or averaging their kappas does not establish a panel coefficient. At least two complete units are required, as in the existing QWK score-comparison interface.

Computation uses the equivalent disagreement ratio, with distances rescaled to the categories actually present in each retained pair. The common scale cancels, avoiding loss of the coefficient when an unused numeric category is very distant. `ScaledObservedDisagreement` and `ScaledExpectedDisagreement` retain those ingredients; agreement columns use the declared full-scale weights. Agreement columns can round to 1 at extreme relative spacings even when kappa is defined. Negative estimates remain visible; zero expected disagreement returns `NA` with a status.

Duplicate rater-unit cells are rejected before dropping missing scores. The case audit distinguishes an absent assignment from an explicit missing score and retains the union of units assigned to either member of the pair. No imputation, confidence intervals, tests or quality cutoffs are supplied. Missingness may affect representativeness. Agreement does not establish reference accuracy or MFRM rater-severity homogeneity. Storage of the weight matrix requires space quadratic in the number of declared categories; these are categorical-score interfaces, not continuous-score ICCs.

Value

A plain list with `table` (one row per pair), `counts` (nonzero retained contingency cells, with category indices), `categories` (indices, values and weighting positions), `weights` (agreement matrix), `cases`, `units`, `settings` and `notes`. `Unit` indexes `units`. Save with `base::saveRDS()` or export data frames with `utils::write.csv()`.

References

Cohen, J. (1968). Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70, 213-220. doi:10.1037/h0026256.

Gwet, K. L. *Handbook of Inter-Rater Reliability*, fifth edition, volume 1, chapter 4, pp. 100-110, equations 4.2.4-4.2.6. https://www.agreestat.com/assets/cac5-chapter4-excerpt.pdf

See Also

`gwet_ac2_table()`, `krippendorff_alpha_table()`, `pairwise_icc_table()`, `score_comparison_table()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("pairwise_kappa_table", run.dontrun = TRUE)
## Not run:
data("mfrmr_example_agreement", package = "mfrmr")
ratings <- mfrmr_example_agreement
head(ratings) # One rater's score for one Person-Task response.
pairs <- pairwise_kappa_table(ratings, rater = "Rater",
  unit = c("Person", "Task"), score = "Score", categories = 0:3)
pairs$table
```

```
pairs$cases # Included marks the shared nonmissing ratings for each pair.
# R2 gives every response one category more than R1: equal rankings do not
# imply agreement. Do not treat two Tasks by the same Person as one response.

## End(Not run)
```

`plot.mfrm_bundle` *Plot report/table bundles with base R defaults*

Description

Plot report/table bundles with base R defaults

Usage

```
## S3 method for class 'mfrm_bundle'
plot(x, y = NULL, type = NULL, ...)
```

Arguments

`x`	A bundle object returned by mfrmr table/report helpers.
`y`	Reserved for generic compatibility.
`type`	Optional plot type. Available values depend on bundle class.
`...`	Additional arguments forwarded to class-specific plotters.

Details

`plot()` dispatches by bundle class:

- `mfrm_unexpected` -> plot(x, type = "unexpected")
- `mfrm_fair_average` -> plot(x, type = "fair_average")
- `mfrm_displacement` -> plot(x, type = "displacement")
- `mfrm_interrater` -> plot(x, type = "agreement")
- `mfrm_facets_chisq` -> plot(x, type = "facets_chisq")
- `mfrm_bias_interaction` -> plot(x, type = "bias")
- `mfrm_bias_count` -> bias-count plots (cell counts / low-count rates)
- `mfrm_fixed_reports` -> pairwise-contrast diagnostics
- `mfrm_visual_summaries` -> warning/summary message count plots
- `mfrm_category_structure` -> default base-R category plots
- `mfrm_category_curves` -> overview (default), ogive, CCC / category probability / conditional probability, cumulative, total-information, and category-specific-information plots
- `mfrm_rating_scale` -> category-counts/threshold plots
- `mfrm_measurable` -> measurable-data coverage/count plots

- mfrm_unexpected_after_bias -> post-bias unexpected-response plots
- mfrm_output_bundle -> graph/score output-file diagnostics, including type = "score_se" when scorefile SE columns are available
- mfrm_residual_pca -> residual PCA scree, parallel-analysis, or loadings views via plot(x, type = "residual_pca")
- mfrm_specifications -> facet/anchor/convergence plots
- mfrm_data_quality -> dashboard, quality-flag, score-map, facet-pattern, and row/category/missing-row plots
- mfrm_facets_fit_review -> FACETS-style df-sensitivity plot
- mfrm_fit_measures -> fit-status counts, Infit/Outfit scatter, measure normal bands, and FACETS-style df-sensitivity plots. For type = "measure_ci", the caption retains the source interval interpretation (including exploratory JML bands); fixed values use open diamonds and finite estimates without intervals use crosses. main = "" omits the title and show_notes = FALSE hides the caption without removing it from saved data. Changing ci_level does not change inferential eligibility. See fit_measures_table().
- mfrm_iteration_report -> replayed-iteration trajectories
- mfrm_subset_connectivity -> subset-observation/connectivity plots
- mfrm_facet_statistics -> facet statistic profile plots
- mfrm_export_bundle / mfrm_summary_appendix_export -> export handoff plots (formats, artifact_groups, selection_tables, selection_handoff, selection_handoff_bundles, selection_handoff_roles, selection_handoff_role_sections, selection_bundles, selection_roles, selection_sections)

If a class is outside these families, use dedicated plotting helpers or custom base R graphics on component tables.

For mfrm_category_curves, pass preset = "monochrome" for grayscale/line-type output. Cumulative .5 boundary lines are shown only for interpretable in-range boundaries by default; use boundary_status = "all" to show every finite boundary estimate or boundary_status = "none" / show_cumulative_boundaries = FALSE to suppress those vertical boundary lines. Use plot_data(x, component = "plot_long") on a category-curve bundle when you want one ggplot2/plotly-friendly table across all curve families. Keep curve families and groups separate in custom figures. Automatic as_ggplot() conversion supports rating-scale counts/thresholds and category-structure counts; it is unavailable for category-curve bundles and category-structure boundary views. Step displays use point estimates and do not turn stored step SEs into crossing, gap or curve intervals.

Value

A plotting-data object of class mfrm_plot_data.

Interpreting output

The returned object is plotting data (mfrm_plot_data) that captures the selected route and reusable data; set draw = TRUE for immediate base graphics.

Typical workflow

1. Create bundle output (e.g., `unexpected_response_table()`).
2. Inspect routing with `summary(bundle)` if needed.
3. Call `plot(bundle, type = ..., draw = FALSE)` to obtain reusable plot data.

Network display controls

For subset, design, rater and halo bundles with `type = "network"`, use `layout` (a two-column coordinate matrix, optionally row-named by all node IDs, or a function of the displayed igraph), `display_min_weight` (absolute edge weight for the picture only), and `graphical` (a named list of plot.igraph arguments, such as vertex.size, vertex.label, vertex.color, edge.curved). Unnamed vectors and coordinates follow the returned `nodes` order. All nodes, including isolates, remain. Edge vectors in `graphical` follow the displayed `edges` order, after any display filter. The default is a deterministic circle; positions do not estimate anything. Labels are hidden by default above 80 nodes; set vertex.label explicitly to override. Supply the same named layout to compare views. Constructor filters such as `min_weight` change calculated metrics; `display_min_weight` does not. Returned `source_edges` retain the calculated graph, `edges` the displayed graph, and `coordinates` the actual positions. Titles are metadata by default; add a title explicitly via `graphical`. See `vignette("mfrmr-networks-and-reporting")` for bipartite assignment, directional disagreement and rater-by-criterion examples inspired by Lamprianou. With `rater_network_analysis()` and `weight_metric = "DirectionalMagnitude"`, `type = "fit"`, `metric = "SWS"` (or `"PUS"`) shows the published network inconsistency index. The returned table retains unavailable rows/statuses; there are no calibrated reference cutoffs. An entirely unavailable index draws an unavailable message and retains the original rows/statuses in the returned table, rather than substituting zero. For signed alpha, `type = "matrix"` returns all eligible signed pair coefficients before graph filtering, retaining zero and undefined values separately. Network drawings use magnitude for edge width and dashed lines for negative agreement. Use `plot.mfrm_network_ergm()` for ERGM coefficient intervals and simulated GOF.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`summary()`, plot(x, type = "unexpected"), plot(x, type = "fair_average"), plot(x, type = "displacement")

Examples

```
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_bundle", run.dontrun = TRUE)
## Not run:
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
t4 <- unexpected_response_table(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 5)
p <- plot(t4, draw = FALSE)
vis <- build_visual_summaries(fit, diagnose_mfrm(fit, residual_pca = "none"))
p_vis <- plot(vis, type = "comparison", draw = FALSE)
spec <- specifications_report(fit)
p_spec <- plot(spec, type = "facet_elements", draw = FALSE)
if (interactive()) {
  plot(
    t4,
    type = "severity",
    draw = TRUE,
    main = "Unexpected Response Severity (Customized)",
    palette = c(higher = "#d95f02", lower = "#1b9e77", bar = "#2b8cbe"),
    label_angle = 45
  )
  plot(
    vis,
    type = "comparison",
    draw = TRUE,
    main = "Warning vs Summary Counts (Customized)",
    palette = c(warning = "#cb181d", summary = "#3182bd"),
    label_angle = 45
  )
}

## End(Not run)
```

`plot.mfrm_clusters` *Plot exploratory groups and imputation sensitivity*

Description

Inspect within-sample separation, individual feature profiles, or pairwise co-membership across imputations using existing clustering results.

Usage

```
## S3 method for class 'mfrm_clusters'
plot(
  x,
  type = c("silhouette", "profile"),
  feature = NULL,
  labels = NULL,
  draw = TRUE,
  preset = "standard",
  ...
)

## S3 method for class 'mfrm_imputed_clusters'
plot(x, ids = NULL, labels = NULL, draw = TRUE, preset = "standard", ...)
```

Arguments

x	An object returned by mfrm_cluster_pam(), mfrm_cluster_kmeans() or mfrm_cluster_imputed(), as appropriate.
type	For a single partition, "silhouette" or "profile".
feature	A single selected feature name, required for type = "profile". Numeric features show means and medians in their original units; categorical features show within-group proportions in the original level order (including unused factor levels).
labels	Whether to display entity IDs on silhouettes or heatmaps. The default displays them for at most 50 entities. No entities are sampled when labels are hidden. Profile plots always label groups and levels.
draw	Draw the plot when TRUE; FALSE only returns plotted values.
preset	Plot style: "standard", "publication", "compact", or "monochrome".
...	Reserved for future use; additional arguments are rejected.
ids	For imputation heatmaps, an optional character vector of distinct entity IDs in the desired display order. Selection affects only the view, not clustering or the denominator. By default all IDs appear in input order.

Details

No model or clustering is refitted. Silhouette widths describe separation in the fitted sample, not stability or probabilities. The dashed line is the overall mean silhouette; excluded entities have no silhouette.

Feature profiles describe one partition; numeric summaries have no confidence intervals. For an imputed result, inspect a completion with plot(x$analyses[[1]], type = "profile", feature = "ExperienceYears"). Cluster numbers must not be averaged across imputations.

Imputation heatmaps show the fraction of all supplied imputations in which each pair belongs to the same group, on a fixed zero-to-one scale. Grey cells are unavailable pairs involving excluded entities, not zero co-membership. These fractions describe sensitivity to imputations under fixed settings, not membership probabilities, sampling stability, or a consensus partition. Rows and columns follow input order or explicit `ids`; no hierarchical clustering is performed. PAM is nonhierarchical and these plots do not provide a dendrogram. Use `mfrm_cluster_hierarchical()` for a separate hierarchical analysis and its dendrogram.

Value

Invisibly, an `mfrm_plot_data` object. Its `data` contains the plotted table or matrix, title, subtitle, legend, and excluded IDs. Heatmaps also retain the displayed IDs and the number of imputations. Use `plot_data()` to extract this payload for custom graphics. Profile plots also retain `observations`: original-unit entity values and group/exclusion status, suitable for custom distribution plots; see `plot_data()`. `as_ggplot()` converts imputation co-membership heatmaps using the saved matrix, ID order, label choice, colours and imputation count. The default and `component = "matrix"` retain the complete view and its fixed zero-to-one scale. Unavailable cells have both grey fill and crosses, distinguishing them from zero even in monochrome. Metadata, including excluded IDs, remain available with `plot_data()`. No values are recomputed or renormalized when IDs are selected. Use `ggplot2::labs(title = NULL, subtitle = NULL, caption = NULL)` to hide annotations while retaining the source data. Silhouette conversion retains negative widths, saved order and the overall mean reference, with group labels independent of colour. Numeric profiles retain original-unit means/medians and counts; circles and triangles are offset vertically to show coincident values without adding intervals. Categorical profiles retain the original category order, unused levels, group counts and a fixed zero-to-one proportion scale. Default and `component = "table"` preserve the complete selected view; categorical profiles also accept `component = "matrix"`. These conversions do not recluster, select groups, or estimate uncertainty.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset

changes appearance, not estimates, confidence levels or diagnostic thresholds.

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`mfrm_cluster_pam()`, `mfrm_cluster_imputed()`, `mfrm_cluster_compare()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_clusters", run.dontrun = TRUE)
## Not run:
if (requireNamespace("cluster", quietly = TRUE)) {
  raters <- data.frame(Rater = paste0("R", 1:6),
    ExperienceYears = c(1, 2, 3, 12, 13, 14),
    Specialty = factor(rep(c("Language", "Science"), each = 3)))
  groups <- mfrm_cluster_pam(mfrm_features(raters, "Rater",
    c("ExperienceYears", "Specialty")), k = 2)
  plot(groups)
  plot(groups, type = "profile", feature = "ExperienceYears")
  plot(groups, type = "profile", feature = "Specialty")
  values <- plot_data(plot(groups, draw = FALSE))
  values$table
}

## End(Not run)
```

plot.mfrm_design_evaluation *Plot a design-simulation study*

Description

Plot a design-simulation study

Usage

```
## S3 method for class 'mfrm_design_evaluation'
plot(
  x,
  facet = c("Rater", "Criterion", "Person"),
  metric = c("separation", "reliability", "infit", "outfit", "misfitrate",
   "severityrmse", "severitybias", "convergencerate", "elapsedsec", "mincategorycount",
   "designdensity", "plannedmissingrate", "linkpersons", "linkfraction", "linkraters",
    "mincommonpersons", "zerocommonpairs", "pairsshorttarget"),
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_design().
facet	Facet to visualize.
metric	Metric to plot.
x_var	Design variable used on the x-axis. When x was generated from a sim_spec with custom public facet names, the corresponding aliases (for example n_judge, n_task, judge_per_person) are also accepted. Role keywords (person, rater, criterion, assignment) are accepted as an abstraction over the current two-facet schema.
group_var	Optional design variable used for separate lines. The same alias rules as x_var apply.
draw	If TRUE, draw with base graphics; otherwise return plotting data.
...	Reserved for generic compatibility.

Details

This method is designed for quick design-planning scans rather than polished publication graphics.

Useful first plots are:

- rater metric = "separation" against x_var = "n_person"

- criterion `metric = "severityrmse"` against `x_var = "n_person"` when you want aligned recovery error rather than raw location shifts
- rater `metric = "convergencerate"` against `x_var = "raters_per_person"`
- sparse linked `metric = "plannedmissingrate"`, `"mincommonpersons"`, `"zerocommonpairs"`, or `"pairsshorttarget"` to review planned missingness and rater-pair linkage separately from recovery metrics

Value

If `draw = TRUE`, invisibly returns a plotting-data list. If `draw = FALSE`, returns that list directly. The returned list includes resolved canonical variables (`x_var`, `group_var`) together with public labels (`x_label`, `group_label`), `design_variable_aliases`, and `design_descriptor`, plus `planning_scope`, `planning_constraints`, and `planning_schema`.

See Also

`evaluate_mfrm_design()`, summary.mfrm_design_evaluation

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_design_evaluation", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-design.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
sim_eval <- saved$result
p <- plot(sim_eval, facet = "Rater", metric = "separation", x_var = "n_person", draw = FALSE)
c(p$facet, p$x_var)

## End(Not run)
```

---

plot.mfrm_diagnostic_screening

Plot a diagnostic-screening simulation study

Description

Builds an integrated visual summary from `evaluate_mfrm_diagnostic_screening()` output. The default view combines legacy residual, strict marginal, strict pairwise, strict combined, and optional report-index review rates so simulation results can be inspected in one operating-characteristic surface.

Usage

```
## S3 method for class 'mfrm_diagnostic_screening'
plot(
  x,
  type = c("overview", "report", "contrast", "runtime"),
  metric = NULL,
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_diagnostic_screening().
type	Plot family. "overview" combines screening and optional report rates or counts. "report" focuses on mfrm_report() review signals. "contrast" plots misspecification-minus-well-specified contrasts. "runtime" plots elapsed-time summaries.
metric	Metric family. Use NULL or "auto" for the default within each type. Supported values are documented by error messages and include "rate", "count", "magnitude", "elapsed", and "per_observation" depending on type.
x_var	Design variable for the horizontal axis. Public design aliases from a simulation specification are accepted.
group_var	Optional additional design variable to include in group labels. Public design aliases are accepted.
draw	Logical; if FALSE, return the plot-data bundle without drawing.
...	Reserved for generic compatibility.

Value

An mfrm_plot_data object with reusable metadata, a long-form plot_long table, and interpretation handoff tables (overview, reading_order, next_actions, reporting_notes, and figure_recipes). When draw = TRUE, the object is returned invisibly after drawing.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_diagnostic_screening", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-diagnostic-screening.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
diag_eval <- saved$result
plot(diag_eval, type = "overview", draw = FALSE)
plot_data(diag_eval, type = "overview", component = "plot_long")
plot_data(diag_eval, type = "overview", component = "next_actions")
```

```
plot_data(diag_eval, type = "overview", component = "figure_recipes")
plot(diag_eval, type = "report", metric = "rate", draw = FALSE)

## End(Not run)
```

`plot.mfrm_dif` *Plot differential facet functioning*

Description

Compare saved group contrasts as ordered bars or a heatmap. Read the analysis method and scale before interpreting the size of a difference.

Usage

```
## S3 method for class 'mfrm_dif'
plot(x, type = c("summary", "heatmap"), ...)
```

Arguments

`x`	A result from `analyze_dff()` or `analyze_dif()`.
`type`	`"summary"` for ordered effects or `"heatmap"` for the interaction grid.
`...`	Named controls described below. Both views accept `draw = FALSE` to return plot data without drawing. Controls for the other view are rejected.

Value

An `mfrm_plot_data` object. The summary's `data$data` holds the displayed contrasts; the heatmap's `data$matrix` holds the plotted matrix. Both retain source rows in `data$pairs`, response matches (when available) in `data$cases`, and source settings in `data$analysis_config`. When supplied to `analyze_dff()`, the separate prospective interaction review is retained in `data$comparison_review`. Its states do not classify the displayed residual/refit effects; inspect its own model and weights. Retain the returned notes and unavailable values when exporting; a plot does not qualify new inference.

Summary controls

- `top_n = 30` limits displayed contrasts; `sort_by` is `"abs_effect"` (default), `"effect"`, or `"classification"`.
- `ci_level = NULL` omits intervals. A level such as `0.95` requests approximate normal intervals from available linked-refit standard errors; residual comparisons require `NULL`.
- `effect_thresholds` adds symmetric effect-size guide lines, not classification boundaries. `effect_axis_label` overrides the axis label.
- `preset` is `"standard"`, `"publication"`, `"compact"`, or `"monochrome"`.

Heatmap controls

- `metric = "obs_exp"` shows cell mean observed-minus-expected scores; `"t"` shows a descriptive scaled residual, not a t-test. These require the cell table from `method = "residual"`. `"contrast"` shows the saved pairwise group contrasts, on the analysis method's scale.

Heatmap appearance

- `show_values = TRUE` labels cells; `value_digits = 2` controls rounding.
- `flag_threshold = NULL` adds no flags. A non-negative value outlines cells at or above that absolute value; `flag_color = "black"` sets the border.
- `scale_limit = NULL` uses the observed range. A positive value fixes symmetric color limits for comparing plots. It does not change saved values.

Read the figure with its source table

Summary labels follow `Level | Group1 | Group2`. Both summary bars and contrast heatmaps use **Group1 minus Group2**. A positive residual contrast means higher mean residual scores in Group1; a positive refit contrast means a larger facet location in Group1. With the usual difficulty/severity orientation, the latter means harder for Group1. Check the model's facet signs and rubric coding before interpreting either as favorable treatment.

`plot_data(p, "pairs")` retains the source rows, counts and analysis conditions. For summaries, `PlotStatus` distinguishes displayed rows, finite contrasts omitted by `top_n`, and unavailable contrasts. The last two do not mean zero or no differential functioning. If all contrasts are unavailable, the summary cannot draw points; inspect `x$dif_table` or a contrast heatmap. Heatmap cells with missing values remain missing. An observed-minus-expected cell mean can still be shown when its cell is below `min_obs`; inspect `sparse`. The scaled-residual and pairwise-contrast routes withhold such values.

`N_Group1` and `N_Group2` count retained response rows. Residual results also give distinct-Person counts, which need not be independent observations. `plot_data(p, "cases")` preserves residual response matches and exclusion reasons; it is `NULL` for refit results, which have no such response audit. Refit source rows instead report `ScaleLinkStatus`, `ContrastComparable`, fixed-target flags and subgroup linking details. A finite point or an adequate linking screen does not establish formal DFF inference. See mfrmr_linking_and_dff for the different linking screens.

Interpreting output

Bars are anchored at zero. Width corresponds to effect size on the contrast's native scale. For `method = "residual"`, this is the observed-minus-expected average difference between groups, in score units. For `method = "refit"`, this is the subgroup parameter difference on the fitted logit scale when linking support allows a comparable contrast. Residual differences do not isolate differential functioning. Linked refit intervals omit estimated-anchor uncertainty and cross-refit covariance.

See Also

`analyze_dff()`, `dif_interaction_table()`, mfrmr_plot_migration

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_dif", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
# Group labels from the same synthetic data used for the saved fit.
ratings <- load_mfrmr_data(example_fit$dataset)
dff <- analyze_dff(fit, diagnostics = diag,
                   facet = "Rater", group = "Group", data = ratings)
unique(dff$dif_table$ClassificationSystem)
p <- plot(dff, type = "summary", draw = FALSE)
comparisons <- plot_data(p, "pairs")
comparisons[, c("Level", "Group1", "Group2", "Contrast",
                "N_Group1", "N_Group2", "PlotStatus")]
table(plot_data(p, "cases")$Reason)
plot_data(p, "notes")

## End(Not run)
```

---

plot.mfrm_dstudy_intervals

Compare D-study confidence interval methods

Description

Compare future plans on one shared coefficient or SEM axis. Each plan has adjacent method rows, distinguished by color, point shape and line type. Method agreement in a single dataset is not evidence of coverage.

Usage

```
## S3 method for class 'mfrm_dstudy_intervals'
plot(
  x,
  draw = TRUE,
  preset = "standard",
  ncol = 2L,
  layout = c("combined", "panels"),
  ...
)
```

Arguments

x	A result from confint.mfrm_multivariate_d_study().

draw	Logical; FALSE returns the plot data without opening a device.
preset	Package plot style, including "standard" (color) and "monochrome". Methods also have distinct point shapes.
ncol	Positive integer number of columns for layout = "panels". Unused by the default combined view.
layout	"combined" (default) groups methods within each plan on one axis, with a common method legend. "panels" retains one panel per future plan, with at most ncol columns.
...	Reserved; additional arguments are rejected.

Details

All methods concern the same score/composite, metric and source data. A filled point has an interval; an open point is an available estimate without an interval. Unavailable intervals are explicitly labeled in the combined view, or marked by an asterisk in the panel view, even if their point estimates are missing. An interval can exist without an admissible ANOVA/MINQUE point estimate. Missing values remain missing; no substitute interval or zero is drawn. The shared axis includes [0,1] for coefficients and [-1,1] for G/Phi differences, expanding to retain all finite estimates and accepted bounds. Delta normal-reference bounds outside [0,1] remain visible; they flag limitations of that approximation, not a larger population parameter space. SEM and SEM-difference axes instead span the estimates and intervals in score units, including zero. Negative SEM differences favor the comparison plan. An SEM interval is uncertainty in measurement error, not an interval for a Person's score. Chi-square and MLS labels retain the calculation used for each plan; their assumptions and coverage claims differ. Method limitations, confidence level, normal-effect assumptions and negative components remain in the returned notes and source object. Plotting neither refits a model nor recalculates an interval. Save the complete interval or plot object with saveRDS() for later use. Nested plan labels show child counts per parent, rather than total child pool sizes; the reference plan uses the same count convention. A mean-square-region difference is a conservative enclosure from shared source uncertainty, not a tight joint image or a paired-delta interval. Its returned source retains the individual projections and reference. REML profile and residual-contrast rows retain the original source points; the separate IntervalCenter and method diagnostics are available in the source interval object. A point can lie outside this separately calculated interval. For GENOVA-adjusted results, points use the selected component treatment; intervals retain their original-source population basis. IntervalEstimate and IntervalBasis in the source table distinguish them. Drawing the two together does not make the original sampling SE an adjusted-estimator SE. Unresolved numerical candidates are retained in the source object but are not drawn as accepted confidence bounds. For score and MLS intervals, an empty confidence set is labeled "Empty set"; it is distinct from an estimated coefficient of zero. MLS rows with FormulaComplete = FALSE retain candidate values where a squared width is undefined. Their conservative bounds are drawn, while the saved interval object and notes retain the reason and affected ranges.

Value

Invisibly, an mfrm_plot_data object retaining the exact table, unavailable rows, source intervals, design_grid, weights and notes. Use plot_data() to inspect them and as_ggplot() on the returned object for an editable plot. No title or explanatory footnote is drawn.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`confint.mfrm_multivariate_d_study()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_dstudy_intervals", run.dontrun = TRUE)
## Not run:
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
d <- mfrm_d_study(g, data.frame(Tasks = c(6, 12)))
ci <- confint(d, assumption = "normal")
p <- plot(ci, draw = FALSE)
plot_data(p, "notes")
if (requireNamespace("ggplot2", quietly = TRUE)) as_ggplot(p)

## End(Not run)
```

---

plot.mfrm_d_study *Plot design comparisons from a main-effects D-study*

Description

Compare planned facet counts using an existing `mfrm_d_study()` result. G concerns relative ordering; Phi also includes shifts in absolute score levels. Larger coefficients mean greater dependability under the selected model and residual assumption, not proven pass/fail accuracy.

Usage

```
## S3 method for class 'mfrm_d_study'
plot(
  x,
  y = NULL,
  type = c("coefficients", "error_variance", "heatmap", "contour", "surface3d"),
  x_var = NULL,
  y_var = NULL,
  group_var = NULL,
  panel_by = NULL,
  panel_grid = NULL,
  metric = NULL,
  draw = TRUE,
  main = NULL,
  palette = NULL,
  preset = c("standard", "publication", "compact", "monochrome"),
  reference = NULL,
  ...
)
```

Arguments

x	An mfrm_d_study() result.
y	Reserved for method compatibility.
type	"coefficients" for G/Phi curves, "error_variance" for error variance curves, or "heatmap", "contour", or "surface3d" for one metric over two facet counts. Error variance is in squared score units, not SEM. Lower error variance is better.
x_var, y_var	Planned-count columns such as "n_Rater". The default horizontal axis is the first count column; surface plots use the next column on the other axis. The two axes must differ.
group_var	Optional additional column distinguishing curves. All non-horizontal facet counts and residual assumptions remain separate within each panel, including when group_var is supplied.
panel_by	One column defining panels, or NULL.
panel_grid	One or two columns defining panels. Use this or panel_by, not both. Surface plots require every other facet count and residual assumption to be constant within each panel. Subset the result or add panels when they vary; they cannot be silently averaged or overlaid.
metric	Optional selection from "G", "Phi", "RelativeErrorVariance", or "AbsoluteErrorVariance", compatible with type. Surface plots require one metric and default to "Phi".
draw	Draw when TRUE; FALSE only returns plot data.
main	Optional plot title.
palette	Optional colors.
preset	Plot style: "standard", "publication", "compact", or "monochrome".

`reference`	Optional distinct finite numeric reference values for curves, between 0 and 1 for coefficients or nonnegative for error variance. `NULL` draws no guides. Not supported for surface plots.
`...`	Reserved for method compatibility.

Details

Points represent requested scenarios; connecting lines and contours are visual guides. Missing estimates remain missing and break curves. If none are available, inspect the D-study table and source variance components. No reference lines are drawn by default. Specify `reference = c(0.70, 0.80)` to reproduce the earlier coefficient guides. User-chosen guides are not universal acceptance criteria or confidence bounds; their values and interpretation remain in the returned plot data. Heatmaps include a numeric color key; exact values remain in the table.

The main-effects G-study combines unmodeled interactions in its residual. Different residual-scaling curves describe assumptions, not confidence bounds. All projections hold estimated components fixed. Check source fit warnings before interpreting even large coefficients. For supported designs with separately estimated interactions, including a single score, use `mfrm_g_study()` and `mfrm_d_study()`.

Value

Invisibly, an `mfrm_plot_data` object with the scenario `table`, metric `series`, axis/group/panel settings, and labels. Use `plot_data()` for custom graphics. Automatic `as_ggplot()` conversion is not provided for this class; the base plots preserve the chosen comparisons.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

mfrm_d_study(), plot.mfrm_multivariate_d_study()

Examples

```
# After creating ds with mfrm_d_study():
# plot(ds, x_var = "n_Rater", panel_grid = c("Metric", "ResidualScaling"))
# For Rater x Task x Occasion scenarios, separate occasions explicitly:
# plot(ds, type = "heatmap", x_var = "n_Rater", y_var = "n_Task",
#      metric = "Phi", panel_by = "n_Occasion")
# With residual_scaling = "sensitivity", use
# panel_grid = c("n_Occasion", "ResidualScaling") instead.
```

---

plot.mfrm_facet_nesting

Plot the pairwise nesting index matrix

Description

Renders the directed nesting index $1 - H(B \mid A)/H(B)$ as a heatmap between facet pairs, highlighting fully nested relationships close to 1. Colour scale runs from 0 (crossed, white / cold) to 1 (fully nested, dark).

Usage

```
## S3 method for class 'mfrm_facet_nesting'
plot(x, preset = c("standard", "publication", "compact", "monochrome"), ...)
```

Arguments

x	An mfrm_facet_nesting object.
preset	Plot preset.
...	Reserved.

Value

Invisibly, the matrix rendered.

Session plot defaults

Set options(mfrmr.plot_preset = "publication") to choose a session default for plotting functions that expose the common preset argument. The supported values are "standard", "publication", "compact" and "monochrome". Precedence is an explicit call argument, then the session option, then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`detect_facet_nesting()`, `analyze_hierarchical_structure()`.

`plot.mfrm_fit` *Plot fitted MFRM results with base R*

Description

Plot fitted MFRM results with base R

Usage

```
## S3 method for class 'mfrm_fit'
plot(
  x,
  type = NULL,
  facet = NULL,
  top_n = 30,
  theta_range = c(-6, 6),
  theta_points = 241,
  title = NULL,
  palette = NULL,
  label_angle = 45,
  show_ci = NULL,
  ci_level = 0.95,
  group = NULL,
  diagnostics = NULL,
  include_fit_measures = TRUE,
  fit_stat = c("Infit", "Outfit"),
  fit_scale = c("mnsq", "zstd"),
  zstd_method = c("facets", "engine"),
```

```
  include_person = FALSE,
  person_subset = NULL,
  top_n_person = 30,
  person_labels = c("flagged", "all", "none"),
  facet_labels = c("all", "flagged", "none"),
  panel = c("combined", "facet"),
  fit_range = c(0.5, 1.5),
  zstd_cut = 2,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  renderer = NULL,
  wright_style = c("native", "facets_style"),
  category_labels = NULL,
  rows_per_logit = 2L,
  wright_range = NULL,
  extreme_placement = c("ends", "estimate"),
  persons_per_star = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  ...,
  level = NULL,
  view = NULL,
  metric = NULL,
  comparison = NULL,
  labels = NULL,
  highlight = NULL
)
```

Arguments

x	An `mfrm_fit` object from `fit_mfrm()`.
type	Plot type. Omit `type` (or use `NULL` / `"wright"`) for the primary native Wright map. Use `"bundle"`, `"all"`, or `"default"` for the three-part fit bundle; otherwise choose one of `"facet"`, `"person"`, `"step"`, `"wright"`, `"pathway"`, `"fit_pathway"`, `"ccc"`, `"ccc_surface"`, or `"category_surface"`. `"pathway"` is the theta-to-expected-score display; `"fit_pathway"` is the fit-statistic-to-measure display. Additional diagnostic and reporting views are listed under "Single-fit views" in mfrmr_plot_migration, with replacements for all former helpers.
facet	Optional facet name for `type = "facet"`.
top_n	Maximum number of non-person facet locations retained by the native Wright map for compact displays. Step transitions are always retained separately and do not consume this limit; any omitted facet locations are counted in the returned `retention` table and disclosed in the plot subtitle/note. Use `Inf` for a complete all-level final map. The FACETS-style payload always retains every fitted facet and step location.
theta_range	Numeric length-2 range for pathway, CCC, and category-surface plot data.
theta_points	Number of theta grid points used for pathway, CCC, and category-surface plot data.

title	Optional custom title.
palette	Optional color overrides.
label_angle	Rotation angle for x-axis labels where applicable. Native Wright maps reserve bottom space for full labels when abbreviating them would make different groups indistinguishable.
show_ci	Whether to add approximate confidence intervals when available. NULL selects the display-specific default: TRUE for the primary native Wright map and type = "fit_pathway", and FALSE for the FACETS-style ruler and other plot types. With renderer = "facets", explicitly setting show_ci = TRUE creates a hybrid display: FACETS-style ruler grammar with mfrmr uncertainty intervals.
ci_level	Confidence level used when show_ci = TRUE.
group	Optional grouping for type = "wright" to overlay per-group person-density curves (DIF / DFF screening view). Either a column name (looked up first in group_data when supplied through ..., then in fit$prep$data) or a vector aligned with fit$facets$person. Ignored for other type values and for wright_style = "facets_style", whose person column is a single FACETS-style star frequency. To pass the source data alongside, use plot(fit, type = "wright", group = "MyC
diagnostics	Optional matching output from diagnose_mfrm(). Recompute it after refitting; mismatched or outdated readiness records are rejected. When supplied, Wright-map standard errors and precision metadata reuse matching rows from diagnostics$measures without replacing fitted coordinates, while pathway plot data reuse fit_measures, fit_status, and curve_fit_status instead of recomputing diagnostics. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
include_fit_measures	If TRUE (default), pathway plot data include tidy fit-measure and fit-status tables for custom R graphics. Set to FALSE when only the curve coordinates are needed.
fit_stat	For type = "fit_pathway", the horizontal fit statistic: "Infit" (default) or "Outfit".
fit_scale	For type = "fit_pathway", use mean squares ("mnsq") or standardized fit ("zstd") on the horizontal axis.
zstd_method	For a ZSTD fit pathway, "facets" (default) uses the FACETS/Wright-Masters fourth-moment df, matching diagnose_mfrm(). "engine" retains the former package-native df convention. Neither establishes complete FACETS output equivalence. Refreshing companions from supplied diagnostics preserves their recorded WHEXACT choice.
include_person	If TRUE, include person rows in a fit pathway.
person_subset	Optional character vector of person IDs retained in a fit pathway before ranking.
top_n_person	Maximum person rows retained in a fit pathway, ranked by distance from expected fit. Use Inf for all selected persons.
person_labels	Person-label policy for a fit pathway: "flagged" (default), "all", or "none".
facet_labels	Facet-level label policy for a fit pathway: "all" (default), "flagged", or "none". This changes drawn labels only; all retained facet rows remain in the draw-free table.

panel	Fit-pathway layout: "combined" (distinguish persons by shape) or "facet" (one base-graphics panel per facet).
fit_range	Mean-square review lines for a fit pathway.
zstd_cut	Absolute ZSTD review line for a fit pathway.
draw	If TRUE, draw the plot with base graphics.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
renderer	Wright-map renderer. Use "native" (default) or "facets" for the FACETS Table 6-style visual layout. This is the canonical selector for new code. The native map is the primary display and includes available facet uncertainty by default. The closest FACETS-style visual uses show_ci = FALSE.
wright_style	Wright-map renderer. "native" preserves the package's histogram, point, range, and facet-SE display. "facets_style" adds a FACETS Table 6-style text ruler with person-frequency stars, signed facet headers, and labeled score-transition lines. It is a visual-layout option, not a claim of numerical equivalence with FACETS. This explicit style name is equivalent to renderer = "facets". Adding show_ci = TRUE to that style is an mfrmr/FACETS hybrid rather than the closest FACETS-style view.
category_labels	Optional score rubric labels used by wright_style = "facets_style". Supply a named character vector keyed by original score, an unnamed vector with one label per retained category, or a data frame with Score and Label columns.
rows_per_logit	Number of FACETS-style ruler rows per logit.
wright_range	Optional finite increasing length-2 logit range for the FACETS-style ruler. NULL derives a range that contains the fitted data.
extreme_placement	In the FACETS-style renderer, place extreme-score persons at the ruler "ends" or retain their fitted "estimate".
persons_per_star	Number of persons represented by one * in the FACETS-style frequency column. NULL chooses a compact value automatically.
show_title	Logical; display the main plot title (default FALSE). Structural panel headings, axis labels, legends, and data labels remain visible.
show_notes	Logical; display explanatory subtitles, footnotes, and review-only title markers. Notes remain available in the returned object, and R warnings are still issued. Default FALSE; use plot_data(p, "notes") for the descriptions and cautions outside the figure.
...	View-specific controls listed in mfrmr_plot_migration. The consolidated routes reject irrelevant or duplicated arguments.
level	Named alternative to ci_level, a single number strictly between 0 and 1. Supply only one of these names, even when equal. Both omitted retain 0.95. This controls the fitted-model plot intervals; attached interval results retain the level chosen when they were computed.
view	Optional display variant for consolidated routes: for example, "pairs" or "heatmap" for agreement, "unified" for the former unified Wright helper, or "funnel" for shrinkage. See mfrmr_plot_migration.

metric	Quantity for a consolidated route, for example "reliability", "separation" or "strata", or "exact" / "correlation" for agreement.
comparison	Another fitted model for paired Wright/CCC displays, or an ordered list of additional fits for type = "trajectory". Plotting does not equate their scales or perform a model-selection test.
labels	Labels for paired models or all trajectory waves, reference first.
highlight	Optional named list of non-person facet level IDs, for example list(Rater = c("R01", "R02")), for Wright maps only (including view = "unified" and the Wright component of a bundle). Selected labels receive a > prefix in both renderers and saved ggplot conversion. Levels outside the usual top_n selection are added, so the displayed count may exceed that limit. Fitted locations and uncertainty do not change; the automatic scale uses all fitted locations before compaction. Selection is not a statistical flag. Inspect locations$Highlighted, retention and notes in the returned plot data. Unknown IDs or levels without finite locations are rejected. NULL selects no levels.

Details

Start with plot(fit): it draws a Wright map of person abilities, rater severities, other facet locations, and category thresholds on one logit scale. No plot options are needed for this first view.

For adjusted-score comparisons, use plot(fit, type = "fair_average", show_ci = TRUE). metric = "FairM" uses mean reference measures; metric = "FairZ" uses a zero reference, not a z-score. PCM averages expectations over administered scales with the table's observation weights and totalscore selection. Its conditional delta-method intervals use the same retained observations as the table: finite direct anchors in ordinary additive RSM/PCM JML are retained even with all-minimum/all-maximum responses, including when totalscore = FALSE. The intervals use the derivative of that same average. RSM/PCM intervals hold thresholds, other effects and reference means fixed; they are diagnostic approximations, not full calibration-uncertainty intervals. Inspect the returned data and notes for the interval method and unavailable rows. See fair_average_table() for reference definitions and plot_fair_average for view-specific controls. For ordinary additive RSM/PCM JML, xtreme = .3, xtreme_method = "facets" requests conditional total-score extrapolation of extreme measures and their fair scores. Primary estimates and fixed anchors are retained. Group-mean anchors are unsupported. The measure SE describes the corrected total; fair-score intervals remain unavailable for positive xtreme. The default xtreme_method = "display" retains the old average-score display convention and leaves fair scores unchanged.

This S3 plotting method provides the core fit-family visuals for mfrmr. When type is omitted, it returns the Wright map alone as an mfrm_plot_data object (the most useful single figure for a first inspection). Pass type = "bundle" (or "all" / "default") to obtain the legacy three-plot mfrm_plot_bundle containing a Wright map, pathway map, and category characteristic curves. The compact native default records any omitted facet locations in data$retention and annotates the subtitle and drawn figure; use top_n = Inf to retain every fitted location in the final Wright map. Every retained native location is labelled. LabelX / LabelY provide initial text positions; base graphics further adjusts these positions for the device's text dimensions, avoiding nearby labels and fitted points where space permits. Leader lines connect displaced text to the unchanged point. label_points retains fitted coordinates and initial text positions for custom renderers. Step transitions share one vertical ladder and their labels include the fitted threshold logit. When the fit records boundary-separated facet levels and no display range was supplied, the native and FACETS-style maps use the same robust automatic range and place those

levels at ruler ends. Exact fitted values and intervals remain in `OriginalEstimate`, `CI_Lower`, and `CI_Upper`; `DisplayEstimate`, `DisplayCI_Lower`, `DisplayCI_Upper`, and the clipping-status columns describe only the rendered coordinates. Endpoint triangles and the plot footer disclose any omitted or clipped interval. The returned object always carries machine-readable metadata through the `mfrm_plot_data` contract, even when the plot is drawn immediately. Set `show_title = FALSE` and `show_notes = FALSE` for a figure whose title and explanation will be supplied by the surrounding document. The returned `data$notes` table contains `Type` and `Text` columns for interpretation, reference-profile conditions, and display/retention notes where applicable; `print()` also prints these notes. `data$display` records the two flags. Original titles, subtitles, coordinates, and readiness metadata are retained. Device-dependent crowding warnings are issued during drawing rather than stored in this draw-free notes table. `as_ggplot()` respects these flags and retains the notes in `attr(plot, "mfrmr_notes")`.

Fit-family Wright, pathway, and CCC displays use a shared, CUD-informed palette. The first eight series use an Okabe-Ito foundation with bright yellow omitted and darker pink/amber/cyan entries for light backgrounds. Beyond eight series, colours come from the dark half of the viridis HCL palette. Expected-score and category curves also vary line type; Wright locations retain their distinct point shapes, and small data labels use dark neutral text. Six line types cycle for larger series sets, so dense plots still require labels, panels, or more space. This is not a guarantee of perceptual separation for every viewer or device. `preset = "monochrome"` applies to these series in both base and ggplot renderers. Custom `palette` values are retained in `data$palette` and used during ggplot conversion; colour overrides keep the non-colour encodings. See https://jfly.uni-koeln.de/color/ for Color Universal Design guidance.

Every fit-derived payload also carries `data$fit_readiness`, `data$interpretation_status`, and `data$interpretation_note`. Availability and interpretability are separate: when a numerical, data, design, or stability status requires review, the coordinates remain available for diagnosis, but the call warns and marks the returned subtitle and drawn title `REVIEW ONLY` by default. Hiding that annotation changes presentation only; it does not change the fit's interpretation status or suppress warnings. A prior-regularized extreme MML EAP remains in `fit$facets$person`. If its source fit is blocked, however, it is omitted from the Wright-map scale so that a finite but non-interpretable trace cannot collapse the display. The exact excluded rows and reason remain in `data$person_exclusions`, and the omission is stated in `data$retention_note`.

`type = "unexpected"` uses the same default as `unexpected_response_table()`: `abs_z_min = 3, rule = "residual"`. The optional `rule = "either"` or `"both"` also uses `prob_max`. Saved table plots retain the recorded thresholds. A fully evaluated screen with no flags yields an empty display and retains its screening summary.

`type = "wright"` shows persons, facet levels, and step thresholds on a shared logit scale. Estimates are plotted as fitted, so the sign convention follows the fit: higher Person values predict higher coded scores in these positive-discrimination models. Whether that means more comprehensible or less comprehensible speech depends on the rubric coding. Under the default negative orientation, higher non-Person facet values reduce expected coded scores; calling this greater severity/difficulty presumes that higher scores represent better performance. Facets listed in `fit_mfrm(positive_facets = ...)` are reversed (higher values raise expected scores); state the active orientation in figure captions when reporting. The returned `notes`, also shown by `print()`, state the recorded facet directions and the score-coding caveat. Missing orientation in an older or imported result is not guessed. Category labels do not recode the data. See mfrmr_visual_diagnostics for reverse-coded comprehensibility, fit, conditional bias and Step explanations.

Set `renderer = "facets"` (equivalently, `wright_style = "facets_style"`) for a line-printer-inspired

common ruler. That mode retains every facet location in its data and shows actual (original, when scores were internally recoded) adjacent score transitions. Its `facets_style` payload contains tidy ruler, person-frequency, facet, step, mean-half-score, header, category-label, and settings tables for custom graphics. `RulerValue` records the nearest discrete ruler row for line-printer reconstruction, whereas step and midpoint lines are drawn at their exact fitted `DrawValue` and step labels print that logit value. The styling emulates the semantics of FACETS Table 6; estimates and standard errors remain those produced by `mfrmr`. Use `show_ci = FALSE` for the closest FACETS-style rendering. If `show_ci = TRUE` is requested, interpret the result as a hybrid that adds mfrmr uncertainty intervals to FACETS-style ruler grammar. Column headings are retained even on narrow devices. If headings or person frequency stars need more space, drawing warns; use a wider device or increase `persons_per_star` for a more compact frequency column.

`type = "pathway"` shows expected score traces and dominant-category regions across theta. This expected-score display is distinct from the Bond-and-Fox-style fit-versus-measure pathway: use `type = "fit_pathway"` for Infit/Outfit on the horizontal axis and measure logits on the vertical axis. Its draw-free plot data include person-selection settings, CI columns, and explicit SE-source metadata. The expected-score pathway draw-free data also includes `pathway_long`, `pathway_annotations`, `fit_measures`, `fit_status`, and `curve_fit_status`, so R users can rebuild the pathway map in ggplot2, plotly, or a report pipeline while keeping the same underfit/overfit labels used by `fit_measures_table()`. `type = "ccc"` shows category response probabilities. Multiple curve groups are faceted rather than overplotted by the native renderer, and category-specific legends use the same colours across panels. Multiple groups or more than five categories use one legend beside the plotting area; colour presets use distinct default colours beyond eight categories. For `GPCM`, these curves retain the estimated slope. With separate owners, each step-level/slope-level pair is a labeled curve group. Such joint profiles have no single-facet Infit/Outfit flag; inspect `fit_measures` or `type = "fit_pathway"` separately. All curve families are reference-profile curves with additive facet main effects and fitted interactions fixed at zero; the native footer and ggplot subtitle disclose that conditioning. Expected-score pathways use the same reference profile and expose the same `curve_basis` table. These curves do not average over the observed rater assignments or show a particular fitted interaction cell. The draw-free `curve_basis`, `CurveBasis`, and `PredictorOffset` fields make that conditioning explicit. `type = "ccc_surface"` or `type = "category_surface"` returns 3D-ready category-probability surface data for external rendering; it deliberately does not add a plotly/rgl dependency or replace the 2D CCC/pathway reporting figures. The returned object includes `category_support`, `interpretation_guide`, and `reporting_policy` tables so retained zero-frequency categories and manuscript-use boundaries remain visible to beginners. The remaining types (`"facet"`, `"person"`, `"step"`, `"shrinkage"`) provide compact location-specific displays.

Value

Invisibly, an `mfrm_plot_data` object (default and for any single `type`), or an `mfrm_plot_bundle` when `type = "bundle"` / `"all"` / `"default"`. Each returned fit plot includes domain-specific readiness and an interpretation status in its data payload, plus a `notes` table and `display` settings. It also includes a one-row `scale_contract` table recording the fitted coordinate basis, population SD when applicable, discrimination basis, and GPCM MML identification convention.

Facet locations and practical equivalence

`plot(fit, type = "rater_severity", facet = "Rater")` orders locations and shows available normal uncertainty intervals. Pass matching `diagnostics` to reuse their precision metadata. The default has no magnitude bands; `show_bands = TRUE` explicitly adds gray guides at +/-0.5 and +/-1.0

logits. These distances from the fitted zero are not equivalence bounds, evidence of homogeneity or calibration standards. The returned `data` table uses numerical absolute-location ranges in `Band`; a common range does not imply small pairwise differences.

Use `p <- plot(fit, type = "rater_severity", diagnostics = diagnostics)` and `plot_data()` to read `"data"` and `"notes"`. Intervals retain their `CIUse`, `SE_Method` and `IntervalStatus`; individual whisker overlap is not a contrast interval. `analyze_facet_equivalence()` separately examines pairwise differences with their full covariance and a prespecified practical bound in its supported MML scope. It does not use the profile's guides. The zero depends on the model constraints/anchors, so read the signed differences and their uncertainty instead of inferring quality from color. Older saved profiles retain their original colors and labels when converted with `as_ggplot()`; regenerate the figure from the saved fit and matching diagnostics for the new display. No refitting is needed.

Corrected JML plots

A fit with an explicit `jml_correction_order` uses `type = "slopes"` (default), `"locations"` or `"steps"`. Locations show one `facet`, defaulting to the first declared facet. `style = "points"` shows estimates without intervals; `"distribution"` shows their empirical cumulative distribution. These displays do not classify rater quality. They support `title`, `caption`, `show_title`, `show_notes`, `show_labels`, `sort`, `palette`, `preset`, `text_scale`, `point_size`, and draw-free `as_ggplot()` conversion. Slopes use a reference of one; location/step displays use zero. Ordinary plot types, `show_ci = TRUE`, and unrelated plot controls are rejected for this estimator. The figure retains point estimates if their RootSE is unavailable; it cannot be drawn if no unambiguous point solution was obtained.

Graphics layout

Single-panel plots advance through a caller's `par(mfrow = ...)` or `layout()` arrangement and restore the style and margins they change. Native Wright maps, multi-group CCC plots, and faceted fit pathways create their own page layouts; call these outside a custom `layout()` arrangement. Use `renderer = "facets"` for a single-panel Wright map in a custom grid. A 7 by 5 inch device is a useful starting size for standalone plots; dense PCM labels benefit from a wider device. Native Wright/pathway plots warn when labels cannot be separated within the available space. Enlarge the device (for example, to 12 by 8 inches) and inspect the result; this warning concerns readability, not model fit. Abbreviated labels retain distinguishing text where possible, falling back to full names if needed. Original names remain available in the plot data. For non-Latin labels, select a graphics device and font that support the characters before plotting; the package retains the caller's font family.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Use `plot(fit)` to inspect the Wright map at a glance.
3. Switch to `type = "pathway"`, `"fit_pathway"`, `"ccc"`, or `"shrinkage"` for the relevant follow-up figure, or `type = "bundle"` for the three-plot overview when preparing a FACETS-style summary.

Further guidance

For a plot-selection guide and extended examples, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnosti package = "mfrmr")`.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`fit_mfrm()`, plot(x, type = "wright", view = "unified"), plot(x, type = "bubble"), mfrmr_visual_diagnostics

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Run each plot command separately to inspect its figure
plot(fit) # Wright map: persons, facets, and category thresholds

# Rater severity estimates (higher means stricter in this example)
plot(fit, type = "facet", facet = "Rater")

# Probability of each score category across the ability scale
plot(fit, type = "ccc")

# Optional: get plot data instead of drawing a figure
```

```
wright <- plot(fit, draw = FALSE)
head(wright$data$locations)
# Find a named judge without refitting or dropping the surrounding scale
judge <- subset(wright$data$locations, Group == "Rater")$Label[1]
focused <- plot(fit, highlight = list(Rater = judge), draw = FALSE)
subset(plot_data(focused, "locations"), Highlighted)
```

`plot.mfrm_multivariate_d_study` *Compare measurement-condition counts in a multivariate D-study*

Description

See how a planned change in the number of tasks, raters, or other conditions affects the dependability of a mean score. Plot an existing D-study result; no model is refitted. Start with `plot(d)` for G and Phi, then use `plot(d, type = "sem")` to examine error in score units. If the result has several composites, select them explicitly with `composite`.

Usage

```
## S3 method for class 'mfrm_multivariate_d_study'
plot(
  x,
  type = c("coefficients", "sem", "error_components"),
  score = NULL,
  x_var = NULL,
  draw = TRUE,
  preset = "standard",
  composite = NULL,
  metric = NULL,
  reference = NULL,
  panel_by = NULL,
  ncol = NULL,
  intervals = FALSE,
  assumption = NULL,
  method = "auto",
  level = 0.95,
  ...
)
```

Arguments

`x`	A result from `mfrm_d_study()`.
`type`	`"coefficients"` shows G and Phi in separate panels; `"sem"` shows relative and absolute standard errors of measurement; `"error_components"` shows each source's contribution to planned error variance, with separate panels for selected scores and error definitions.

`score` One or more distinct original score names, in legend order. An explicit name always selects an original score, even if a composite has the same name. If both `score` and `composite` are `NULL`, select the sole composite, or the first score when there are no composites. Several composites require an explicit selection. `plot_data()` identifies the selection and composite weights; a suggested title is returned rather than drawn on the figure.

`x_var` A count column in `x$design_grid`: `"Tasks"`/`"Raters"` for the task/rater interface, or a facet label supplied through `facets`. By default use the last varying count column, or the last column if none varies. A one-facet result permits only its included facet. Each line holds the other facet count constant.

`draw` Draw the plot when `TRUE`; `FALSE` only returns its data.

`preset` Plot style: `"standard"`, `"publication"`, `"compact"`, or `"monochrome"`. The first three use distinct curve colors; monochrome uses one dark color. Point shapes and line types also distinguish curves. An explicit preset overrides `options(mfrmr.plot_preset = ...)`. `as_ggplot()` retains colors saved in a plot payload.

`composite` One or more distinct composite names from `x$coefficients`, as defined by a weight-matrix column, or `"Composite"` for vector weights. Supply either `score` or `composite`, not both.

`metric` Optional single metric compatible with `type`: `"G"` or `"Phi"` for coefficients; `"RelativeSEM"` or `"AbsoluteSEM"` for SEM. For error components, select `"RelativeErrorVariance"` or `"AbsoluteErrorVariance"`. `NULL` shows both metrics in separate panels.

`reference` Optional numeric reference values: between 0 and 1 for coefficients, nonnegative for SEM. `NULL` draws no reference lines. The same values apply to each panel. These are user-chosen guides, not universal acceptance rules or confidence bounds.

`panel_by` Optional non-horizontal count column in `x$design_grid`, such as `"Raters"` when `x_var = "Tasks"`. Each count gets a separate panel; selected scores/composites keep the same color, line type and point shape across panels. `NULL` retains the overlaid curves.

`ncol` Optional positive integer number of panel columns. Requires `panel_by` for curves; by default use up to two columns. Error-component plots use one column by default and do not use `panel_by`. With both curve metrics selected, panels run through all counts for the first metric, then the second.

`intervals` `FALSE` (default) draws point curves. `TRUE` computes intervals and adds ribbons in one call; also specify `assumption = "normal"` and select one `metric`: `"G"`, `"Phi"`, or, with `type = "sem"`, `"RelativeSEM"` or `"AbsoluteSEM"`. Alternatively, supply a saved `confint(x)` result, or a list with one result per selected score/composite, to avoid recalculation. Saved results must use one method/level and match the source, weights and design grid; retain unavailable rows. `NULL` is equivalent to `FALSE`. Use `plot(ci)` separately to compare several interval methods.

`assumption` With `intervals = TRUE`, explicitly specify `"normal"` for independent normal random effects; this assumption is not tested.

`method` With `intervals = TRUE`, the interval method passed to `confint.mfrm_multivariate_d_study()`. `"auto"` selects exact F for one-facet G and generalized CI for one-facet Phi,

also for nested plans with source child counts per parent unchanged. For two crossed random facets, "auto" selects approximate "mls" null inversion for G/Phi. Undefined MLS widths are conservatively retained and flagged in the saved interval result. For alternatives or changed nested child counts, select "approximate_f" or "mean_square_region". For SEM, "auto" uses "mls" for supported balanced crossed/nested sources: chi-square for a single mean square, Graybill-Wang for positive sums, Ting for signed sums. Two-facet SEM also accepts "mean_square_region" for conservative normal-model intervals. "reml_profile" explicitly fits approximate G/Phi intervals for balanced or identifiable incomplete sources; lme4 and minqa are required for fitting. It preserves the original curve points and reports separate REML estimates in the saved intervals. Its finite-sample/boundary coverage is not guaranteed. For one-facet RelativeSEM, explicit "residual_contrast" accepts retained ANOVA/MINQUE ratings, including incomplete sources. It transforms a residual chi-square interval without likelihood fitting, retaining original curve points and a separate projected center in the saved table. For one-facet G, explicit "spectral_f" accepts retained ANOVA/MINQUE ratings, including incomplete sources, using a normal-model variance-ratio F pivot. It keeps the original points; sparse sources can yield wide bands. Balanced crossed G also accepts explicit "score", an experimental extended-nuisance score interval calculated from stored mean squares. It preserves the source points; empty sets and numerical failures remain missing ribbon segments, with their reasons in the saved interval object. No method is silently substituted after failure.

`level` Confidence level for `intervals = TRUE`, default 0.95.

`...` Reserved for future use; additional arguments are rejected.

Details

G describes consistency of relative ordering: for example, ranking examinees. Phi also includes shifts caused by easier tasks or more lenient raters and concerns absolute score levels. Higher coefficients indicate greater dependability under the fitted model; neither gives the probability of a correct pass/fail decision. No universal acceptable cutoff is drawn.

SEM is the square root of error variance. Lower values indicate less error in the units of the selected mean score or composite. Relative SEM concerns ordering; absolute SEM also includes condition-wide shifts. SEM is not a confidence interval for G or Phi. The two SEM panels share a scale, but different scores or differently scaled composites need not have comparable units. Weights are not normalized. Changing weights can also change the meaning of the score; a higher G or Phi alone does not justify a new set of weights.

With `type = "error_components"`, compare how complete plans change the sources of error. Each point is a projected variance component divided by its planned replication count. Relative error includes Person interactions and residual; absolute error also includes condition-wide shifts. For two crossed facets, Person-by-rater is divided by the rater count, Person-by-task by the task count, and the highest interaction/residual by their product. In a nested design, child counts are per parent; the parent and Person-by-parent terms use the parent count, and child-within-parent and residual terms use the product. See `mfrm_d_study()` for the formulas. Sources absent from an error definition are omitted, including the Person universe component.

Contributions add to error **variance**, in squared score units, not to SEM. Composite projections use the full covariance matrices and the saved weights. Negative component estimates remain visible

even when the total error variance is negative and SEM is unavailable; they do not represent negative population variances. A negative contribution is not evidence that adding observations causes more population error. The plot compares point projections, not sampling uncertainty or an optimal allocation. Do not pass curve-only `x_var`, `panel_by`, `reference`, or `intervals`. Use a small set of relevant plans, and increase figure height for many scores or plans. Stored components and plans are checked against the saved D-study estimates without refitting the G-study.

Points represent only the supplied scenarios. Lines connect them as visual guides for each selected score at a fixed count of the other facet, without fitting a curve or evaluating intermediate designs. For example, with tasks on the horizontal axis, compare points along one line to change tasks while keeping raters constant. For overlaid curves, each legend entry identifies the score and the other facet count. With `panel_by`, the panel identifies the count and the legend identifies the score. Compare rater counts within the same score.

Multivariate means several score criteria, not a requirement for a 3D plot. Use these 2D panels to compare counts while holding the other facet fixed. With many lines, use `panel_by` to separate the other facet count, or specify a smaller set of relevant plans when creating the D-study. Panel axes share limits; only requested scenarios are drawn, even in a nonrectangular grid. Panel labels identify counts (and metrics when needed), while the figure title and cautions remain in the returned data. Increase figure height for many panels or select one `metric`. This method does not draw a heatmap or 3D surface. For uncertainty in a prespecified change, use `mfrm_multivariate_d_compare()` and its plot method; an SEM curve is not a confidence band. `plot_data()` supplies exact values, including unavailable estimates, and `preset = "monochrome"` retains line/point distinctions.

Coefficients also depend on variation among persons. If universe-score variance increases while error variance stays constant, G and Phi increase but SEM does not decrease. Therefore, comparing coefficients across different populations does not isolate improvement in the assessment. Read the SEMs, score units and target population alongside the coefficients.

Scenarios retain the G-study's crossed or nested structure, with complete balanced future conditions shared by every person and score. For a nested facet, the axis or legend says "per" to identify its count within each parent, for example "Raters per Task"; this is not the total rater pool size. Even when the G-study used incomplete data, the figure does not describe reliability of that sparse roster. Point projections use the estimated covariance components. By default no sampling intervals are shown. `intervals = TRUE` uses `confint.mfrm_multivariate_d_study()` for each selected score/composite, without changing the G-study. `method = "reml_profile"` fits a separate likelihood model once per selected score/composite. `"residual_contrast"` projects the retained ratings for one-facet relative SEM, and `"spectral_f"` does so for one-facet G; other methods use saved ANOVA summaries. To customize other inference settings such as pivotal `draws` or `seed`, pass saved `confint()` results instead. This alternative does not recalculate intervals. Source summaries, score weights, plans and stored point estimates are checked before overlaying. Older interval objects without source summaries must be recreated with `confint(x)`; their forest plots still work. Ribbons connect the supplied endpoints; intermediate plans are not evaluated. Delta normal-reference ribbons may extend outside `[0,1]`; both renderers expand the axis rather than clipping the saved limits. This indicates an approximation limitation, not a population coefficient outside its range. Missing intervals break the ribbon and leave open points when estimates exist. An isolated interval is a vertical segment. An interval can remain visible without a point estimate. Boundary line types distinguish overlapping bands in monochrome. Confidence levels, method limitations, numerical diagnostics and unavailable reasons remain in the returned object, not figure footnotes.

Ribbons do not establish simultaneous coverage across scores or metrics. Joint coverage across future plans, if any, depends on the chosen method and its model assumptions; drawing a band

supplies no additional guarantee. Overlap is not a test of score or plan differences. Measurement SEM and paired-difference intervals cannot be used as coefficient ribbons. SEM ribbons show sampling uncertainty in the population SEM, not intervals for Person scores. Their axes retain score units and include the entire interval. MLS bands are approximate; do not read a narrow ribbon as a guarantee of coverage. See `confint.mfrm_multivariate_d_study()` for the equations, small-sample evaluation and boundary limitations. A flattening curve can indicate limited gains from adding one facet alone; it does not identify a cost-effective optimum. The highest point alone does not account for uncertainty in the estimated components. Read the size of differences as well as their ordering. Equal products of rater and task counts give equal rating counts, not necessarily equal examinee burden or total cost. Unavailable candidates prevent a complete comparison for that metric.

Unavailable estimates stay missing. A panel with no estimates states the reason; otherwise returned notes report unavailable scenarios. The returned `unavailable` table preserves their counts, metrics, and `Status` values. Each panel uses its metric's status, so an unavailable Phi does not hide an available G. Non-PSD component matrices are described in the returned notes; inspect `x$component_diagnostics` before using raw projections. Increasing planned counts does not repair those component estimates. The saved `negative_components` convention is preserved in curves and error-contribution plots. Notes identify raw, GENOVA ALGORITHM or GENOVA EMS projections; original G-study diagnostics remain available. GENOVA-adjusted results accept original-source population intervals calculated for that treatment by `confint(x)`. Bands retain their original bounds and may not contain adjusted points; notes and saved intervals retain the distinction. A saved interval with a different component treatment is rejected. Missing values are never plotted as zero. Previously saved D-study objects retain their recorded values and availability: rerun `mfrm_d_study()` on the saved G-study to apply current rules.

Value

Invisibly, an `mfrm_plot_data` object. Use `plot_data()` to extract `table` (the selected scores/composites), `series` (one row per scenario and plotted metric, including missing values), `unavailable`, `design_grid`, `weights`, score identity, axis/group names, labels, `component_note`, and legend settings. In `series`, `Status` is specific to `Metric`; `table` retains the original row status and metric-specific columns. `weights` contains a named vector for one composite, a named matrix for several composites, or `NULL` for original scores. `reference_lines` and `notes` retain the chosen guides and their interpretation. When `panel_by` is supplied, `panels` maps `series$Panel` to its metric, count and label; `ncol` records the resolved layout for base and ggplot rendering. With `intervals`, `series` also retains `Lower`, `Upper`, `IntervalStatus`, `IntervalAvailable` and `IntervalRun` (connected ribbon segments). `intervals` retains the complete input objects, including diagnostics; `unavailable_intervals` retains missing bounds separately from missing point estimates. Save the complete plot object with `saveRDS()` to replay. For `"error_components"`, `table` and `series` contain the same long contribution table: `Source`, planned `Multiplier`, projected `ComponentVariance`, `Contribution`, `DiagonalContribution` and `CovarianceContribution`. The latter two separate within-score variance and cross-score covariance terms; they sum to `Contribution`. `TotalErrorVariance` and `SEMStatus` retain the planned total and its availability; `totals` retains all original D-study metrics and statuses for selected scores. `panels`, `legend`, `weights`, `design_grid`, `component_diagnostics` and `notes` preserve the interpretation. `as_ggplot()` preserves these comparisons for editing or export with the optional ggplot2 package; for example, `as_ggplot(d, type = "sem")`.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `...`. Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`mfrm_d_study()`, `mfrm_g_study()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_multivariate_d_study", run.dontrun = TRUE)
## Not run:
# Question: how much would doubling the common tasks change dependability?
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
d <- mfrm_d_study(g, data.frame(Tasks = c(3, 6, 12)),
  weights = c(V = -1, W = 1))
plot(d) # Default: the difference W minus V; higher G/Phi is better.
plot(d, type = "sem") # Error in difference-score units; lower is better.
plot(d, score = "V") # Inspect an original score separately.
# Add intervals for both criteria in the same call.
ribbons <- plot(d, score = c("V", "W"), metric = "Phi", reference = 0.8,
  intervals = TRUE, assumption = "normal")
plot_data(ribbons, "intervals") # Saved intervals and their diagnostics.
# 0.8 above is an illustrative user criterion, not a recommended cutoff.
values <- plot_data(plot(d, draw = FALSE))
values$table # Exact values and availability, rather than reading off a line.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  p <- as_ggplot(d, type = "sem")
  print(p)
  # ggplot2::ggsave("d-study-sem.png", p, width = 7, height = 7, dpi = 300)
```

```
}

# A weight matrix names several choices; explicitly select the plotted one.
alternatives <- mfrm_d_study(g, data.frame(Tasks = c(3, 6, 12)),
  weights = cbind(Equal = c(V = 0.5, W = 0.5), Difference = c(V = -1, W = 1)))
plot(alternatives, composite = "Equal")
plot(alternatives, composite = "Difference", type = "sem")
if (requireNamespace("ggplot2", quietly = TRUE)) {
  print(as_ggplot(alternatives, composite = "Equal"))
}
# Which sources explain error in the difference score?
sources <- plot(alternatives, composite = "Difference", type = "error_components")
plot_data(sources, "table") # Includes signed cross-score covariance terms.
plot_data(sources, "totals") # Error totals, SEM, G/Phi and availability.

## End(Not run)
```

`plot.mfrm_multivariate_gstudy` *Inspect sources of observed-score variation*

Description

Plot raw variance components for selected scores or one prespecified weighted score, or inspect the component covariance matrices. Then use `mfrm_d_study()` to assess how changing the measurement conditions affects G, Phi and SEM.

Usage

```
## S3 method for class 'mfrm_multivariate_gstudy'
plot(
  x,
  score = NULL,
  weights = NULL,
  draw = TRUE,
  preset = "standard",
  type = c("components", "covariance"),
  ncol = NULL,
  intervals = FALSE,
  assumption = NULL,
  level = 0.95,
  method = "auto",
  draws = 20000L,
  seed = 20261016L,
  ...
)
```

Arguments

x	A result from mfrm_g_study().
score	Original score names in display order. Multiple scores produce separate component panels. The default selects the first score for type = "components", or all scores for type = "covariance". Do not supply both score and weights.
weights	Optional named numeric vector defining one composite. Name every score exactly once, with finite weights and at least one nonzero value. Names determine order. Signed weights permit difference scores; weights are not normalized. For a saved D-study, use its selected weight vector, for example plot(d$gstudy, weights = d$weights).
draw	Draw the figure; FALSE only returns its data.
preset	Plot style: "standard", "publication", "compact", or "monochrome". Omission uses the session plot default.
type	"components" for variance-component points, or "covariance" for one matrix heatmap per source. With intervals = TRUE, covariance instead displays one signed interval plot for exactly two selected scores. Covariance plots do not accept weights.
ncol	Positive number of panel columns. Defaults to at most two.
intervals	Logical; FALSE (default) shows point estimates. TRUE adds component confidence intervals in the same call; also specify assumption = "normal". Available for complete balanced ANOVA component plots, including supported nesting. Covariance intervals require an explicit method and exactly two scores. Explicit method = "reml_profile" also provides variance and two-score covariance intervals for identifiable incomplete sources, fitting a separate REML model while preserving the original points. method = "residual_contrast" uses retained ratings without likelihood fitting, but only Residual receives an interval. Other points remain visible with unavailable intervals and their reasons. Other methods use saved ANOVA summaries without refitting.
assumption	With intervals = TRUE, explicitly specify "normal" for independent normal random effects with constant component variances. This assumption is not tested.
level	Pointwise confidence level for intervals = TRUE, default 0.95.
method	Component interval method passed to confint.mfrm_multivariate_gstudy(). Single-score/composite variances use "auto" or "mls" for balanced ANOVA, or explicit experimental "reml_profile" for balanced/identifiable incomplete sources. Profile intervals have approximate calibration and no accompanying sampling SE. Covariance intervals require explicit "gci", "regression_region" or "reml_profile"; none is an automatic default. Explicit "residual_contrast" also supports incomplete sources for Residual only: exact normal-model variance limits or conservative covariance limits. It retains original points and a separate interval center without supplying the original estimator's sampling SE.
draws, seed	Numerical settings for covariance method = "gci". Defaults are 20000 pivotal draws and seed 20261016. See the confidence interval help for requirements and numerical endpoint bands.
...	Reserved; additional arguments are rejected.

Details

Each point is a variance-component estimate in squared score units. For weights w, the component is $w'\Sigma_s w$, including between-score covariances. Multiplying all weights by a constant multiplies each component by its square. Components describe the source model at the individual rating level; they have not been divided by future facet counts. They are not percentages, measurement-error shares for a future mean score, or MFRM severity estimates. A larger Person component describes greater person variation, not a defect in the assessment. A large rater component does not by itself identify which rater should receive feedback.

Negative estimates remain to the left of zero as open circles; nonnegative estimates use filled circles. An unconstrained negative estimate is not a negative population variance. Inspect the data/design and component diagnostics before interpreting it. No component is clipped, normalized or removed. A covariance matrix can be non-positive-semidefinite even when its projection for the selected score is positive; this is retained in the returned table and notes.

With `intervals = TRUE`, horizontal bars replace the stems from zero. They show the nonnegative-space intervals from `confint.mfrm_multivariate_gstudy()`: approximate Ting MLS intervals for components and chi-square intervals for the combined residual. They are pointwise, not simultaneous across components or scores. Negative point estimates can lie outside their nonnegative-space intervals. Square outlines mark estimates without an interval; consult `IntervalStatus` and `unavailable_intervals` for the reason. An empty interval or degenerate mean squares do not mean zero uncertainty. Raw endpoints, SEs and the full interval tables remain in the returned data. These intervals concern source variances, not future mean-score errors, G/Phi or off-diagonal covariance elements. With `intervals = FALSE`, stems run from zero to the estimate and do not represent uncertainty.

With `method = "reml_profile"`, the bars instead show approximate restricted-likelihood component intervals. The points remain the original ANOVA/MINQUE estimates; the separate `IntervalCenter` is retained in the table. Profile `SE` and inapplicable MLS `RawLower/RawUpper` columns are `NA`; `SEStatus` explains this. Numerical availability does not establish coverage, especially near zero variances. The interval surrounds the REML center but does not enumerate remote disconnected accepted regions.

With `method = "residual_contrast"`, only Residual receives a bar, based on a projection that removes all nonresidual effects. Points still show original ANOVA/MINQUE components; `IntervalCenter` records the projected residual estimate. No remaining residual degrees of freedom or numerically unresolved residual variation means unavailable limits. Exact variance coverage and conservative covariance coverage require the normal residual model at a fixed observed assignment; they do not extend to other sources or nonrandom omissions. See *Residual contrasts* in the interval help.

Multiple-score component panels share axes. Compare their magnitudes only when the score units are comparable. Covariance heatmaps show the selected submatrix for every source, with variances on the diagonal and covariances off the diagonal. Covariance units are the product of the two score units; these are not correlations. All panels share a color scale centered at zero and include signed numeric labels. Color presets use a diverging palette; `"monochrome"` uses a gray scale. Values are neither standardized nor converted to correlations. A negative covariance can be legitimate; a negative variance is an unconstrained estimate, not a population variance. `ComponentPSD` and returned notes describe the full source matrix even if only a subset of scores is shown. A large covariance alone does not diagnose halo. Cell labels are rounded for display; the table keeps exact values.

With `type = "covariance", intervals = TRUE`, select exactly two scores and an explicit method.

A single horizontal interval plot across sources replaces the heatmaps. Covariance endpoints remain signed, including negative values. "gci" has approximate coverage and separate Monte Carlo endpoint bands; "regression_region" has conservative normal-model coverage but may be very wide. Square outlines retain unavailable intervals. SEStatus is separate: a one-df block can prevent an SE while still permitting a conservative interval. Open circles mark negative covariances, which can be legitimate. Explicit "reml_profile" instead fits a joint normal model, also for incomplete sources, and retains its separate IntervalCenter and full numerical diagnostics. Its chi-square calibration is experimental and can under-cover with few facet levels; no sampling SE is supplied. None of these methods is a correlation interval.

All estimated interactions are shown. The table preserves the original Source identifiers and adds a readable Label. For a crossed Person, Rater and Task design, the last label is "Person x Rater x Task + residual". Its Source remains Residual: with one observation per cell, the highest-order interaction and other residual variation cannot be estimated separately. A one-facet design instead shows "Person x Task + residual" (or the declared facet name); a nested design shows, for example, "Person x Rater(Task) + residual". No extra independent error component should be added to these combined estimates.

Measurement error is not synonymous with the component called Residual. For the supported random-facet mean-score projections, relative error includes the person-by-condition interactions and combined residual; absolute error additionally includes condition main effects and interactions without Person. The D-study divides each component by its applicable future counts, then adds these terms. Use plot(d, type = "sem") to view the square roots of the two error variances. Neither SEM nor a residual-component point is a sampling confidence interval. See mfrm_d_study() for the formulas and assumptions, and vignette("mfrmr-gtheory") for a worked source-to-error example.

This plot reads stored component estimates and preserves missing-data rules. Explicit REML intervals fit a separate model without replacing those points; other interval methods use the saved summaries. Titles and explanations are returned separately from the figure. Use plot_data(p, "notes") and keep them when sharing a figure. Both base R graphics and optional as_ggplot() conversion are supported.

Value

Invisibly, an mfrm_plot_data object with table (source, display label, selected estimate, sign status and full component-matrix PSD diagnostic), weights, score, kind, source_design, source_rows, axis labels and notes. plot, panels and ncol describe the view. Component tables include Score; covariance tables include Row, Column and Element (variance or covariance), with a shared fill_limits and heat_colors palette. With intervals, table also includes SE, Lower, Upper, RawLower, RawUpper, Level, Method, IntervalStatus and IntervalAvailable. intervals retains one complete confint() table per selected score or composite, with its assumptions and source context; unavailable_intervals retains rows without a drawable interval. Status still describes the sign of the point estimate. REML component tables additionally retain IntervalCenter, SEStatus, candidate bounds and Validation; full numerical profiles are retained in intervals. Saving the plot preserves them for as_ggplot() reuse. Covariance interval plots have kind = "Covariance", one forest panel, and the original pair in score. The full interval table retains covariance units, assumptions, numerical bands and SE/interval diagnostics. Save the result with saveRDS() for later as_ggplot() conversion, without recalculating intervals, or save the G-study to redraw it with base graphics.

See Also

mfrm_g_study(), plot.mfrm_multivariate_d_study()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_multivariate_gstudy", run.dontrun = TRUE)
## Not run:
tasks <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(tasks, c("V", "W"), rater = NULL)
p <- plot(g, weights = c(V = .5, W = .5), preset = "monochrome",
          intervals = TRUE, assumption = "normal")
plot_data(p, "table")
plot_data(p, "notes")
plot(g, score = c("V", "W"))
covariance_plot <- plot(g, type = "covariance")
plot_data(covariance_plot, "table")
if (requireNamespace("ggplot2", quietly = TRUE)) {
  as_ggplot(p)
}

## End(Not run)
```

---

plot.mfrm_recovery_simulation

Plot parameter-recovery simulation results

Description

Plot parameter-recovery simulation results

Usage

```
## S3 method for class 'mfrm_recovery_simulation'
plot(
  x,
  y = NULL,
  type = c("summary", "coverage", "errors", "scatter", "replications"),
 metric = c("rmse", "bias", "mae", "correlation", "coverage", "mcse_bias", "mcse_rmse",
    "raw_rmse", "raw_bias", "mean_se", "se_available"),
  parameter_type = NULL,
  facet = NULL,
  comparison = c("aligned", "unaligned"),
  draw = TRUE,
  ...
)
```

Arguments

x	Output from simulate_mfrm_recovery().
y	Reserved for S3 generic compatibility.
type	Plot route: "summary" draws a metric from summary(x)$recovery_summary, "coverage" draws 95% coverage by parameter group, "errors" draws row-level recovery-error distributions, "scatter" draws truth against estimated values, and "replications" summarizes run status.
metric	Summary metric used when type = "summary". Supported values are "rmse", "bias", "mae", "correlation", "coverage", "mcse_bias", "mcse_rmse", "raw_rmse", "raw_bias", "mean_se", and "se_available".
parameter_type	Optional parameter type filter, such as "person", "facet", "step", "slope", or "population".
facet	Optional facet filter.
comparison	Error/estimate scale for row-level routes. "aligned" uses EstimateAligned / ErrorAligned; "unaligned" uses Estimate / ErrorRaw on the same comparison scale.
draw	If TRUE, draw with base graphics. If FALSE, return an mfrm_plot_data object with reusable plot tables and metadata.
...	Reserved for generic compatibility.

Details

These plots are intended as simulation-review graphics. They do not replace the row-level x$recovery table or the ADEMP metadata; they make the main recovery estimands easier to inspect during model-development and design checks. Coverage is displayed only for groups with valid centered structural intervals, conditional on availability. Summary/coverage routes refresh aggregates from saved rows; their data$availability_table preserves all selected groups, including those omitted from drawing. Inspect counts, availability rates, MCSEs and CIReasons there and in summary(x). A wholly unavailable coverage plot gives an informative error; it does not draw missing coverage as zero. See simulate_mfrm_recovery() for the target and the treatment of legacy saved intervals. A zero mean-aligned bias is an identification identity, not evidence of unbiased estimation.

Value

An mfrm_plot_data object. When draw = TRUE, the object is returned invisibly after drawing.

See Also

simulate_mfrm_recovery(), summary()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_recovery_simulation", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
```

```
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-recovery.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
rec <- saved$result
plot(rec, type = "summary", metric = "rmse", draw = FALSE)

## End(Not run)
```

`plot.mfrm_slope_intervals` *Display saved GPCM intervals and bootstrap results*

Description

Plot the selected inference result without fitting or recomputing covariance. These plots describe discrimination or specified contrasts, not rater severity, agreement, quality or recommended scoring weights.

Usage

```
## S3 method for class 'mfrm_slope_intervals'
plot(
  x,
  title = "GPCM slope uncertainty",
  subtitle = paste(attr(x, "target"), attr(x, "method"), paste0(100 * attr(x, "level"),
   "%"), if (attr(x, "simultaneous") == "none") "pointwise" else "Bonferroni", sep =
     " | "),
  reference = NULL,
  draw = TRUE,
  ...,
  type = c("interval", "profile"),
  caption = NULL
)

## S3 method for class 'mfrm_gpcm_bootstrap'
plot(
  x,
  title = "GPCM bootstrap inference",
  subtitle = "Fitted-model parametric bootstrap; unresolved refits retained",
  draw = TRUE,
  ...
)
```

Arguments

x	Saved `confint.mfrm_fit()` output from `parm = "slopes"`, or `bootstrap_mfrm_gpcm()` output.

title, subtitle	Optional returned metadata; NULL removes it. The interval description defaults to the saved target, method, confidence level and any saved numerical or bootstrap cautions. Custom text overrides this default, while the diagnostics remain in the plot data and interval tables. Titles and notes are not drawn. Use `plot_data(p, "notes")` for descriptions and cautions, and `plot_data(p, "title")` for the suggested figure title.
reference	Optional finite numeric vertical reference. Default NULL; no quality threshold or scoring-policy cutoff is selected.
draw	TRUE draws the ggplot; FALSE returns it only.
...	For bootstrap slope results, passed to `confint()` to select its target and level. Otherwise unused.
type	For saved slope intervals, `"interval"` displays bounds; `"profile"` displays saved likelihood evaluations and the same-target Wald limits. The default is `"profile"` for saved profile calculations and `"interval"` otherwise. This never initiates profiling.
caption	For saved slope intervals, optional returned caption. NULL removes it. Two-family intervals state the distinct family references by default. Profile plots supply a default explanation of line styles and missing coordinates when omitted.

Details

Crosses retain unavailable intervals. Open circles retain intervals with infinite endpoints; the plot does not replace them with finite bounds. Incomplete intervals retain the computed endpoint as a tick, without a connecting interval line. A missing endpoint is not an infinite endpoint or a calibrated one-sided interval. Finite review-only bounds use a dashed line. The table retains `IntervalStatus`, `CIEligible`, `CIUse` (when saved), all bounds and failure reasons; returned notes explain the display. For a bootstrap LRT, a histogram shows resolved null statistics and a line the observed statistic. The returned notes count every unresolved replicate.

Use `apa_table()` for target-aware tables, `plot_data()` for plotted values, and `as_ggplot()` for customization. Attach one result or a named list with `mfrm_results(fit, intervals = list(slopes = ci, curves = curves), compute = "never")`. Named plot routes become `gpcm_slopes`, `gpcm_curves`, etc. Exact source matching is required; default fit diagnostics are preserved. `export_mfrm_results()` saves these tables and plots; replay reloads the saved RDS, preserving the selected methods without rerunning estimation. Markdown report summaries include the finite, unbounded and unavailable interval counts and the saved inference cautions/reasons.

For positive slope or ratio intervals spanning many orders of magnitude, use `as_ggplot(ci) + ggplot2::scale_x_log10()`. Finite bounds and estimates must be strictly positive. Keep a linear axis for signed differences. Changing the axis does not narrow the interval or make it more reliable.

Value

Invisibly a ggplot, retaining an `mfrm_plot_data` attribute for `plot_data()`. No fitting or resampling is performed.

Examples

```
# ci <- confint(fit, parm = "slopes", scale = "standardized", method = "sandwich")
```

```
# plot(ci, title = NULL)
# apa_table(ci)
# plot_data(ci, "table")
```

---

plot_data *Extract reusable data from an mfrmr plot object*

Description

plot_data() is a small accessor for users who want to build custom base-R, ggplot2, plotly, or table-based displays from mfrmr plot helpers. It accepts an existing mfrm_plot_data object, or any mfrmr object whose plot() method supports draw = FALSE. Use plot_data_components() first when you want to inspect which components are available before extracting one.

Titles, subtitles, and explanatory footnotes are omitted from figures by default. Use plot_data(p, "title") for the suggested title and plot_data(p, "notes") for a Type/Text table of descriptions and cautions. This works for returned base-graphics payloads and mfrmr-generated ggplots, including after saveRDS()/readRDS(). Printing an mfrm_plot_data object also displays its notes in the console. Axes, legends, panel headings and data/status symbols remain in the figure. Removing figure text does not remove numerical warnings or strengthen the interpretation of a result.

Usage

```
plot_data(
  x,
  component = NULL,
  type = NULL,
  ...,
  format = c("native", "long"),
  value_cols = NULL
)
```

Arguments

x	An mfrm_plot_data object, or a fitted/report/review object with a plot(..., draw = FALSE) method.
component	Optional single component name inside the reusable plot data. When NULL, the full plot-data list is returned.
type	Optional plot type passed to plot() when x is not already an mfrm_plot_data object.
...	Additional arguments passed to plot(..., draw = FALSE) when x is not already an mfrm_plot_data object.
format	"native" (default) returns the saved component unchanged. "long" reshapes a selected table or matrix without dropping missing cells.

value_cols — Explicit table columns to gather into Variable and Value. Other columns, and an added SourceRow, remain identifiers/metadata. Select compatible value types and comparable units; estimates, standard errors and interval bounds are distinct quantities, not replicate draws. Matrices instead yield Row, Column, RowLabel, ColumnLabel and Value, with indices retaining identity even if labels repeat. Already-long tables (including cluster-profile observations) need no reshaping.

Value

The full reusable plot-data list, or the selected component.

Custom distribution plots

A cluster profile's observations component contains one row per original entity and selected feature, with ID, Feature, Value, Cluster, Included and MissingReason. Missing values and omitted entities remain. Use Included to select the plotted population explicitly. table is the summary, not raw observations. Export either with write.csv; use the observations for custom box/violin/raincloud plots. Feature units are the original units, supplied by the study. A smooth density is usually inappropriate for a few ordinal bins. vignette("mfrmr-networks-and-reporting") demonstrates custom plots and CSVs.

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Draw the Wright map and keep its reusable data
wright <- plot(fit)
plot_data(wright, "notes")

# Extract the plotted locations as a table
locations <- plot_data(wright, component = "locations")
head(locations)

# For table extraction alone, no graphics device is needed
locations_only <- plot_data(fit, component = "locations")
head(locations_only)
```

---

plot_data_components *List reusable components in mfrmr plot data*

Description

plot_data_components() is a companion to plot_data(). It returns a compact table that tells users which plot-data components are available, what shape they have, and which ones are most useful for custom graphics, dashboards, or report assembly.

Usage

```
plot_data_components(x, type = NULL, ...)
```

Arguments

x	An mfrm_plot_data object, or a fitted/report/review object with a plot(..., draw = FALSE) method.
type	Optional plot type passed to plot() when x is not already an mfrm_plot_data object.
...	Additional arguments passed to plot(..., draw = FALSE) when x is not already an mfrm_plot_data object.

Value

A data frame with one row per reusable plot-data component.

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Discover which tables the default Wright map provides without drawing it
plot_data_components(fit)

# Extract one of the listed components
locations <- plot_data(fit, component = "locations")
head(locations)

# A different plot type has different reusable tables
plot_data_components(fit, type = "ccc")
```

---

pool_mfrm_imputed *Pool common facet targets across imputed rating analyses*

Description

Combine results such as rater severity or a prespecified difference between raters across completed-data fits. Rubin's rules include uncertainty within each fit and variation between imputations. The calculation uses the full covariance for one non-Person facet; it does not pool Person ability scores.

Usage

```
pool_mfrm_imputed(
  x,
  facet,
  contrasts = NULL,
  ci_level = 0.95,
  df_complete = Inf
)

## S3 method for class 'mfrm_pooled'
print(x, ...)

## S3 method for class 'mfrm_pooled'
summary(object, ...)

## S3 method for class 'mfrm_pooled'
plot(
  x,
  draw = TRUE,
  preset = "standard",
  title = paste("Pooled", x$settings$facet, "estimates and contrasts"),
  subtitle = sprintf("%d imputations | %.0f%% pointwise MI intervals",
    x$settings$imputations, 100 * x$settings$ci_level),
  ...
)
```

Arguments

x	For `pool_mfrm_imputed()`: Output from `fit_mfrm_imputed()`. All completions must have inference-ready RSM/PCM MML fits and unregularized observed information. For `plot()`: Output from `pool_mfrm_imputed()`.
facet	A non-person facet column name, for example `"Rater"`.
contrasts	Optional numeric matrix with one named row per target and columns named by all facet levels. For example, a row with coefficients `c(1, -1, 0)` estimates the first rater minus the second. Without it, each facet level is reported. Columns are aligned by level, not position.
ci_level	Confidence level, strictly between zero and one.
df_complete	Complete-data degrees of freedom, a positive number or `Inf` (default). `Inf` uses the large-sample complete-data approximation. A finite value applies the Barnard–Rubin adjustment. No residual degrees of freedom are inferred from rating-row counts: ratings within a person are not independent sampling units. A supplied finite value requires a defensible complete-data reference distribution for the chosen target.
...	Unused.
object	An object returned by `pool_mfrm_imputed()`.

draw	Logical. With FALSE, return the exact plot data without opening a graphics device.
preset	A package plotting preset, such as "standard" or "monochrome".
title, subtitle	Optional text; NULL omits it. The default subtitle retains a notice when complete-data information is weak. Full cautions remain in the returned plot data even with custom or omitted text.

Details

For a common target, Qbar is the mean completed-data estimate, Ubar the mean complete-data covariance and B their between-imputation covariance. The total is T = Ubar + (1 + 1/m) B. The scalar t-reference degrees of freedom are (m - 1) / lambda^2, with lambda = (1 + 1/m) B / T, before any finite complete-data adjustment. A zero between-imputation variance has infinite large-sample degrees of freedom. Full covariances are transformed before scalar inference, so rater differences include covariance between the two estimates. The finite-data adjustment uses $\nu_{obs} = \nu_{com}(\nu_{com}+1)/(\nu_{com}+3)\,\bar{U}/T$ and $1/\nu = 1/\nu_{old} + 1/\nu_{obs}$. These t references are approximations, not exact finite-sample pivots for an MFRM. Positive interval widths that cannot be distinguished from zero at the returned estimate, and endpoints outside floating-point range, are withheld with a reason rather than reported as zero uncertainty or infinite bounds.

All fits must share categories, facet levels, signs, anchors, interactions and identification. Known anchors are fixed and their uncertainty is excluded. Singular or regularized free-parameter information and any failed/ineligible completion prevent pooling. A target fixed by a constraint is labelled "fixed"; it is not evidence of perfect precision. A verified unregularized inverse of ill-conditioned information can be used with a warning identifying the affected imputations. Rubin pooling does not repair unreliable complete-data approximations. Review interval widths, boundary proximity and quadrature sensitivity before interpreting the pool.

The intervals are pointwise model-based multiple-imputation intervals, conditional on adequate proper imputations and complete-data inference. They are not robust intervals, simultaneous rater decisions or general coverage guarantees. Do not pool EAPs, posterior SDs, fit statistics, cluster labels or likelihood-ratio tests with this function. See the executable assigned-score example in vignette("mfrmr-response-imputation"). It compares joint RSM predictive completions with direct observed-score inference. Proper calibration priors in the imputer and MML estimates in the completed-data analyses do not give exactly identical Bayesian moments: that interpretation is a large-sample approximation requiring review. The finite-imputation correction and MonteCarloSE assume independent completed-data draws. Repeating completions is not additional Monte Carlo evidence; correlated MCMC draws require sampling review. Identical completed values can occur legitimately and are not automatically rejected.

Congeniality concerns the imputation distribution and the complete-data analysis together, including its variance estimator. Preserving categories or including coefficient uncertainty in an ordinal imputer does not prove congeniality with an adjacent-category MFRM. Under incompatibility or model misspecification, Rubin intervals can be too narrow or too wide. Bootstrapping only the imputation-model training data, then completing and analyzing the original roster, is not bootstrap inference for the whole MI analysis. Bartlett and Hughes (2020) study the latter; their results require conditions including a consistent point estimator and appropriate resampling units. This function implements Rubin pooling, not that outer bootstrap procedure.

plot() method: Plot pooled facet estimates or prespecified contrasts

For `plot()`: Plots use the stored pointwise multiple-imputation intervals and preserve target order. Open diamonds identify fixed targets without an inferential interval. A zero reference line does not define a practical importance threshold. The plot does not refit, select significant raters, adjust for multiplicity or turn severity differences into rater quality.

Value

An `mfrm_pooled` object containing `table`, `within`, `between` and `total` covariance matrices, `estimates` by imputation, each `complete_covariance`, the exact `contrasts`, `facet_levels`, `settings` and source `analyses`. `MonteCarloSE` is the estimated Monte Carlo SE of the pooled point estimate, `sqrt(B / m)`; it is not its inferential SE. `MissingVarianceFraction` is `(1 + 1/m) B / T`, not the raw missing-rate. Fixed targets retain their value and zero variance, but no inferential interval or degrees of freedom. No imputation is omitted. `information_review` retains any numerical-refinement review at each imputation's list position. `cautions` identifies affected imputations; when present, `InferenceCaution` also accompanies the pooled table. `IntervalReason` distinguishes available intervals, fixed targets, numerical resolution loss and intervals outside the numeric range. An unavailable interval retains its point estimate and variance. `notes` records the imputation and inference assumptions for saved reuse.

For `plot()`: Invisibly, an `mfrm_plot_data` object containing the plotted `table`, exact contrast coefficients and inference settings. Use `plot_data()` for custom graphics. `as_ggplot()` uses the saved pointwise MI interval endpoints; it does not substitute normal intervals from the standard errors. Default and `component = "table"` retain the complete interval view. Target order, contrasts, degrees of freedom, confidence level and complete-data information cautions remain attached through `plot_data()`. Numerical interval-availability reasons and saved inference notes are retained. Both base and ggplot views show a cross for a missing interval. Fixed targets have open diamonds without intervals. In ggplot conversion, infinite endpoints have arrows (their tips are display limits, not finite bounds); missing intervals have crosses at the saved estimate. Missing point estimates cause an error rather than being placed at zero. Use source `title`/`subtitle` arguments or `ggplot2::labs()` to change text; hiding headings does not remove the attached inference cautions.

Bounded evaluation

A joint-RSM imputation example was examined using 200 independent datasets, each analyzed with MAR and low-score-dependent MNAR missingness. The design had 80 Persons, three fixed raters, two criteria, scores 0–2, forty completions and a correctly specified known N(0,1) Person distribution. Under MAR, fixed-rater contrast coverage was 96.5 percent among 198 available nominal 95 percent intervals (95 percent Monte Carlo bounds 92.9–98.6). Two imputation posteriors missed the sampling-diagnostic threshold; counting them as unsuccessful gives 191/200 available-and-covered trials (95.5 percent). Under MNAR, coverage was 26.0 percent (Monte Carlo bounds 20.1–32.7) and bias was +0.574 logits, although both missing fractions were near 14 percent. The MAR imputer cannot correct selection depending on missing scores. These results concern that imputer, contrast and design, not arbitrary supplied completions or general coverage. See the assigned-score vignette for direct-MML/Bayesian comparisons, interval widths and failure accounting.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option,

then "standard". For example, preset = "standard" overrides a session set to "monochrome". Explicit preset = NULL retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of plot() for report bundles use the same option through .... Plots without a common preset argument, including extended-model plots with their own palette controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with as_ggplot() uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply preset explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with old <- options(mfrmr.plot_preset = "monochrome") followed by options(old). Use options(mfrmr.plot_preset = NULL) to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

References

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley. Barnard, J. and Rubin, D. B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika*, 86, 948–955. doi:10.1093/biomet/86.4.948.

Bartlett, J. W. and Hughes, R. A. (2020). Bootstrap inference for multiple imputation under uncongeniality and misspecification. *Statistical Methods in Medical Research*, 29, 3533–3546. doi:10.1177/0962280220932189.

See Also

mice::pool.scalar(), review_mfrm_imputations(), fit_mfrm_imputed()

pool_mfrm_imputed()

---

precision_review_report

Build a precision review report

Description

Build a precision review report

Usage

```
precision_review_report(fit, diagnostics = NULL)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional matching output from diagnose_mfrm(). Recompute it after refitting; mismatched or outdated readiness records are rejected.

Details

This helper summarizes how mfrmr derived SE, CI, and reliability values for the current run. It also includes a source-grounded fit/separation basis table so users can keep mean-square fit, ZSTD standardization, Rasch/FACETS-style separation, and package QC thresholds in distinct reporting categories.

Value

A named list with:

- profile: one-row precision overview
- checks: package-native precision review checks
- fit_separation_basis: source-grounded fit/separation reporting boundary
- approximation_notes: detailed method notes
- settings: resolved model and method labels

What this review means

precision_review_report() is a structured prerequisite review for precision claims. It tells you how the package derived uncertainty summaries for the current run and how cautiously those summaries should be written up.

What this review does not justify

- It does not, by itself, validate the measurement model or substantive conclusions.
- A favorable precision tier does not override convergence, fit, linking, or design problems elsewhere in the analysis.
- Fit and separation rows in this report are reporting/validation boundaries, not standalone success criteria.

Interpreting output

- profile: one-row overview of the active precision tier and recommended use.
- checks: package-native review checks for SE ordering, reliability ordering, coverage of sample/population summaries, and SE source labels.
- fit_separation_basis: source-grounded boundary table for fit and separation reporting.
- approximation_notes: method notes copied from diagnose_mfrm().

Recommended next step

Use the `profile$PrecisionTier` and `checks` table to decide whether SE, CI, and reliability language can be phrased as model-based, should be qualified as hybrid, or should remain exploratory in the final report.

Typical workflow

1. Run `diagnose_mfrm()` for the fitted model.
2. Build `precision_review_report(fit, diagnostics = diag)`.
3. Use `summary()` to see whether the run supports model-based reporting language or should remain in exploratory/screening mode.

Regularized or fallback SEs remain diagnostic only. Numerical convergence does not establish inferential support. Older reports without the recorded regularization distinction must be recreated with `precision_review_report(fit)`; the existing fit can be reused without refitting.

See Also

`diagnose_mfrm()`, `facet_statistics_report()`, `reporting_checklist()`

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Review support for standard errors, intervals, and reliability summaries
diagnostics <- example_fit$diagnostics
precision <- precision_review_report(fit, diagnostics = diagnostics)
review <- summary(precision)
review$checks # Check statuses and reasons before making precision claims
```

---

predict.mfrm_random_rater

Predict category probabilities for observed or replacement raters

---

Description

Evaluate a shared-rater RSM at explicitly supplied ability values.

Usage

```
## S3 method for class 'mfrm_random_rater'
predict(
  object,
  newdata,
  ability,
  rater = c("observed", "new"),
  quad_points = 41L,
  ...
)
```

Arguments

object	A result from fit_mfrm_random_rater().
newdata	Data frame with the fitted rater column and every fixed-facet column. Observed-rater labels must be known; replacement-rater labels must be new. Fixed-facet levels must always be known.
ability	A finite numeric value for all rows, one value per row, or the name of a numeric column of newdata. These are specified abilities on the fitted logit scale (mean ability zero, Rasch slope one), not estimated person scores. A value of one is one logit, not one population SD.
rater	"observed" uses the approximate conditional distribution of an observed rater given the calibration responses. "new" integrates a replacement rater from the estimated population distribution. The target is explicit; an unfamiliar label is never silently treated as observed.
quad_points	Normal quadrature points for the prediction integral, from 7 to 121; default 41. Compare orders when needed.
...	Unused.

Details

Predictions condition on estimated calibration and supplied ability. They do not propagate calibration-estimation uncertainty or uncertainty about ability. Observed-rater integration uses the conditional Laplace mean/mode and covariance, not the calibration-adjusted prediction SE. New-rater integration uses the population SD: substituting severity zero generally produces different probabilities.

Rows are marginal probabilities for individual future ratings, not a joint distribution or an interval for their average. Ratings with the same new rater ID share one random severity in the model; multiplying these row probabilities would discard that dependence. This function does not score latent abilities from responses or impute missing assigned scores. Use score_mfrm_random_rater() for conditional Person scoring from a complete joint response roster. Supplying a score column in newdata does not update rater distributions or estimate abilities. Use the scoring function and its complete-roster rules when the input is a new batch of observed responses.

Value

A list with probabilities (rows by categories), expected_scores, newdata, ability, settings and matching source metadata for mfrm_results(). No model is refitted. Save this list together

with the fitted model and prediction inputs. The settings record the effect-sharing unit and whether integration uses the observed rater's conditional distribution or the replacement-rater population. These are preserved in the prediction settings table of `mfrm_results()`.

`predict.mfrm_testlet` *Score Persons using a fixed testlet calibration*

Description

Recompute Person scores from all supplied rows, integrating local effects within each Person/testlet block and conditioning on fitted calibration.

Usage

```
## S3 method for class 'mfrm_testlet'
predict(
  object,
  newdata = NULL,
  level = 0.95,
  quad_points = object$settings$quad_points,
  missing = if (is.null(newdata)) object$settings$missing else "fail",
  persons = NULL,
  ...
)

## S3 method for class 'mfrm_testlet_scores'
summary(object, ...)

## S3 method for class 'mfrm_testlet_scores'
print(x, ...)

## S3 method for class 'mfrm_testlet_scores'
plot(
  x,
  draw = TRUE,
  style = c("interval", "precision", "distribution"),
  sort = c("input", "estimate", "uncertainty"),
  palette = c("accessible", "mono"),
  title = NULL,
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  reference = 0,
  text_scale = 1,
  point_size = 2.5,
  ...
)
```

Arguments

object	A ready fit_mfrm_testlet() result.
newdata	Long-format ratings with the fitted column names. NULL reuses the source assignment, including missing-score rows. New Person and testlet labels are allowed; every fixed-facet level must be known. If the testlet column is also a fixed facet, its labels must therefore be known. For example, with testlet = "Rater" and "Rater" in facets, this function cannot score a new rater whose fixed severity was not fitted.
level	Conditional equal-tail posterior interval probability; default 0.95. These are not calibration-adjusted frequentist confidence intervals.
quad_points	Local-effect quadrature order, 7 to 121. Default uses the fitting order; each Person is also checked at 2 * quad_points + 1.
missing	"fail" or "omit". Default reuses the fit's missing-score policy for source replay and requires explicit omission for new data.
persons	Distinct Person IDs to return, or NULL for all. This selects outputs while retaining the complete supplied scoring roster.
...	Unused.
x	For predict.mfrm_testlet(): A scoring result. For plot(): A result from predict.mfrm_testlet().
draw	FALSE returns plot data without opening a device.
style	"interval" keeps every labeled row; "precision" plots estimates against interval width (upper minus lower bound); "distribution" shows the empirical cumulative distribution of finite point estimates, excluding prior-only and unavailable scores. No density is reconstructed.
sort	Row order for interval plots: input order, increasing estimate, or increasing interval width. Missing values come last; ties retain input order. Sorting is descriptive, not a test of differences.
palette	"accessible" uses blue and dark orange with filled/open symbols; "mono" uses dark grey with the same symbol distinctions.
title, caption	NULL uses the default text; a string replaces it; "" removes it. Interpretation notes remain in the saved plot data.
show_title, show_notes	Show the title and caption, respectively. FALSE hides text without deleting its metadata.
show_labels	Show IDs on interval and precision plots. Distribution plots summarize estimates and do not label individual IDs. For crowded precision plots use FALSE and consult the retained table.
reference	Vertical reference line in logits; NULL omits it. This is an orientation aid, not a quality threshold.
text_scale	Positive multiplier for text size.
point_size	Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

Details

All supplied rows for a Person are scored jointly. This does not append to cached responses or condition on a stored local-effect mode. The saved settings retain this integration, effect-sharing and roster interpretation. `mfrm_results()` retains the complete `scoring_data` as its `scoring_roster` table, including explicitly omitted scores, alongside observed block counts. Column names and row order are preserved. To add ratings for an existing Person, supply that Person's complete set of ratings once, with memberships that correctly identify effects shared within that set. Scores from separate calls are not automatically linked within a Person or testlet. No fixed effects are re-estimated. Unequal block sizes do not imply equal block weights. The influence of an additional response depends on its block and the fitted model; integrating local effects is not a general correction for assignment or content bias. See `vignette("mfrmr-testlet-applications")` for a complete-roster comparison of one-point changes in five- versus two-criterion tasks.

EAP, posterior SD and intervals use continuous ability integration. Interval endpoints invert the continuous CDF rather than a finite quadrature-grid CDF. Local-effect integration is compared at two orders; errors or unresolved differences retain a row with unavailable scores and an explanation. An explicitly omitted, entirely missing Person returns the fitted or specified mean-zero normal ability population with status `"prior_only"`, not evidence of measured average ability.

Calibration is held fixed, including ability and testlet variances. If the fitted ability variance is zero, all requested Persons retain unavailable rows with a reason; no zero-width ability interval is reported. At an estimated zero variance this is scoring under that fitted submodel, not evidence that dependence is absent. Prior/posterior terminology refers to latent Person scoring; calibration remains frequentist MML. A 480-dataset same-source simulation estimating both variances, replayed after correcting numerical start selection, found Person-interval coverage of 94.4% with 120 Persons and 91.7% with 24 Persons under balanced positive local dependence; coverage was 93.3% in one sparse, unequal-block condition with 120 Persons. The original numerical failures and their repair remain separately documented. Accounting for dependence improved coverage relative to ordinary RSM, but did not consistently improve EAP mean squared error. See `vignette("mfrmr-testlets")` for the comparison, Monte Carlo uncertainty and numerical-selection correction. These intervals neither correct calibration-estimation uncertainty nor guarantee 95% coverage for each fixed ability. Person contrasts and simultaneous decisions are not provided by this route.

`plot()` method: Plot conditional Person scores from a testlet model

For `plot()`: In the default interval view, empty rows remain labeled. Open circles mark prior-only results; filled points are response-based conditional scores. The intervals exclude calibration-estimation uncertainty and are not tests of Person differences. `as_ggplot()` preserves all labeled rows, prior-only symbols and the conditional-interval note. `plot_data()` retains statuses and reasons for custom graphics or table export; conversion does not rerun scoring.

Value

An `mfrm_testlet_scores` object with one table row per requested Person, omitted-row accounting, block sizes, settings, calibration and prediction-source metadata for `mfrm_results()`. `summary()` reports all statuses; `plot()` returns `plot_data()` when `draw = FALSE`. Saved scoring results need no live optimizer. Keep `table$Status` and `table$Reason` with exported scores: `prior_only` describes the assumed population without observed responses for that Person; `unavailable` withholds the estimate, SD and bounds with a recorded cause.

For `plot()`: Invisibly, an `mfrm_plot_data` object retaining all requested Persons, their statuses, interval endpoints and settings; use `plot_data()`.

Choosing a view

For `plot()`:

Interval plots answer where estimates lie and how uncertain they are. Precision plots help identify estimates with wider intervals; width is neither a fit statistic nor a reliability coefficient. Finite intervals are required and excluded rows remain in `display_data` with reasons. Distribution plots summarize the selected point estimates, which are affected by shrinkage and the selected roster. They do not estimate the latent population distribution, show posterior densities, or imply independent observations.

Accessible and reusable output

For `plot()`:

Colours are supplemented by point shapes; open circles mean prior only. Use `palette = "mono"` for monochrome reproduction. `plot_data()` retains the complete source `table`, `display_data` with inclusion reasons, `alt_text`, interpretation `notes`, and display settings. Supply the text alternative and table alongside exported figures; image files alone do not automatically expose that information to screen readers. `as_ggplot()` preserves the view and display controls, supplies `labs(alt = ...)`, and allows further `labs()`, `theme()` and scale edits. Rendering does not fit, score or resample. Use the same controls through `plot(results, type = "scores", style = "precision")`.

See Also

`fit_mfrm_testlet()`, `plot.mfrm_testlet_scores()`, `mfrm_results()`

`plot.mfrm_testlet()`, `plot_data_components()`, `mfrmr_interval_guide()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("predict.mfrm_testlet", run.dontrun = TRUE)
## Not run:
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$testlet$fit
scores <- example$testlet$scores
# To recompute these conditional scores:
# ratings <- load_mfrmr_data("example_core")
# scores <- predict(fit, persons = as.character(unique(ratings$Person)[1:4]))
# Selecting outputs retains the complete source roster for model maps.
scores$table
plot(scores)

## End(Not run)
```

predict_mfrm_population *Forecast population-level MFRM operating characteristics for one future design*

Description

Forecast population-level MFRM operating characteristics for one future design

Usage

```
predict_mfrm_population(
  fit = NULL,
  sim_spec = NULL,
  n_person = NULL,
  n_rater = NULL,
  n_criterion = NULL,
  raters_per_person = NULL,
  design = NULL,
  reps = 50,
  fit_method = NULL,
  model = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  seed = NULL
)
```

Arguments

fit	Optional output from fit_mfrm() used to derive a fit-based simulation specification.
sim_spec	Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec(). Supply exactly one of fit or sim_spec.
n_person	Number of persons/respondents in the future design. Defaults to the value stored in the base simulation specification.
n_rater	Number of rater facet levels in the future design. Defaults to the value stored in the base simulation specification.
n_criterion	Number of criterion/item facet levels in the future design. Defaults to the value stored in the base simulation specification.
raters_per_person	Number of raters assigned to each person in the future design. Defaults to the value stored in the base simulation specification.

design	Optional named design override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). The nested named-facet form design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both design and the scalar count arguments.
reps	Number of replications used in the forecast simulation.
fit_method	Estimation method used inside the forecast simulation. When fit is supplied, defaults to that fit's estimation method; otherwise defaults to "MML".
model	Measurement model used when refitting the forecasted design. Defaults to the model recorded in the base simulation specification.
maxit	Maximum iterations passed to fit_mfrm() in each replication.
quad_points	Quadrature points for fit_method = "MML".
residual_pca	Residual PCA mode passed to diagnose_mfrm().
seed	Optional seed for reproducible replications.

Details

predict_mfrm_population() is a **scenario-level forecasting helper** built on top of evaluate_mfrm_design(). It is intended for questions such as:

- what separation/reliability would we expect if the next administration had 60 persons, 4 raters, and 2 ratings per person?
- how much Monte Carlo uncertainty remains around those expected summaries?

The function deliberately returns **aggregate operating characteristics** (for example mean separation, reliability, recovery RMSE, convergence rate) rather than future individual true values for one respondent or one rater.

If fit is supplied, the function first constructs a fit-derived parametric starting point with extract_mfrm_sim_spec() and then evaluates the requested future design under that explicit data-generating mechanism. This should be interpreted as a fit-based forecast under modeling assumptions, not as a guaranteed out-of-sample prediction. Generating parameters are held fixed at the values in that specification. Monte Carlo SEs describe simulation precision conditional on these values; they do not include source-fit uncertainty or uncertainty about whether the generating assumptions describe a future administration.

When that fit-derived or manually built simulation specification stores an active latent-regression population generator, the helper still operates at the **design / operating-characteristic** level. It repeatedly simulates person-level covariates and responses, refits the MML population-model branch, and summarizes the resulting facet-level behavior. This is distinct from the fitted-model posterior scoring provided by predict_mfrm_units().

GPCM forecasts are available with caveats through the same repeated simulation/refit design route used by evaluate_mfrm_design(). They summarize design-level operating characteristics under the supplied or fit-derived slope-aware specification; they do not validate operational scoring, diagnostic-screening or signal-detection rules, slope-recovery adequacy, or arbitrary-facet planning.

Value

An object of class `mfrm_population_prediction` with components:

- `design`: requested future design
- `forecast`: facet-level forecast table
- `overview`: run-level overview
- `simulation`: underlying `evaluate_mfrm_design()` result
- `fit_identity`: source-fit metadata for fit-derived forecasts; `NULL` for an explicitly supplied simulation specification
- `sim_spec`: simulation specification used for the forecast
- `facet_names`: public non-person facet names carried by the simulation specification
- `design_variable_aliases`: public aliases for `n_person`/`n_rater`/`n_criterion`/`raters_per_person`
- `design_descriptor`: role-based description of design variables carried from the underlying planning object
- `planning_scope`: explicit record of the supported planning contract, including a `facet_manifest`
- `planning_constraints`: explicit record of mutable/locked design variables
- `planning_schema`: structured planning metadata carrying the role table, supported boundary, mutability map, and facet manifest
- `gpcm_boundary`: `GPCM` caveat row when a `GPCM` forecast route is used
- `settings`: forecasting settings
- `ademp`: simulation-study metadata
- `notes`: interpretation notes

Interpreting output

- `forecast` contains facet-level expected summaries for the requested future design.
- `Mcse*` columns quantify Monte Carlo uncertainty from using a finite number of replications.
- `design_variable_aliases` and `design_descriptor` carry the same public naming metadata used by the underlying planning object. They rename the standard two non-person facet roles for presentation, but they do not turn the current planner into a fully arbitrary-facet simulator.
- If `sim_spec$population$active = TRUE`, the forecast summarizes repeated latent-regression MML refits under that stored person-level generator; it is still a scenario forecast rather than direct posterior scoring for one observed sample.
- `simulation` stores the full design-evaluation object in case you want to inspect replicate-level behavior.

What this does not justify

This helper does not produce definitive future person measures or rater severities for one concrete sample. It forecasts design-level behavior under the supplied or derived parametric assumptions.

References

The forecast is implemented as a one-scenario Monte Carlo / operating- characteristic study following the general guidance of Morris, White, and Crowther (2019) and the ADEMP-oriented reporting framework discussed by Siepe et al. (2024). In `mfrmr`, this function is a practical wrapper for future-design planning rather than a direct implementation of a published many-facet forecasting procedure.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

Saved-result reuse

Fit-derived forecasts must match the source fit when supplied to archive or replay helpers. Explicit `sim_spec` forecasts remain separate scenarios; attaching them does not establish that their assumptions match the fit. Fit-derived forecasts with a fixed `population_sd` are unavailable: the design-simulation refits do not preserve that variance constraint or an anchor-defined origin. An explicit `sim_spec` can describe a separate forecasting scenario, but does not reproduce those source-fit constraints. See mfrmr_compatibility_layer for migration and replay behavior.

See Also

`build_mfrm_sim_spec()`, `extract_mfrm_sim_spec()`, `evaluate_mfrm_design()`, summary.mfrm_population_prediction

Examples

```
# Long-running workflow; run explicitly when needed.
# example("predict_mfrm_population", run.dontrun = TRUE)
## Not run:
spec <- build_mfrm_sim_spec(
  n_person = 16,
  n_rater = 3,
  n_criterion = 2,
  raters_per_person = 2,
  assignment = "rotating"
)
pred <- predict_mfrm_population(
  sim_spec = spec,
  design = list(person = 18),
  reps = 1,
  maxit = 30,
  seed = 123
)
s_pred <- summary(pred)
s_pred$forecast[, c("Facet", "MeanSeparation", "McseSeparation")]

## End(Not run)
```

predict_mfrm_units *Score future or partially observed units under the fitted scoring basis*

Description

Score future or partially observed units under the fitted scoring basis

Usage

```
predict_mfrm_units(
  fit,
  new_data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
  interval_level = 0.95,
  scoring_quad_points = 31L,
  readiness_policy = c("error", "review"),
  n_draws = 0,
  seed = NULL,
  adaptive_quad_points = NULL,
  scoring_prior = NULL,
  cut_scores = NULL
)

## S3 method for class 'mfrm_unit_prediction'
summary(object, digits = 3, ...)
```

Arguments

fit	Output from fit_mfrm() estimated with method = "MML" or method = "JML". When fit uses the latent-regression MML branch (posterior_basis = "population_model"), score the target persons with the same background-variable contract via person_data.
new_data	Long-format data for the future or partially observed units to be scored.
person	Optional person column in new_data. Defaults to the person column recorded in fit.
facets	Optional facet-column mapping for new_data. Supply either an unnamed character vector in the calibrated facet order or a named vector whose names are the calibrated facet names and whose values are the column names in new_data.
score	Optional score column in new_data. Defaults to the score column recorded in fit.

`weight` Optional weight column in `new_data`. Defaults to the weight column recorded in `fit`, if any.

`person_data` Optional one-row-per-person data.frame with the background variables required by a latent-regression fit. Ignored for ordinary fitted-object posterior scoring. For intercept-only latent-regression fits (`population_formula = ~ 1`), `mfrmr` reconstructs the minimal one-row-per-person table internally from the scored person IDs. This is the scoring-time table for `new_data`, not the fit object's replay/export provenance table. For categorical background variables, supply values on the same coding scale used at fit time; the fitted factor levels and contrasts are reused when building the scoring design matrix. Fitted standardization and polynomial/spline bases are also reused; they are not re-estimated from the scoring cohort.

`person_id` Optional person-ID column in `person_data`. Defaults to `person` when that column exists, otherwise `"Person"` for the canonical scoring layout.

`population_policy` How missing background data are handled when `fit` uses the latent-regression branch. `"error"` (default) requires complete person-level covariates for all scored persons; `"omit"` drops scored persons lacking complete covariates and records that omission in `population_review`.

`interval_level` Posterior interval level returned in `Lower`/`Upper`.

`scoring_quad_points` Number of Gauss-Hermite nodes used only for this scoring call. It is independent of the quadrature order used while fitting `fit`; the default is 31 and values below 2 are refused. The fixed or adaptive integration mode is inherited from `fit`.

`readiness_policy` How a source fit that is not scoring-ready is handled. `"error"` (default) refuses scoring. `"review"` permits an explicitly review-only fitted-object calculation and labels the returned estimates and settings accordingly; it does not make the source fit ready. GPCM MML and RSM/PCM normal population models may pass the conditional-scoring checks described below even when their global inference audit remains incomplete. GPCM population models with background covariates still require explicit review. Invalid calibration parameters or inconsistent stored prior parameters are refused under either policy. Older population results without the new check records retain their review-only restriction.

`n_draws` Optional number of quadrature-grid posterior draws to return per scored person. Use 0 to skip draws.

`seed` Optional seed for reproducible posterior draws.

`adaptive_quad_points` Optional vector of at least two distinct integer orders >= 3, for example `c(31, 61)`. Adds `quadrature_review`, comparing a fixed-prior grid with grids centered and scaled to each Person's posterior. Calibration parameters and the prior are held fixed. Inspect both fixed/adaptive differences and movement between adaptive orders; this diagnostic does not replace estimates, intervals or draws. Every scoring call automatically adds reference orders and uses them to check numerical score accuracy, including ordinary RSM/PCM with the retained prior; see below.

scoring_prior	Optional list(mean = ..., sd = ...) specifying one normal scoring prior shared by all scored persons, on the unchanged calibration ability scale. sd is a standard deviation, not a variance. NULL (default) retains the MML scoring prior; JML uses a post-hoc standard-normal reference prior. Both values must be finite; SD and its representable variance must be positive. Overrides of population models with background covariates are not supported. A supplied prior does not refit the calibration, estimate a new population, or bypass source checks. Numerical scoring checks run under the supplied prior.
cut_scores	Optional finite, strictly increasing ability cuts with optional distinct names. Requires ordinary RSM/PCM and unit weights. A decisions table returns continuous probabilities above cuts and within bands (Lower < ability <= Upper), conditional on the point calibration and actual scoring prior. Calibration/population and cut-score estimation uncertainty are excluded. This is not a Person-difference test or a probability of passing a future administration. Cut labels alone do not establish CEFR alignment. Tiny tails have absolute, not relative, numerical accuracy. Use apa_table(x, which = "decisions") or export the saved table.
object	Output from predict_mfrm_units().
digits	Number of digits used in numeric summaries. Decision probabilities, cut boundaries and scoring priors are not rounded.
...	Reserved for generic compatibility.

Details

predict_mfrm_units() is the **individual-unit companion** to predict_mfrm_population(). It uses the fitted calibration and, when available, the fitted one-dimensional population model to score new or partially observed persons via Expected A Posteriori (EAP) summaries on a quadrature grid.

With the default scoring_prior = NULL, when the original fit uses ordinary method = "MML", the posterior summaries use that fitted MML calibration and a standard normal scoring prior, unless a population model was fitted. When the original fit uses the latent-regression MML branch, the scoring prior is the fitted conditional normal population model $\theta \mid x \sim N(x^\top \hat{\beta}, \hat{\sigma}^2)$, so the returned summaries are population-model-aware posterior EAP estimates. When the original fit uses method = "JML", mfrmr applies the fitted facet/step parameters with a standard normal reference prior on the quadrature grid, so the returned person scores remain fitted-object EAP summaries rather than direct JML estimates from the fitting step.

When the fitted population model is intercept-only (population_formula = ~ 1), predict_mfrm_units() still uses the fitted population-model basis, but it can reconstruct the minimal scored-person table internally because no background covariates are needed beyond the person IDs in new_data.

The current GPCM branch is included in this scoring layer, so fitted GPCM objects can be used for the same fitted-object posterior summaries. This does not imply that every downstream diagnostic or reporting helper has already been generalized to GPCM.

This is appropriate for questions such as:

- what posterior location/uncertainty do these partially observed new respondents have under the existing calibration?
- how uncertain are those scores, given the observed response pattern?

All non-person facet levels in new_data must already exist in the fitted calibration. The function does **not** recalibrate the model, update facet estimates, or treat overlapping person IDs as the same latent units from the training data. Person IDs in new_data are treated as labels for the rows being scored.

When n_draws > 0, the returned draws component contains discrete quadrature-grid posterior draws that can be used as approximate plausible values under the fitted scoring basis. They should be interpreted as posterior uncertainty summaries, not as deterministic future truth values.

For JML fits, this scoring stage is intentionally post hoc: mfrmr uses the fitted facet and step parameters from the joint-likelihood fit, then adds a standard normal reference prior by default (or the explicit scoring_prior) only for the scoring layer so that new or partially observed units can be summarized on a quadrature grid. This is a practical fitted-object EAP procedure, not a claim that the original JML fit itself estimated a population model. It does not reproduce the training Person joint maximum-likelihood estimates or provide ML/WLE scoring for new Persons. Posterior uncertainty conditions on the fitted calibration; it does not include calibration-parameter uncertainty.

summary() method: Summarize posterior unit scoring output

Value

An object of class mfrm_unit_prediction with components:

- estimates: posterior summaries by person
- decisions: optional cut-score and ability-band probabilities, with boundaries, labels, priors, availability and uncertainty basis
- draws: optional quadrature-grid posterior draws
- fit_identity: source calibration, fitted population coding and scoring status used to check reuse with fit-level exports
- row_review: row-level preparation review for new_data
- population_review: optional person-level omission review for latent-regression scoring
- quadrature_review, quadrature_overview: optional unrounded numerical integration comparison and its compact overview
- input_data: cleaned canonical scoring rows retained from new_data
- person_data: cleaned or supplied person-level background data used for latent-regression scoring; NULL otherwise
- settings: scoring settings
- notes: interpretation notes

For summary(): An object of class summary.mfrm_unit_prediction with:

- estimates: posterior summaries by person
- decisions: unrounded conditional probabilities, when requested
- row_review: row-preparation review
- population_review: optional person-level omission review for latent-regression scoring
- settings: scoring settings
- notes: interpretation notes

Experimental two-family scoring

Two-family GPCM MML supports new Persons with known non-Person facet levels, unit weights and the retained fixed N(0,1) prior. Both slope components, facet locations and the second owner's category steps remain fixed. New combinations of known levels are allowed under the fitted product model; this does not validate those combinations empirically. Repeated Person-facet combinations are rejected. Missing responses are omitted with a warning and row review; Persons without valid responses receive no prior-only score. Unknown levels or scores outside the original category coding are rejected. Prior overrides and person-level population inputs are not supported.

The saved specification, point estimates, objective and convergence must agree before either readiness policy permits scoring. A finer rule (normally `max(q + 10, 2 * q - 1)`) must change the source negative log likelihood per Person by at most 1e-6 and have maximum absolute gradient per Person at most 1e-6 for conditional source scoring. Source-readiness or refinement failures require explicit `readiness_policy = "review"`; invalid calibration records or failed convergence cannot be enabled by review. No covariance is required. Each new batch also receives the EAP/SD integration checks described below. These tolerances are numerical checks, not statistical validity thresholds.

EAP, posterior SD and continuous equal-tail intervals condition on the point calibration and fixed prior, excluding calibration uncertainty. Numerical checks do not complete the identification/boundary audits or establish sampling coverage, population transport or decision accuracy. Settings retain both owners, component slopes, locations, steps, category coding, observed calibration combinations and separate source/batch checks through saveRDS() and summary(). `extract_mfrm_calibration()` provides the corresponding portable format 6 route after passing source checks. Use `mfrm_results(fit, scores = scores)` to attach matching saved native or format-6 scores for reports and exports without rescoring. The saved source identity, calibration and scoring records must agree with the supplied fit; source/batch checks, omitted rows and review-only labels remain visible.

Conditional scoring and source checks

A finite calibration can define a person's posterior score without proving that the calibration is the unique global optimum. For one-family GPCM MML and models with an estimated RSM/PCM normal population, this function checks the current likelihood, gradient and unregularized joint information using the same local-solution machinery as `compare_mfrm()`. Detected identification deficiencies, inadequate categories and failed convergence do not pass. RSM/PCM regressions additionally require the consistent full-rank source, unit weights, direct MML and centered unanchored facets/steps used by `mfrm_population_intervals()`. Scoring reuses the fitted predictor coding and conditional residual variance; no population uncertainty is propagated. Boundary and identification audits labelled "not evaluated" remain so; they are not relabelled as completed by these local checks.

Shared-owner GPCM JML can also use conditional scoring with unit weights and no anchors/interactions. Fresh joint objective/gradient and unregularized full-curvature checks must pass; known Person/additive/slope boundary certificates are not overridden by finite optimizer coordinates. Incomplete global identification/boundary audits remain incomplete. Source curvature includes free Person parameters and uses a dense matrix; saved portable calibration avoids repeating it. No MML integration check or formal slope interval is inferred from this JML calculation.

One-family MML calibration checks also compare the retained solution at its fitting order and a higher order (normally `2 * q - 1`, at least `q + 10`; `q + 10` is used if the larger rule is not representable). They require NLL movement <= 1e-5, gradient movement <= 1e-4 and a higher-order gradient <=

1e-4, in addition to the existing local-solution criteria. These are numerical screening tolerances, not statistical coverage or global-existence guarantees.

For every native scoring route, after a calibration passes, the actual reported EAP and SD are compared with adaptive reference orders under the same calibration and scoring prior. The reference includes the scoring order (at least 3 for this comparison) and a higher order chosen by the same rule. EAP/SD discrepancies and changes in adaptive EAP, SD and log marginal likelihood must each be <= 1e-5. Increase `scoring_quad_points` if this check fails. `readiness_policy = "review"` retains and labels those scores for inspection instead of stopping. Nonfinite posterior calculations or invalid parameters cannot be enabled by review.

`settings$local_calibration_review` records the source checks and `settings$score_integration_review` records the per-Person numerical checks. New outputs set `settings$score_integration_required = TRUE`; their saved checks must remain with the scores. Older outputs without these checks can still be summarized, but are not retrospectively numerically qualified. `SourceScoringReady` concerns the calibration; `ScoreIntegrationReady` concerns the reported scores. `EstimateUse` identifies any scores requiring integration review. These records accompany summaries, draws and exports. The conditional route has `source_scoring_status = "conditional"` and does not change the fit's global `InferenceReady` or boundary/identification status. Information-matrix work respects `mfrmr.max_information_bytes`; unavailable checks require explicit review, rather than accepting a stale covariance.

For example, a rubric calibration that passes these checks can score new respondents while holding its estimated normal prior fixed. Passing does not establish that the next cohort has the same ability distribution. Calibration and prior estimation uncertainty remain excluded from posterior intervals, and no portable GPCM calibration artifact is created here.

Scoring a later cohort

By default the scoring prior is retained from the fitted model; this function does not estimate a new cohort's population mean or spread from `new_data`. For example, a calibration from an advanced class may not supply a suitable prior for a beginning class, even when both classes use the same rubric. With only a few ratings, scores can depend substantially on that prior. The number of ratings alone is not an information measure: their calibrated difficulties, discriminations and observed categories also matter. Inspect extreme response patterns separately; increasing the rating count does not guarantee less prior sensitivity for every person.

A narrower prior can yield narrower posterior intervals without any new rating evidence. This is greater certainty under a stronger assumption, not evidence that the assessment became more informative. Changing the prior while fixing calibration is a sensitivity analysis; estimating the calibration again changes a different part of the analysis. Neither action alone establishes coverage for a new cohort.

Use `scoring_prior = list(mean = 0.5, sd = 1.5)`, for example, to inspect another normal-prior assumption while retaining the calibration. Choose plausible values for your setting; these example numbers are not a default recommendation. Compare the same response rows under both priors. The supplied prior is labelled as an analyst assumption, even if its numbers equal the retained prior. It does not repair a poorly estimated calibration.

`PriorMean`/`PriorSD` describe the prior actually used for scoring; `RetainedPriorMean`/`RetainedPriorSD` describe the original scoring prior. `PriorSource` distinguishes `"retained"` from `"user_supplied"`. `RetainedPrior` distinguishes an estimated normal population from the standard normal reference (including post hoc JML scoring). These unrounded columns accompany estimates, posterior

draws and their summaries. settings$prior_comparison records both priors by person, and settings$retained_posterior_basis records the original basis. Do not edit fit$population: its consistency with the retained calibration is checked independently of scoring_prior. Saving a prediction with saveRDS() stores a result; it does not create a portable GPCM calibration for future scoring.

Interpreting output

- estimates contains posterior EAP summaries for each person in new_data.
- Lower and Upper are continuous equal-tail posterior interval bounds at the requested interval_level, computed by numerical CDF inversion. EAP, SD, and optional draws still use the selected quadrature rule. These intervals condition on the fitted calibration and scoring prior, including any estimated population coefficients and variance. They exclude uncertainty from estimating calibration or population parameters; they are not confidence intervals at each fixed true Person ability. Their interpretation also depends on the scoring prior: a new population with a different ability mean, spread or shape can have different coverage. More quadrature points check numerical approximation under the same prior; they do not establish that this prior matches the new population.
- SD is posterior uncertainty under the fitted scoring basis used for scoring.
- Estimate tables retain IntervalLevel without rounding, the prior form, per-Person PriorMean and PriorSD, calibration method, interval method and uncertainty interpretation. WeightedLikelihood indicates whether any response contribution for that Person was raised to a non-unit weight. Such weights do not by themselves establish equivalent independent ratings or frequentist coverage. Draw tables retain their discrete-grid basis and the same prior and calibration interpretation.
- draws, when requested, contains approximate plausible values on the fitted quadrature grid.
- population_review, when present, records whether scored persons were omitted because their background data were incomplete for a latent-regression fit.
- quadrature_review, when requested, retains unrounded per-Person fixed/adaptive log-marginal, EAP and posterior-SD differences, changes between adaptive orders, and computation status/reasons. summary() also supplies a compact quadrature_overview. A computed status does not certify accuracy; inspect the differences and any unavailable rows.

Re-summarize older results to recover recorded interval settings and readable notes without changing numerical scores. Prior means/SDs for older estimated-population results may be unavailable because those parameters were not retained in the result. Re-score with the existing fitted model to retain them; no calibration refit is needed. An unrecorded interval algorithm is labelled unavailable, not assumed to use continuous quantiles.

What this does not justify

This helper does not update the original calibration, estimate new non-person facet levels, or produce deterministic future person true values. It scores new response patterns under the fitted calibration and, when applicable, the fitted one-dimensional population model.

References

The posterior summaries follow the usual quadrature-based EAP scoring framework used in item response modeling under calibrated parameters (Bock & Mislevy, 1982, pp. 432-433; Bock & Aitkin,

1981). When `fit` uses the latent-regression branch, `mfrmr` scores under the fitted conditional normal population model in the general plausible-values spirit discussed by Mislevy (1991). Optional posterior draws are exposed as quadrature-grid plausible-value-style summaries for practical many-facet scoring rather than as a claim of full ConQuest numerical equivalence. When the source fit is `JML`, the same literature supports the quadrature-based scoring layer, but the standard normal prior is a package-level reference prior introduced for post hoc scoring rather than an estimated population distribution. The local JML objective, gradient and curvature cutoffs are package numerical criteria, not literature-derived guarantees of calibration accuracy or interval coverage. The scoring literature does not establish the completeness of the source boundary audit.

- Bock, R. D., & Mislevy, R. J. (1982). *Adaptive EAP estimation of ability in a microcomputer environment*. Applied Psychological Measurement, 6(4), 431-444. doi:10.1177/014662168200600405.
- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Mislevy, R. J. (1991). *Randomization-based inference about latent variables from complex samples*. Psychometrika, 56(2), 177-196.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176.

Corrected JML

An experimental shared-owner corrected GPCM calibration can score new Persons after fresh checks of its adjusted equation, full Jacobian and saved parameter identities. These checks never apply an unadjusted likelihood-stationarity requirement to the corrected root. Unit weights and known facet levels are required; unresolved roots cannot be enabled by `readiness_policy = "review"`. No model is refitted. The returned EAPs use the corrected point calibration and a separate normal reference prior, not the original profiled Person estimates. Continuous posterior intervals exclude calibration uncertainty and residual calibration bias. The correction order and estimator remain recorded in the score table and saved settings. `extract_mfrm_calibration()` provides the corresponding portable route.

Conditional ability decisions

Supply `cut_scores` on the calibration's ability scale to obtain `decisions`, with one row per Person and requested event. `Type = "above_cut"` gives $P(\theta > c \mid y, \widehat{\psi})$; `Type = "ability_band"` gives the probability between successive cuts, including the two outer bands. `Lower < ability <= Upper` defines every row. `Label` uses the supplied cut names (or `Cut1`, ...) and `Band1`, ... in increasing order. These events overlap: only the ability-band probabilities sum to one.

`PriorMean` and `PriorSD` record the actual Person-specific normal prior; `Status` and `Reason` retain scoring availability. `available_conditional` means the applicable numerical checks passed, not that the model or the cut's substantive meaning has been validated. `review_only` requires inspection before interpretation. Source-roster scoring may return `prior_only` for missing-only Persons; portable scoring instead retains them as `unavailable` with `NA` probabilities. Fitted-object scoring omits Persons with no retained rows; inspect its row and population reviews. A Person absent from the input is never invented.

Probabilities integrate the continuous posterior, rather than counting `sample_mfrm_plausible_values()` draws or applying a normal approximation to EAP/SD. They condition on point calibration, population/prior parameters and supplied cuts, as recorded in `Uncertainty`. They do not propagate

estimation error, compare two Persons or predict passing a future test. CEFR labels require externally justified cuts on this same scale. Very small tails have absolute, not relative, numerical accuracy; a returned zero or one is not a claim of impossibility or certainty.

Review the scoring population as well as the calibration. An estimated normal prior is treated as known during scoring; its sampling uncertainty can matter especially with a small calibration cohort. A larger calibration cohort can improve parameter precision without correcting a prior that mismatches a new population's mean, spread or shape. Passing numerical checks does not establish that probabilities are accurate for that new population. A fixed reference prior is not a universal remedy: justify it for the target population on the calibration scale. If a prior override is supported, comparing externally justified alternatives is a sensitivity analysis, not evidence that the alternative giving the preferred classification is valid. Population-averaged accuracy also does not guarantee accuracy near each cut, within every subgroup or for each individual Person.

Use `score_mfrm_persons()` for eligible ordinary RSM MML source Persons, `predict_mfrm_units()` for ordinary RSM/PCM MML or JML fitted-object scoring, and `score_mfrm_calibration()` for eligible continuous-posterior RSM/PCM artifacts. Each route retains its existing source and population restrictions. Unit weights are required; GPCM, shared-rater and testlet cut probabilities are not supported. JML uses a separate scoring prior, not a population estimated by JML. `NULL` leaves the existing output unchanged.

`summary(x)$decisions` and `apa_table()` with `which = "decisions"` reuse saved probabilities without rounding them. Fitted-object and portable score batches also support `build_summary_table_bundle()` and `export_summary_appendix()`. Save the complete result with `saveRDS()` to retain cuts, priors and review states for later reporting. Existing score plots display EAPs and intervals, not these event probabilities.

Saved-result reuse

Fit-level export and replay helpers check that supplied scores describe the same calibration, fitted population coding and scoring status. Recreate older scores from the saved fit and original scoring inputs/settings; the calibration does not need refitting. See mfrmr_compatibility_layer.

See Also

`score_mfrm_persons()`, `score_mfrm_calibration()`, `classification_table()`, `apa_table()`, `predict_mfrm_population()`, `fit_mfrm()`, summary.mfrm_unit_prediction

`predict_mfrm_units()` for an executable scoring and reporting example; `apa_table()`, `export_summary_appendix()`

Examples

```
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- fit_mfrm(
  toy[toy$Person %in% keep_people, , drop = FALSE],
  "Person", c("Rater", "Criterion"), "Score",
  method = "MML",
  quad_points = 61
)
raters <- unique(toy$Rater)[1:2]
criteria <- unique(toy$Criterion)[1:2]
new_units <- data.frame(
```

```
  Person = c("NEW01", "NEW01", "NEW02", "NEW02"),
  Rater = c(raters[1], raters[2], raters[1], raters[2]),
  Criterion = c(criteria[1], criteria[2], criteria[1], criteria[2]),
  Score = c(2, 3, 2, 4)
)
# Illustrative boundaries only: these labels do not establish CEFR levels.
cuts <- c(Basic = -0.5, Advanced = 0.8)
pred_units <- predict_mfrm_units(toy_fit, new_units,
  scoring_quad_points = 121, cut_scores = cuts)
summary(pred_units)$estimates[, c("Person", "Estimate", "Lower", "Upper")]
pred_units$decisions

# Optional prior-sensitivity comparison performs another scoring calculation.
# example("predict_mfrm_units", run.dontrun = TRUE)
## Not run:
apa_table(pred_units, which = "decisions")
# Compare assumptions on the same response rows, without refitting.
alternative <- predict_mfrm_units(
  toy_fit, new_units, scoring_prior = list(mean = 0.5, sd = 1.5),
  scoring_quad_points = 121, cut_scores = cuts
)
comparison <- dplyr::bind_rows(
  "Retained prior" = pred_units$estimates,
  "Alternative prior" = alternative$estimates, .id = "Scenario"
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(comparison, ggplot2::aes(Scenario, Estimate)) +
  ggplot2::geom_pointrange(ggplot2::aes(ymin = Lower, ymax = Upper)) +
  ggplot2::facet_wrap(~ Person) + ggplot2::theme_minimal() +
  ggplot2::labs(x = NULL, y = "Posterior EAP",
    caption = "95% posterior intervals conditional on calibration and each prior")
}
# Retain the complete result; summaries and tables do not score again.
saved <- tempfile(fileext = ".rds")
saveRDS(pred_units, saved)
apa_table(readRDS(saved), which = "decisions")
unlink(saved)

## End(Not run)
```

`print.mfrm_apa_text` *Print APA narrative text with preserved line breaks*

Description

Print APA narrative text with preserved line breaks

Usage

```
## S3 method for class 'mfrm_apa_text'
print(x, ...)
```

Arguments

x	Character text object from build_apa_outputs()$report_text.
...	Reserved for generic compatibility.

Details

Prints APA narrative text with preserved paragraph breaks using cat(). This is preferred over bare print() when you want readable multi-line report output in the console.

Value

The input object (invisibly).

Interpreting output

The printed text is the same content stored in build_apa_outputs(...)$report_text, but with explicit paragraph breaks.

Typical workflow

1. Generate apa <- build_apa_outputs(...).
2. Print readable narrative with apa$report_text.
3. Use summary(apa) to check completeness before manuscript use.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("print.mfrm_apa_text", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
apa <- build_apa_outputs(fit, diag)
apa$report_text

## End(Not run)
```

---

q3_statistic *mfrmr standardized/aggregated-residual Q3-style screen*

Description

Computes mfrmr's Q3-style screening index: the Pearson correlation of **standardized residuals**, after mean-aggregation to one residual per Person-by-level cell, between every pair of levels of a chosen facet. Large absolute values identify pairs whose residuals show stronger positive co-movement or negative counter-movement than the main-effects model expects. This is not the raw-residual Yen (1984) Q3 statistic.

Usage

```
q3_statistic(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  min_pairs = 5L,
  yen_threshold = 0.2,
  marais_threshold = 0.3,
  relative_offset = 0.2
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output. Computed on demand when omitted.
facet	Facet whose levels are paired (default "Rater").
min_pairs	Minimum number of persons with finite aggregated residuals at both levels required to retain a pair; a single integer of at least three. Unavailable pairs remain in the table with NA correlations and flags (mirrors plot(x, type = "local_dependence")).
yen_threshold	Legacy-named argument retained for compatibility. It applies the strict absolute heuristic |Q3-style| > threshold (default 0.20). Yen (1984) did not propose this as a general cutoff, and it is not calibrated for mfrmr's standardized/aggregated-residual index.
marais_threshold	Stricter community-convention threshold (default 0.30). Marais (2013, p. 121) reports this as a value "often considered" in the literature, not as her own recommendation; her actual recommendation is the relative comparison implemented by relative_offset.
relative_offset	Screening offset for the relative-flag rule |Q3 - mean(Q3)| > relative_offset (default 0.20). This is a simplified screening approximation of the relative comparison advocated by Marais (2013) and operationalized by Christensen et al. (2017) as Q3_* = Q3_max - mean(Q3). Christensen et al. (2017) demonstrate empirically that no single critical value is appropriate across designs and recommend a parametric bootstrap; the fixed 0.20 here is a screening default, not a substitute for that bootstrap.

Value

An object of class mfrm_q3 containing:

pairs A data frame with one row per unordered facet-level pair and columns Level1, Level2, Q3, N, AbsQ3, YenFlag, MaraisFlag, RelativeFlag, and a textual Interpretation summarising which thresholds were exceeded. N is the number of persons with finite aggregated residuals at both levels. Unavailable pairs have NA correlations and flags, with an explanation in Interpretation. YenFlag is a compatibility name and does not imply that Yen (1984) proposed the fixed default.

`summary` One-row tibble with `MeanQ3`, `MaxAbsQ3`, the three flagged-pair counts, `CandidatePairs`, `AvailablePairs`, and `UnavailablePairs`. Flagged counts are unavailable if no pair can be evaluated.

`thresholds` The thresholds used, for reproducibility. The `yen` name is retained for compatibility.

`facet` The facet whose levels were paired.

`notes` Residual aggregation and interpretation qualifications, retained with the saved result.

Save the complete result with `base::saveRDS()` and export `q3$pairs` or `q3$summary` with `utils::write.csv()`. There is no dedicated `plot(q3)` method. To obtain its companion matrix, use `plot(fit, type = "local_dependence", diagnostics = diagnostics, facet = facet, min_pairs = min_pairs, draw = FALSE)` with the same inputs. Save that plot payload separately for conversion with `as_ggplot()` without recalculation. The heatmap displays correlations; it does not apply the three flag rules.

Statistic definition and distinction from Yen (1984)

The name **Q3-style** is deliberate. This implementation differs from Yen's (1984) original Q3 definition in two respects that together affect threshold interpretation.

(1) Standardized vs raw residuals. Yen (1984, eqs. 7-8, p. 127) defines `Q3 = cor(d_i, d_j)` where `d_{ik} = u_{ik} - P_hat_{ik}` is the **raw** residual. mfrmr uses **standardized** residuals `Z = (u - P_hat) / sqrt(Var(u))` because that is what `diagnose_mfrm()` stores. Because model-based residual scale factors can vary across observations, the resulting pair correlations and their null distribution differ from raw-residual Q3.

(2) Mean-aggregation. When the facet being paired (e.g. `Rater`) has multiple residual rows per (Person, Level) cell because of additional facets in the design (e.g. multiple `Criterion` rows per Person-Rater cell), the standardized residuals are first **mean-aggregated to one value per (Person, Level) cell**, and the Pearson correlation is taken over those mean-aggregated residuals. Yen's original formulation takes the correlation directly over per-(Person, Item) residuals, without aggregation. Aggregation changes the analysis unit to persons. Each correlation is based on persons observed at both levels, so its magnitude and null distribution are not directly comparable with raw-row Q3.

For both reasons, treat the values returned here as a **screening summary** rather than a direct substitute for the published Q3 thresholds. A formal raw-residual Q3 procedure would require a separately implemented and validated design-specific bootstrap; mfrmr does not currently provide that procedure.

Dimensionality and network interpretation

Choose `facet` for the question: pairing Rater levels describes rater residual associations, not a count of rubric dimensions. Associations may reflect additional ability structure, shared performance effects, rater effects or assignment patterns. A thresholded correlation graph is not a partial-correlation network or a fitted residual-network model; its community count is not a dimensionality test. Missing pairs are not zero edges. See `analyze_residual_pca()` and `vignette("mfrmr-visual-diagnostics")` for complementary checks and the limits of exploratory network analysis.

References

- Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement, 8*(2), 125-145. doi:10.1177/014662168400800201

- Chen, W.-H., & Thissen, D. (1997). Local dependence indexes for item pairs using item response theory. *Journal of Educational and Behavioral Statistics, 22*(3), 265-289. doi:10.3102/10769986022003265
- Marais, I. (2013). Local dependence. In K. B. Christensen, S. Kreiner, & M. Mesbah (Eds.), *Rasch models in health* (pp. 111-130). London: ISTE / Wiley.
- Christensen, K. B., Makransky, G., & Horton, M. (2017). Critical values for Yen's Q3: Identification of local dependence in the Rasch model using residual correlations. *Applied Psychological Measurement, 41*(3), 178-194. doi:10.1177/0146621616677520
- Nason, K., & DeMars, C. (2025). Another look at Yen's Q3: Is .2 an appropriate cut-off? *Journal of Educational Measurement, 62*(2), 345-359. doi:10.1111/jedm.12432

See Also

plot(x, type = "local_dependence"), `diagnose_mfrm()`

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
q3 <- q3_statistic(fit)
q3$summary
# Look for: use MaxAbsQ3 and the pair table to rank follow-up. The default
#   0.20/0.30 absolute rules are uncalibrated heuristics for this Q3-style
#   index, not standalone local-independence tests.
head(q3$pairs)
```

---

rater_centrality_table

Describe rater centrality using RSM residuals and rater-specific PCM steps

---

Description

Place indirect centrality indices alongside their model basis, exposure and availability. These statistics describe fitted ratings; they do not classify raters or estimate a centrality parameter.

Usage

```
rater_centrality_table(fit = NULL, threshold_fit = NULL, rater_facet = "Rater")
```

Arguments

fit	Optional native additive RSM fit from fit_mfrm(), for residual versus expected-score correlations and ordinary Infit/Outfit.
threshold_fit	Optional native additive PCM fit with step_facet equal to rater_facet, for the spread of rater-specific step estimates. At least one of fit and threshold_fit must be supplied.
rater_facet	Name of the fitted non-Person rater facet.

Details

Eckes and Jin (2021, pp. 68-70, 77, 83-85) distinguish RSM residual-based indices from threshold SD under a rater-specific PCM. Accordingly, this function never assigns a shared RSM threshold SD to individual raters or substitutes a criterion-specific PCM for a rater-specific PCM. Supplying only one model leaves the other indices unavailable with an explicit reason. Both models must use the same retained response roster, including row multiplicities, category mapping, facet signs, centering and anchors. Row order may differ. Each model's estimator and readiness are retained.

Current support is ordinary JML or MML, unit observation weights, additive RSM/PCM and no facet shrinkage. Imported fits, interactions, generalized slope models, corrected JML, testlet and shared-random-rater models require their own definitions. There is no model fitting, quadrature or covariance estimation here. RSM response expectations are evaluated at saved point estimates using the same probability calculation as diagnose_mfrm(). MML uses saved Person EAP scores: these are plug-in expectations, not posterior-integrated or posterior-predictive expectations. Estimation and assignment effects can affect these indices even without true centrality.

ResidualExpectedCorrelation is the unweighted Pearson correlation of raw residuals (observed minus expected score) with expected scores within a rater. More negative correlations are consistent with scores compressed toward the middle. Standardized residuals are not used. A correlation requires at least two complete rows with nonconstant residuals and expected scores. Missing or nonfinite expectations, or a nonfinite primary Person/facet estimate involved in a row, make that rater's residual indices unavailable; incomplete subsets are not silently analyzed. Infit and Outfit additionally require strictly positive finite response variances in every row. Infit is sum(residual^2) / sum(variance); Outfit is mean(residual^2 / variance).

ThresholdSD is the sample SD across the complete set of a rater's m adjacent-category step estimates (divisor m - 1, explicitly chosen here; the paper does not specify the divisor). ThresholdVariance is its square. Larger spreads are consistent with centrality and smaller spreads with extremity. Steps are centered offsets in logits, not cumulative thresholds; adding the same rater severity/location to each step leaves their SD unchanged. At least two finite steps are required: a binary rubric cannot supply this spread. Steps are kept in category order. Disorder is reported, not repaired by sorting estimates; spread alone does not establish proper category functioning. ObservedSD describes the fitted internal score coding, with original-category labels retained in categories.

Infit/Outfit are supporting fit descriptions. Low Infit can also reflect halo, restricted score ranges or other mechanisms. No default .75, -.30 or 1.40 cutoff, combined flag, significance test or confidence interval is supplied. Threshold uncertainty, sparse exposure, different ability mixes, criterion mixtures and dependence between ratings affect interpretation. This function does not implement the Bayesian FM-SC centrality parameter omega, its uncertainty or ability adjustments, and does not reproduce FACETS numerical output merely by using related formulas. The package's ej2021 example data are synthetic, not the original TestDaF samples.

In the direct FM-SC model, rater severity shifts category locations while a positive rater-specific omega multiplies category step offsets. Omega therefore changes the response model and its ability estimates; it is not the SD of separately fitted PCM steps or a GPCM discrimination coefficient. Jin and Wang (2018) introduce this model; Eckes and Jin (2021, p. 71) give its RSM formulation. They are references for the distinction, not additional estimation methods available through this function.

Value

A list containing `table` (one row per rater), `thresholds` (category-ordered PCM step estimates), `categories` (counts on the retained roster), `sources` (model settings, estimates and readiness), `settings` and `notes`. Each index has an availability reason; source readiness is separate from numerical availability. Save with `base::saveRDS()` and reopen with `base::readRDS()`; export the data frames with `utils::write.csv()`. Reopening the result does not recalculate indices. This plain list has no dedicated `plot()` or automatic aggregate-report section. For a custom figure, retain each index's reason and source-status columns; do not mix threshold SD, residual correlation and fit on one scale. Graph centrality from `rater_halo_network_analysis()` has a different meaning.

References

Eckes, T., & Jin, K.-Y. (2021). Measuring rater centrality effects in writing assessment: A Bayesian facets modeling approach. *Psychological Test and Assessment Modeling, 63*(1), 65-94. Definitions and actual analyses: pp. 68-70 and 77; comparisons and limitations: pp. 83-88. Full text.

Jin, K.-Y., & Wang, W.-C. (2018). A new facets model for rater's centrality/extremity response style. *Journal of Educational Measurement, 55*(4), 543-563. doi:10.1111/jedm.12191. Direct model, not implemented here.

See Also

`score_comparison_table()`, `interrater_agreement_table()`, `diagnose_mfrm()`, `mfrm_report()`, mfrmr_methods_reference

Examples

```
ratings <- load_mfrmr_data("example_core")
rsm <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 100)
pcm <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "PCM", step_facet = "Rater",
                maxit = 100)
centrality <- rater_centrality_table(rsm, pcm, rater_facet = "Rater")
centrality$table
centrality$thresholds
# Review both SourceStatus columns before interpreting finite indices.
```

`rater_halo_network_analysis` *Analyze rater-by-criterion halo-effect networks*

Description

Analyze rater-by-criterion halo-effect networks

Usage

```
rater_halo_network_analysis(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  criterion_facet = NULL,
  context_facets = NULL,
  method = c("spearman", "pearson", "kendall"),
  min_pair_n = 5,
  alpha = 0.05,
  p_adjust = "bonferroni",
  min_abs_weight = 0,
  halo_weight_review = 0.5,
  halo_contrast_review = 0.1,
  min_retained_halo_edges = 1,
  positive_only = TRUE,
  include_graph = FALSE,
  bootstrap = 0L,
  conf_level = 0.95,
  seed = NULL,
  halo_reference = c("different_criteria", "all")
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`rater_facet`	Name of the rater-like facet.
`criterion_facet`	Name of the criterion, rubric, task, or item-like facet used to form rater-by-criterion nodes.
`context_facets`	Facets defining rows in the reshaped wide matrix. Defaults to the person facet plus any facets other than the rater and criterion facets. Include `"Person"` for the independent-Person association screen; without it, correlations remain descriptive.
`method`	Correlation method used for rater-by-criterion node pairs.

`min_pair_n` Minimum shared contexts required to estimate a node-pair relationship.

`alpha` Adjusted p-value threshold for retaining edges. Set to `1` to disable p-value filtering, including when inference is unavailable; shared-count, weight and sign filters still apply. This changes retained-edge counts, graph topology, and graph centrality, but not the full-pair halo/non-halo distribution summaries.

`p_adjust` Multiple-comparison adjustment passed to `stats::p.adjust()`. The default `"bonferroni"` follows the conservative screening used in Lamprianou's halo-network example.

`min_abs_weight` Minimum absolute correlation retained as a graph edge. Like `alpha`, this is an analysis filter for the retained graph, not merely a display filter.

`halo_weight_review` Same-rater cross-criterion mean absolute correlation at or above which a rater is marked for review. This is a package screening default, not a validated halo cut point.

`halo_contrast_review` Minimum difference between a rater's mean halo edge weight and incident non-halo edge weight for a stronger review flag. This is a package screening default, not a validated halo cut point.

`min_retained_halo_edges` Minimum retained halo edges required before a strong `"warning"` status is assigned.

`positive_only` If `TRUE`, negative correlations are kept in `pair_metrics` but excluded from the graph edge table and retained-edge counts. Unfiltered distribution and per-rater summaries use absolute correlations regardless of this setting; inspect `MeanSignedHaloWeight` before interpreting a review flag.

`include_graph` If `TRUE`, include the underlying `igraph` object.

`bootstrap` Number of person-cluster resamples for the halo contrast; `0` (default) requests no interval. At least 199 when requested; use 999 or more for reporting and assess Monte Carlo stability.

`conf_level` Confidence level for the pointwise percentile interval.

`seed` Optional seed, preserving the caller's RNG state.

`halo_reference` `"different_criteria"` compares within-rater cross-criterion correlations with between-rater cross-criterion correlations. `"all"` also includes same-criterion between-rater edges, matching the paper's descriptive all-non-halo reference.

Details

`rater_halo_network_analysis()` reshapes rating data so that each rater-by-criterion combination is a node. Edges are correlations between those node score vectors across shared contexts. Edges connecting two nodes from the same rater but different criteria are labelled `"halo"`; all other retained edges are labelled `"non_halo"`.

This response-profile network does not evaluate assignment connectedness or establish a common scale. Its degree, strength, betweenness, and closeness columns are graph-theoretic centrality summaries after adjusted-p, `min_abs_weight`, and sign filters are applied.

Per-rater ReviewStatus combines same-rater cross-criterion mean weight, incident non-halo comparison weight, and the number of retained halo edges. A "warning" means these criteria converge strongly enough to prioritize follow-up; "review" means at least one screening criterion is elevated. Neither label is a causal halo diagnosis, and the default review thresholds are package conventions rather than validated decision rules. Strong negative same-rater correlations are anomalous but are not evidence of a positive halo pattern; consult signed correlations in pair_metrics and MeanSignedHaloWeight.

The key descriptive comparison is the distribution of halo-edge weights versus non-halo-edge weights. A larger halo-edge distribution is consistent with a halo pattern, but this function deliberately reports it as a screening diagnostic. MeanHaloWeight and MeanNonHaloWeight use all finite absolute correlations meeting min_pair_n, before adjusted-p, min_abs_weight, and sign filtering; the corresponding MeanRetained* columns describe the filtered graph. Candidate and unavailable pair counts accompany these available-pair means. An incomplete comparison cannot receive an unqualified no-flag assessment.

WelchT, WelchDF and WelchP remain as unavailable compatibility columns. The correlations share observations, raters and nodes; an independent-sample test of their difference is not supplied. Pairwise correlation tail areas screen association under their sampling assumptions, not halo itself. For each pair, these tail areas require "Person" in context_facets and no repeated Person among its shared contexts. Otherwise the correlation is retained but PValue and PAdjusted are unavailable. N counts shared contexts, Persons counts distinct Persons when traceable, and PValueStatus explains availability. Repeating a Person across tasks does not create independent observations. No clustered test is substituted. Distinct identifiers do not establish independence: sampling dependence, fixed rater effects, ties and small-sample approximations still require review. Adjustments use the family of available pairwise p-values; counts of available and unavailable p-values are separate from correlation counts.

With alpha < 1, an unavailable p-value excludes that pair from the graph and prevents an unqualified "ok" review for an affected rater. alpha = 1 explicitly requests a descriptive graph without the p filter; it does not make the missing p-values available or validate review flags. No retained graph edges is not evidence that halo is absent; retain the originally specified thresholds. Recreate older network results from the existing fit and matching diagnostics before summary, plotting or export.

The node vectors contain observed scores, not model residuals. Repeated ratings within a context-by-node cell are averaged using observation weights; the selected correlation then compares available context means. Ability differences, related criteria and the assignment design can produce strong associations without a causal halo effect. Graph centrality here is distinct from middle-category use measured by rater_centrality_table().

A horn effect refers to an unfavorable impression spreading to ratings on other criteria. It is not a negative criterion correlation or simply high rater severity: lowering several criteria together can still produce positive correlations. Neither the sign of a network edge nor Infit/Outfit classifies halo versus horn; this function provides no such classification.

Direct halo models are a separate approach. Jin and Chiu (2022) introduce the MRFM-H mixture Rasch facets model; Jin and Eckes (2025) distinguish general-impression and inadequate-criterion-discrimination mechanisms in MRFM-H(GI) and MRFM-H(ID). These models are not implemented here. Network summaries do not estimate their latent classes or adjust measurements for those mechanisms. See mfrmr_methods_reference for choosing the target.

Save the complete result with base::saveRDS() before repeated plotting. plot(out, type = "edge_distribution") shows available absolute correlations before edge filters; type = "halo_summary" uses the selected

metric. type = "matrix" shows signed retained correlations on a -1 to 1 scale, with grey unavailable/filtered entries. These views support as_ggplot(); the network view uses base graphics or custom graphics from plot_data(). Plot payloads retain settings, caveats and denominators outside the figure. Export the result's data frames with utils::write.csv(); this standalone bundle is not an automatically included section of mfrm_report().

Value

A bundle of class mfrm_halo_network containing:

summary One-row halo-network summary and halo/non-halo contrast.

node_metrics Rater-by-criterion node strength and graph-theoretic centrality.

edge_metrics Retained graph edges.

pair_metrics All estimated node-pair correlations with explicit retention flags for each graph filter, shared-context and Person counts, and p-value availability reasons. RetainedByP is TRUE for every pair when the p filter is disabled by alpha = 1.

halo_summary_by_rater Per-rater summaries of same-rater criterion-pair edges, including ReviewStatus and ReviewReason.

caveats Interpretation notes.

Dependence-aware halo contrasts

halo_contrast is the unweighted mean of signed same-rater correlations minus the mean of the declared between-rater reference correlations. It uses the original available pair set before p-value/sign/edge filtering; it may therefore differ from the legacy retained-edge summary. Report the actual criterion/overlap mix in contrast_pairs; the default removes same-criterion reference edges but does not balance unequal criterion-pair counts.

Bootstrap resampling draws whole persons, keeping all their contexts, raters and criteria together, and recomputes every fixed pair. Repeated draws of a person remain repeated observations. This preserves within-person dependence and shared-edge sampling dependence. It assumes independent, representative person clusters and conditions on the observed raters, criteria and pair set. It does not generalize to a population of raters, account for clustering of persons within schools, establish causal halo, or validate missingness assumptions. A person context must be retained for resampling.

bootstrap_draws retains every attempted replicate and its status. If any required correlation becomes undefined (e.g. constant bootstrap scores), the interval is withheld rather than conditioning on successful replicates or silently changing the target edge set. Few persons/sparse overlap can make this method unsuitable. Percentile intervals are approximate, pointwise and not selection-adjusted. No independent-edge Welch p-value is supplied.

Relation to MFRM and published network analyses

These are observed-score relationships, not fitted residuals or adjusted Rasch parameters. Lamprianou (2025) compares agreement, directional and rater-by-criterion networks with MFRM; a close empirical correspondence is not numerical equivalence. Signed alpha edges and weighted directional SWS/PUS indices are available here; fit_rater_network() supplies binary ERGM estimation. Halo-edge associations can reflect trait covariance and rubric overlap; rater_halo_network_analysis()

offers person-cluster bootstrap contrasts instead of an independent-edge test. Use `mfrm_residual_network()` for residual associations and `subset_connectivity_report()` for prefit assignment links. See `vignette("mfrmr-networks-and-reporting")` for complementary views, paired overlap, direction and interpretation. `plot.mfrm_bundle()` documents custom layouts and display-only filters, which do not change the computed network.

References

Lai, E. R., Wolfe, E. W., & Vickers, D. (2015). Differentiation of illusory and true halo in writing scores. *Educational and Psychological Measurement, 75*(1), 102-125.

Lamprianou, I. (2025). Network Analysis for the investigation of rater effects in language assessment: A comparison of ChatGPT vs human raters. *Research Methods in Applied Linguistics, 4*, 100205.

Jin, K.-Y., & Chiu, M. M. (2022). A mixture Rasch facets model for rater's illusory halo effects. *Behavior Research Methods, 54*, 2750-2764. doi:10.3758/s13428021017213. Direct model, not implemented here.

Jin, K.-Y., & Eckes, T. (2025). When raters generalize: Examining sources of halo effects with mixture Rasch facets models. *Behavior Research Methods, 57*, Article 149. doi:10.3758/s13428-025026676. Direct models, not implemented here.

See Also

`rater_network_analysis()`, `interrater_agreement_table()`, `plot.mfrm_bundle()`

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
if (requireNamespace("igraph", quietly = TRUE)) {
  halo <- rater_halo_network_analysis(fit)
  halo$summary
  head(halo$halo_summary_by_rater)
  plot(halo, type = "edge_distribution", draw = FALSE)
}
```

`rater_network_analysis` *Analyze rater agreement, disagreement, and severity-direction networks*

Description

Analyze rater agreement, disagreement, and severity-direction networks

Usage

```
rater_network_analysis(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  context_facets = NULL,
  mode = c("agreement", "disagreement", "severity_direction"),
  weight_metric = NULL,
  min_pair_n = 1,
  min_weight = 0,
  score_diff_tolerance = 0,
  severity_continuity = 0.5,
  exact_warn = 0.5,
  corr_warn = 0.3,
  include_graph = FALSE,
  alpha_level = c("interval", "ordinal", "nominal", "ratio"),
  alpha_categories = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`rater_facet`	Name of the rater-like facet. If omitted, mfrmr uses the same heuristic as `interrater_agreement_table()`.
`context_facets`	Facets defining shared scoring contexts. By default, the person facet and all non-rater facets are used.
`mode`	Network definition. `"agreement"` builds an undirected network whose edge weights represent observed agreement. `"disagreement"` builds an undirected network whose edge weights represent observed disagreement. `"severity_direction"` builds a directed network: an edge from rater A to rater B means A assigned higher scores than B in shared contexts and is therefore relatively more lenient under the usual higher-score-is-better rating convention.
`weight_metric`	Pair-level weight used for `"agreement"` or `"disagreement"` networks. Defaults to `Exact` for agreement and `MAD` for disagreement. Available pair columns include `Exact`, `Adjacent`, `Corr`, `MAD`, `OneMinusExact`, and `AbsMeanDiff`. `KrippendorffAlpha` retains signed alpha edges in agreement mode. Directed mode accepts `DirectionalHigherProp` (default, count/N) or `DirectionalMagnitude` (sum of positive score differences/N, for Lamprianou's weighted indices).
`min_pair_n`	Minimum number of shared contexts required for a rater pair to contribute an edge.
`min_weight`	Minimum edge weight retained in the graph. This is an analysis threshold: it changes graph topology and all graph-derived centrality summaries, not only the displayed edges.
`score_diff_tolerance`	Score-difference tolerance for directed severity networks. With the default `0`, any higher score contributes to the outgoing leniency edge. This is an analysis

tolerance: increasing it changes directional counts, strengths, and SeverityIndex; it is not a plot-only filter.

severity_continuity
: Continuity constant added to incoming and outgoing strengths before computing the finite severity index -log((OutStrength + c) / (InStrength + c)). The default 0.5 is a package finite-value correction. Setting c = 0 gives the uncorrected published form but can produce non-finite values for zero strengths.

exact_warn, corr_warn
: Passed to interrater_agreement_table() to keep pair flags consistent with the tabular agreement view.

include_graph
: If TRUE, include the underlying igraph object in the returned bundle.

alpha_level, alpha_categories
: Measurement level and category order for KrippendorffAlpha; see krippendorff_alpha_table(). The default interval level follows Lamprianou (2025); ordinal alpha requires a declared order.

Details

This function implements a package-native rater-effect network view complementary to MFRM output. It follows the pairwise-network logic used in Lamprianou's rater-effect network work: nodes are raters, edges summarize pairwise relationships among raters in shared scoring contexts, and directed disagreement edges can be interpreted as relative leniency/severity indicators. These network summaries are descriptive diagnostics, not Rasch logit estimates and not formal fit statistics. They describe score relationships conditional on observed shared contexts; they do not test the assignment graph's connectedness or establish a common measurement scale.

Degree, Strength, Betweenness, and Closeness are graph-theoretic quantities computed after min_pair_n, min_weight, and (for directed networks) score_diff_tolerance are applied. They are not rating-scale central tendency or restriction-of-range measures. Use mfrm_network_analysis() for assignment/co-observation connectedness.

For mode = "severity_direction", outgoing strength means the rater more often assigned higher scores than comparison raters; incoming strength means comparison raters more often assigned higher scores than this rater. The reported SeverityIndex is positive for relatively severe raters and negative for relatively lenient raters, but it is on a network-analysis scale and should not be read as an MFRM severity logit. A rater without any retained directional comparisons has an unavailable index, not a balanced index. Except for signed alpha, zero or negative undirected weights remain in the pair table but do not form graph edges. Graph distances summarize reachable pairs only, so they do not describe distance across disconnected components. Recreate older network results from the existing fit and matching diagnostics with the original settings before summary, plotting or export.

Value

A bundle of class mfrm_rater_network containing:

summary One-row graph summary.

node_metrics Rater-level degree, strength, graph-theoretic centrality, and severity-direction summaries.

edge_metrics Retained rater-pair network edges.

`pair_metrics` All estimated pairwise agreement and directional comparison metrics, including `EligiblePair`, before `min_weight` filtering.

`caveats` Interpretation notes and sparse-design warnings.

`source_interrater` The underlying `interrater_agreement_table()` output used for agreement statistics.

Signed agreement and weighted network fit

Pairwise alpha uses only shared finite scoring contexts and pair-specific marginals. Negative alpha is retained, zero remains a non-edge, and undefined expected disagreement remains `NA` with `AlphaStatus`. Alpha is not a correlation. `Weight` is its magnitude for drawing; `SignedWeight` retains its sign. `Strength` is absolute strength; `PositiveStrength`, `NegativeStrength` (magnitude) and `SignedStrength` separate its meanings. Path centralities are withheld for signed alpha. These are pairwise coefficients, not the multi-rater alpha for the whole study. Repeated rater-context cells follow the existing weighted-mean score-profile aggregation. Shared contexts are then equally weighted; row weights are not treated as independent replicates or precision weights for alpha/SWS/PUS.

With `DirectionalMagnitude`, `network_fit` implements equations 1–3 of Lamprianou (2025). Let w_ij be the sum of positive score differences from i to j, T_ij the number of shared contexts (including ties), and S_i = sum_j(w_ij/T_ij) / sum_j(w_ji/T_ij). Severity is -log(S_i). SWS_i = sum_j(w_ij S_j/S_i + w_ji S_i/S_j) / sum_j(T_ij) - 1. PUS_i = mean_j(w_ij w_ji / T_ij^2), over compared partners including zero-disagreement partners. `network_fit` uses all eligible pairs before `min_weight` filtering, with no continuity correction. This differs from the filtered, optionally corrected `node_metrics$SeverityIndex`. `score_diff_tolerance` omits differences at or below the tolerance, retaining the full difference above it; zero is the unmodified definition. Report tolerance, eligible contexts, rubric range and units. Higher scores here mean numerically higher; interpret leniency only after checking rubric direction.

Zero incoming/outgoing strength can give infinite severity. SWS is unavailable when any required ratio is undefined or nonpositive; PUS can still be zero. Negative SWS values are preserved: the subtraction of one does not guarantee nonnegativity. Under a positive score multiplier a, severity is unchanged, SWS + 1 scales by a and PUS by a^2. These are descriptive inconsistency indices, not standardized Rasch MnSq, universally expectation-one statistics, or calibrated tests. No universal cutoffs or p-values are attached. A high PUS can reflect balanced reversals; low PUS may reflect sparse disagreement. Check overlap and score range before interpreting either.

Relation to MFRM and published network analyses

These are observed-score relationships, not fitted residuals or adjusted Rasch parameters. Lamprianou (2025) compares agreement, directional and rater-by-criterion networks with MFRM; a close empirical correspondence is not numerical equivalence. Signed alpha edges and weighted directional SWS/PUS indices are available here; `fit_rater_network()` supplies binary ERGM estimation. Halo-edge associations can reflect trait covariance and rubric overlap; `rater_halo_network_analysis()` offers person-cluster bootstrap contrasts instead of an independent-edge test. Use `mfrm_residual_network()` for residual associations and `subset_connectivity_report()` for prefit assignment links. See `vignette("mfrmr-networks-and-reporting")` for complementary views, paired overlap, direction and interpretation. `plot.mfrm_bundle()` documents custom layouts and display-only filters, which do not change the computed network.

References

Lamprianou, I. (2018). Investigation of rater effects using Social Network Analysis and Exponential Random Graph Models. *Educational and Psychological Measurement, 78*(3), 430-459.

Lamprianou, I. (2025). Network Analysis for the investigation of rater effects in language assessment: A comparison of ChatGPT vs human raters. *Research Methods in Applied Linguistics, 4*, 100205.

See Also

`interrater_agreement_table()`, `plot(x, type = "agreement")`, `mfrm_network_analysis()`, `plot.mfrm_bundle()`

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
if (requireNamespace("igraph", quietly = TRUE)) {
  rn <- rater_network_analysis(fit, diagnostics = example_fit$diagnostics,
    mode = "severity_direction")
  rn$summary
  head(rn$node_metrics)
  plot(rn, type = "severity", draw = FALSE)
}
```

---

rating_scale_table *Build a rating-scale diagnostics report*

Description

Build a rating-scale diagnostics report

Usage

```
rating_scale_table(
  fit,
  diagnostics = NULL,
  whexact = FALSE,
  drop_unused = FALSE
)
```

Arguments

fit	Output from `fit_mfrm()`.
diagnostics	Optional matching output from `diagnose_mfrm()`. Stored step uncertainty and marginal-fit results are reused when supplied. Without diagnostics, only observed/plug-in category statistics and step points are calculated; no covariance or full diagnostic calculation is started.

whexact	Retained for call compatibility. Category ZSTD is unavailable because grouping by the observed category changes its reference distribution.
drop_unused	If TRUE, remove categories with zero count from the displayed category table. Usage totals still cover the full declared scale; unavailable counts remain visible, and score-support caveats are retained.

Details

This helper provides category usage/fit statistics and threshold summaries for reviewing score-category functioning. category_table is a global observed-score screen. owner_category_table retains every declared category within each level of the fitted step_facet (or one Common family for RSM). Its percentages use that owner's total weight. These are observed ratings, not independent Persons or planned assignments. Missing categories alone cannot establish structural absence. drop_unused only filters the global display; owner counts stay complete.

Typical checks:

- sparse category usage (Count, ExpectedCount)
- category fit (Infit, Outfit, ZStd)
- threshold ordering within each StepFacet (threshold_table$Estimate, GapFromPrev)

Value

A named list with:

- category_table: counts, generating measures and descriptive residuals. AvgMeasure is the observed-category average oriented predictor; ExpectedMeasure weights all predictors by their category probabilities. AvgPersonMeasure and ExpectedAverage retain distinct Person-only and predicted-score summaries. Category ZSTD and fit flags are NA: selecting by observed outcome destroys the universal unit-residual baseline.
- owner_category_table: unfiltered counts within each step owner
- model_context: model, estimator, step/slope owners and step scale
- notes: interpretation outside the figure
- threshold_table: model step/threshold estimates and stored uncertainty
- summary: one-row summary with available/unavailable category counts and adjacent-threshold comparisons. Category-fit means identify their available denominators and describe displayed rows. Missing fit statistics/flags stay NA, including unused categories with no estimable category fit.
- category_usage: counts across the full scale, before drop_unused
- threshold_coverage: available, unavailable and decreasing adjacent-pair counts; NotApplicable distinguishes binary scales from missing estimates
- caveats: structured score-support warning/review rows
- diagnostic_mode: character scalar carried from diagnostics$diagnostic_mode ("legacy", "both", or "marginal_fit"); used by downstream reporting helpers to pick the correct expected-count basis
- marginal_fit: list bundle from diagnostics$marginal_fit when strict marginal fit was computed, otherwise NULL. Carries the raw OverallRMSD / OverallMaxAbsStdResidual / per-cell tables that feed the MarginalOverallRMSD columns in summary.

Interpreting output

Start with summary:

- UsedCategories close to total Categories suggests that most score categories are represented in the observed data.
- very small MinCategoryCount indicates potential instability.
- ThresholdMonotonic = FALSE indicates disordered thresholds within at least one threshold set. In PCM fits, inspect threshold_table by StepFacet before drawing scale-wide conclusions.

Then inspect:

- category_table for global category-level misfit/sparsity.
- threshold_table for adjacent-step gaps and ordering within each StepFacet.

Stored diagnostics$parameter_uncertainty$steps are matched by threshold family and step, with fitted estimates and source identity checked. Inspect SE_Status, CIEligible, CIUse and UncertaintySourceStatus. Missing rows remain unavailable; duplicate or mismatched keys are rejected. Older diagnostics without a verifiable fitted-calibration identity must be regenerated with diagnose_mfrm(fit, ...); model refitting is not required. Reordering stored step rows does not change their keyed correspondence. A bare fit supplies step points with SE_Status = "not_supplied". A binary centered step has no free within-owner contrast (fixed_by_step_centering).

Native thresholds here are centered step offsets. They are not full ability-axis crossings, cumulative-50% boundaries or decision cuts. An SE for a step does not include the covariance required for a crossing or an adjacent-step difference. Threshold disorder is descriptive, not an automatic category-merging rule. Two-family GPCM diagnostics remain unsupported by this helper; use its dedicated response/curve routes.

Typical workflow

1. Fit model: fit_mfrm().
2. Build diagnostics: diagnose_mfrm().
3. Run rating_scale_table() and review summary().
4. Use plot() to visualize category profile quickly.

Save the complete result to retain owner counts, uncertainty status and notes. plot(out, type = "counts") shows global counts; use plot(out, type = "thresholds") for the stored step points. Both support as_ggplot() without recalculating values. Step figures do not draw stored SEs as crossing, gap or curve intervals; inspect threshold_table separately.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and vignette("mfrmr-visual-diagnos
package = "mfrmr").

Output columns

The `category_table` data.frame contains:

Category Score category value.

Count, Percent Observed count and percentage of total. With observation weights, counts are sums of weights, not independent sample sizes. Missing or invalid scores/weights make the usage counts unavailable.

AvgMeasure, ExpectedMeasure Observed-category mean oriented predictor and its model expectation, weighted by category probabilities across all retained observations.

AvgPersonMeasure Mean person measure for respondents in this category; distinct from the oriented predictor.

ExpectedAverage Mean predicted score among observations in this observed category; distinct from the expected generating measure.

Infit, Outfit Descriptive category residual mean squares.

InfitZSTD, OutfitZSTD Unavailable (NA); conditioning on the observed category does not preserve a universal unit-residual reference.

ExpectedCount, DiffCount Expected count and observed-expected difference.

LowCount Logical; `TRUE` if count is below minimum threshold.

InfitFlag, OutfitFlag, ZSTDFlag Unavailable (NA). Outcome-selected residual summaries do not have a universal unit baseline or calibrated ZSTD reference. Low-count and generating-measure warnings remain descriptive.

ZeroCount, UnusedCategoryType, WeaklyIdentified, CategoryCaveat Structured score-support caveats for retained zero-count categories.

The `owner_category_table` contains `StepOwner` (facet name), `StepFacet` (level), `Category` (model score), `CategoryLabel` (original score), raw and weighted counts (`RawCount`, `Count`), denominators (`OwnerRatings`, `OwnerWeight`), `Percent`, `ZeroCount` and `UsageStatus`. Status distinguishes observed, not observed, zero weight, no exposure and unavailable counts. No-exposure counts are zero but percentages are unavailable.

The `threshold_table` data.frame contains:

Step Step label (e.g., `Step_1`, `Step_2`). Use `LowerCategory` and `UpperCategory` to identify the corresponding score transition.

Estimate Estimated threshold/step difficulty (logits).

StepOwner Name of the facet owning the steps, or `Common` for RSM.

StepFacet Threshold family identifier when the fit uses facet-specific threshold sets.

SE, CI_Lower, CI_Upper, SE_Status, CIEligible, CIUse Stored step uncertainty and reporting status, without new inference.

LowerCategoryLabel, UpperCategoryLabel Original scores at the transition.

LowerOwnerCount, UpperOwnerCount, OwnerSupportStatus Weighted counts and support within this threshold owner, not pooled frequencies.

GapFromPrev Difference between adjacent numbered thresholds within the same `StepFacet`. Missing thresholds are not skipped to form a gap. No automatic category-merging rule is applied.

ThresholdMonotonic Logical flag repeated within each threshold set. `FALSE` records at least one decreasing adjacent pair; `TRUE` requires every expected pair to be available and nondecreasing, allowing numerical differences up to `sqrt(.Machine$double.eps)`. Equal thresholds meet this descriptive condition. Otherwise the flag is `NA`. A binary scale has only one threshold and no applicable ordering comparison. This is a statement about point estimates, not a test of category adequacy.

LowerCategory, UpperCategory, WeaklyIdentified, ThresholdCaveat Adjacent score-category support metadata. Thresholds adjacent to retained zero-count categories are flagged for cautious interpretation.

References

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573. doi:10.1007/BF02293814
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174. doi:10.1007/BF02296272
- Linacre, J. M. (2002). What do Infit and Outfit, mean-square and standardized mean? *Rasch Measurement Transactions, 16*(2), 878. (Source for the 0.5-1.5 mean-square heuristic review interval; this is not a source for threshold-gap rules.)
- Wind, S. A. (2023). *Detecting rating scale malfunctioning with the partial credit model and generalized partial credit model*. Educational and Psychological Measurement, 83(5), 953-983. doi:10.1177/00131644221116292 (Recent simulation evidence on PCM- and GPCM-based rating-scale diagnostics; useful for interpreting the `summary(t8)$summary` flags in the `GPCM` route.)

See Also

`diagnose_mfrm()`, `measurable_summary_table()`, `plot.mfrm_fit()`, mfrmr_visual_diagnostics

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Review category use and the fitted transitions between scores
categories <- rating_scale_table(fit)
review <- summary(categories)
review$summary
categories$model_context
head(categories$owner_category_table)

# Bars show observed counts; the line shows model-expected counts
# Recreate saved tables with the original fit and diagnostics after updating.
plot(categories)
```

`read_facets_fit_table` *Read a FACETS fit table for fit review*

Description

For standalone plots with source-state checks, use `read_facets_results()`.

Usage

```
read_facets_fit_table(
  file,
  facet = NULL,
  facet_map = NULL,
  format = c("auto", "delimited", "scorefile"),
  facet_col = NULL,
  level_col = NULL,
  delimiter = NULL,
  encoding = "UTF-8"
)

import_facets_fit_table(
  file,
  facet = NULL,
  facet_map = NULL,
  format = c("auto", "delimited", "scorefile"),
  facet_col = NULL,
  level_col = NULL,
  delimiter = NULL,
  encoding = "UTF-8"
)
```

Arguments

`file`	Path to a FACETS-derived fit table. A character vector of files is accepted. A directory containing `score.N.txt` files is also accepted.
`facet`	Optional facet name to assign when the file does not contain a facet column. Use this for one-facet CSV exports.
`facet_map`	Optional character vector mapping FACETS score-file numbers to facet names, for example `c("1" = "Person", "2" = "Rater")`. If unnamed, positions are used as score-file numbers.
`format`	File format. `"auto"` detects FACETS `score.N.txt` files from their name/header; `"delimited"` reads a CSV/TSV/semicolon table; and `"scorefile"` reads a FACETS score-file table, including comma, tab or semicolon exports with facet heading lines. Fixed-field input requires validated all-field headings with element number after the statistics; appended facet fields are separated from the

element label. Headerless, element-first and selected-field fixed layouts are rejected. Use named delimited columns for those layouts and for non-ASCII labels.

`facet_col`, `level_col`
: Optional explicit column names for delimited tables when automatic detection is not sufficient. For score files, `level_col` can override the column immediately before `F-Number`.

`delimiter`
: Optional delimiter for delimited tables. If omitted, comma, tab, and semicolon are detected from the header line.

`encoding`
: Source file encoding, explicitly converted to UTF-8.

Details

This helper does not run FACETS. It reads FACETS output that already exists on disk and normalizes it to columns that `facets_fit_review()` can consume: `Facet`, `Level`, `Estimate`, `SE`, `N`, `Infit`, `Outfit`, `InfitZSTD`, `OutfitZSTD`, `DF_Infit`, and `DF_Outfit`. File imports also retain the exact reported numeric tokens (for example, `InfitZSTDRaw = "2.0"`) and their displayed decimal counts. FACETS comparison runs should first request the highest practical output precision (for example, `Udecim=8` for Measure and SE). If the resulting displayed absolute ZSTD is still exactly equal to 2, it is classified as `display_boundary_indeterminate`: the text file does not expose whether the unrounded value lies just below, at, or just above the conventional threshold.

Two common workflows are supported:

- a FACETS score file such as `score.2.txt`, where the facet name is supplied by `facet_map` or inferred as `Facet2`. Comma, tab and semicolon score files with field names and fixed-field score files using the FACETS manual column positions are supported;
- a CSV/TSV table already exported from FACETS or a harmonization script, with FACETS-style column names such as `Infit MnSq`, `Outfit ZStd`, and `T.Count`.

After import, pass the table to `facets_fit_review()`. Inspect `review$external_table_quality` first when the FACETS export is partial, duplicated, or missing MnSq/df columns. Then inspect `review$external_comparison` for supplied FACETS-vs-mfrmr differences and `review$df_sensitivity` / `review$df_sensitive` for engine-vs-FACETS-style df/ZSTD convention sensitivity. Use `plot(review, type = "df_sensitivity")` for a quick visual check of the largest ZSTD shifts caused by df convention.

Value

A tibble with standardized fit-table columns suitable for `facets_fit_review(fit, facets_fit = read_facets_fit_table(...))`.

See Also

`facets_fit_review()`, `diagnose_mfrm()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("read_facets_fit_table", run.dontrun = TRUE)
## Not run:
path <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    Facet = "Rater", Level = "R1", Infit = 1.02, Outfit = 0.98,
    InfitZSTD = 0.3, OutfitZSTD = -0.2, DF_Infit = 12, DF_Outfit = 13
  ),
  path,
  row.names = FALSE
)
read_facets_fit_table(path)

## End(Not run)
```

`read_facets_results` *Read FACETS results for standalone visualization*

Description

Import reported element measures, fit statistics and optional Graphfile curves without refitting a model. Use `plot.mfrm_facets_results()` to inspect the saved results.

Usage

```
read_facets_results(
  file = NULL,
  facet = NULL,
  facet_map = NULL,
  format = c("auto", "delimited", "scorefile"),
  facet_col = NULL,
  level_col = NULL,
  delimiter = NULL,
  encoding = "UTF-8",
  source_version = NULL,
  measure_units = NULL,
  common_scale = FALSE,
  graph_file = NULL
)

## S3 method for class 'mfrm_facets_results'
summary(object, ..., component = c("elements", "curves"))

## S3 method for class 'mfrm_facets_results'
print(x, ...)
```

```
## S3 method for class 'mfrm_facets_results'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  component = c("elements", "curves")
)

## S3 method for class 'mfrm_facets_results'
plot(
  x,
  type = c("measure", "wright", "fit", "infit_outfit", "ccc", "expected"),
  facets = NULL,
  statistic = c("infit", "outfit"),
  draw = TRUE,
  show_title = FALSE,
  show_notes = FALSE,
  show_legend = TRUE,
  scales = NULL,
  label_map = NULL,
  ...
)
```

Arguments

`file`	Scorefile input as described below, or `NULL` when only `graph_file` is supplied.
`facet`	Optional facet name to assign when the file does not contain a facet column. Use this for one-facet CSV exports.
`facet_map`	Optional character vector mapping FACETS score-file numbers to facet names, for example `c("1" = "Person", "2" = "Rater")`. If unnamed, positions are used as score-file numbers.
`format`	File format. `"auto"` detects FACETS `score.N.txt` files from their name/header; `"delimited"` reads a CSV/TSV/semicolon table; and `"scorefile"` reads a FACETS score-file table, including comma, tab or semicolon exports with facet heading lines. Fixed-field input requires validated all-field headings with element number after the statistics; appended facet fields are separated from the element label. Headerless, element-first and selected-field fixed layouts are rejected. Use named delimited columns for those layouts and for non-ASCII labels.
`facet_col, level_col`	Optional explicit column names for delimited tables when automatic detection is not sufficient. For score files, `level_col` can override the column immediately before `F-Number`.
`delimiter`	Optional delimiter for delimited tables. If omitted, comma, tab, and semicolon are detected from the header line.
`encoding`	Source file encoding, explicitly converted to UTF-8.

source_version — Optional FACETS version recorded with these exports. Unknown versions remain NA; the reader does not infer a version.

measure_units — Optional description of the reported measure units, after checking FACETS reporting transformations. Unknown units remain NA.

common_scale — Logical; explicitly declare that the imported facets have compatible units and origins for a shared-axis Wright map. Defaults to false. This declaration does not transform, center or reverse measures.

graph_file — Optional path to one FACETS Graphfile with headings, in CSV, tab-separated or validated ASCII fixed-width format. Supply an export from the same analysis when also importing element results. File identity is recorded, but the reader cannot certify that the exports belong together.

object — A saved mfrm_facets_results object.

... — Must be empty.

component — "elements" (default) or "curves" for Graphfile results.

x — Saved output of read_facets_results().

Also for summary(): A saved mfrm_facets_results object.

row.names — Optional row names for the exported data frame.

optional — Passed to as.data.frame().

type — "measure" for labelled element locations by facet; "wright" for points by facet on one measure axis; "fit" for measure versus mean square; or "infit_outfit" for Infit versus Outfit. Wright maps require the explicit common_scale = TRUE declaration at import. "ccc" plots Graphfile category probabilities and "expected" plots Graphfile expected scores, each in separate scale panels.

facets — Optional character vector of facet names to display.

statistic — "infit" or "outfit", used for type = "fit".

draw — Logical; draw with base graphics when true.

show_title, show_notes — Logical; show title/interpretation notes in the plot, off by default. Notes always remain in the returned data.

show_legend — Logical; display the source-status symbol key or category curve key below the panels. Defaults to true.

scales — Optional vector of Graphfile scale numbers for curve plots. Facet selectors do not apply to curves, and scale selectors do not apply to element plots.

label_map — Optional data frame with character columns Facet, ElementID and DisplayLabel for element plots. Keys must match exported IDs in x, including any leading zeros, and must be unique within a facet. A partial map is allowed; unmapped rows retain their source labels. Mapping is checked against the full result before facets selection. This changes display labels only, not source labels or element identity. Curve plots do not accept this map. Only type = "measure" draws element labels; the other element plots retain them in returned plot data.

Details

Each file must describe one analysis. Supply a vector of facet files or a directory containing `score.N.txt` files, or a named delimited table. Comma, tab and semicolon Scorefiles retain their native facet heading lines. Numeric facet numbers default to `Facet1`, `Facet2`, etc.; supply `facet_map` to name them. A data frame is also accepted. The supported layouts follow `read_facets_fit_table()`. Use named delimited files for non-ASCII labels; fixed-width imports require ASCII because byte and character widths can differ. For text output from FACETS, request `Scorefile=score.txt`, `Heading lines=Yes`, `QM=Double` and `CSV=Commas`, `Tabs` or `Semicolons`. Check that the files were actually generated; a zero process exit code alone does not establish a successful export. Specify the encoding of the existing file; the reader does not configure FACETS input encoding or repair labels already lost during export. Element IDs are preserved separately from display labels. Where no ID was exported, the label is the key; ambiguous duplicate keys are rejected, not averaged. Unknown source status remains unknown.

FACETS status codes distinguish anchor-flagged, estimated, extreme, no-data, unmeasurable and single-observation elements. `StatusRaw = "1"` and the retained `SourceStatus = "anchored"` mean an anchor flag, not proof that the individual measure was fixed. FACETS 4.5.0 uses this flag for estimated group-anchor members as well as direct anchors and singleton groups. `GroupRaw` can also describe a reporting group; it does not establish the anchor subtype. Neither a group's imported size nor a reported SE resolves that ambiguity. Check the original `Labels=` specification for the anchor mode and Table 7 for row annotations. A Table 7 G annotation alone need not distinguish the G and X input modes. This reader does not parse those sources or infer a subtype. Extreme group members can instead carry Status -2/-3, so `AnchorFlagged` does not count every member of an anchored group. Finite extreme measures are retained as reported locations, not ordinary finite maximum-likelihood estimates. The status alone does not identify their adjustment rule: native X group anchoring can report a different location from G anchoring. Reported fit values for extremes can be placeholders, so their fit plots are unavailable. No-data and unmeasurable elements remain in the returned table. Missing or censored values are never replaced with zero. Marginal source SEs are retained for inspection, but do not supply joint covariance or validated confidence intervals. The object contains no likelihood, response data, inferred thresholds, new-person scoring or calibration-inclusive inference.

Source measures, signs and rounding are retained. A positive or negative sign describes the source model contribution; the plot does not multiply measures by that sign. Check the original specification before interpreting severity, ability or comparing facet origins. Reading results does not validate their convergence, model fit or substantive interpretation.

Graphfiles preserve the reported `Scale`, `Measure`, `Expected`, `ExpCat` and `Prob:<category>` fields. Scale numbers identify Table 8 subtables, not automatically items or facets. Category labels can differ from response codes (including binary exports labelled 1/2); no recoding is performed. `Expected` is used directly, not recomputed from probabilities or `ExpCat`. The Graphfile Measure axis is not automatically a Person ability axis or the coordinate system of Scorefile measures; `measure_units` and `common_scale` apply only to element plots.

Graph imports require unambiguous headings and strictly monotone distinct finite measures within each scale. Missing coordinates/values are retained with reasons and break curve lines. Probability sums are checked against the sum of half the printed units of precision, plus floating-point tolerance; sums outside that rounding allowance are withheld from probability plots. Incomplete probability vectors remain flagged, with available individual probabilities retained. Nothing is renormalized, smoothed or extrapolated. These are source-consistency checks, not validation of the fitted model.

`plot()` method: Plot imported FACETS measures, fit statistics and curves

For `plot()`: Each measure panel has its own axis; the Wright display shares one measure axis without centering, reversing signs or inventing category thresholds. Filled circles denote estimated elements, squares direct/group anchor flags, downward/upward triangles extreme low/high source status, diamonds single observations and open circles unknown source status. The symbol key is in `data$legend`. A square marks source Status 1; it does not identify an individually fixed measure. Group-anchor members can vary around a fixed group mean, and `GroupRaw` may instead be a reporting group. Check the original specification or Table 7 to distinguish direct and group anchors. Returned notes retain this limitation, also when plotting imports saved by an earlier version; the source object is not changed. Recreate saved plot-data objects with `plot()` to obtain the revised legend and notes. Duplicate display labels are disambiguated by element key in measure plots. Use `label_map` to display local names while retaining the original FACETS labels and IDs. It cannot recover names lost in an export; supply the known correspondence yourself. Character preservation and font rendering are separate: choose an installed font with the required glyphs on the graphics device. With Cairo support, `grDevices::png()` with `type = "cairo"` and `grDevices::cairo_pdf()` accept a `family` such as `"Meiryo"` on Windows when installed. Font availability differs across systems. The plot does not install fonts or change global font settings. Infit/Outfit reference lines at one describe the model reference, not a fairness or acceptability cutoff. No confidence bars are calculated. Extreme/single-observation fit placeholders and unmeasurable/no-data elements are retained as unavailable. A plot can have no usable points; inspect its row-level reasons. Unknown status is displayed explicitly, not certified. Graphfile curves join consecutive exported coordinates with straight line segments; this does not calculate new model predictions. Missing or invalid rows break the lines. The legend identifies source category labels. Expected scores and category labels retain their source coding. Each panel has its own reported Measure axis, independent of element-plot scale declarations. For curves, `data$data` holds the raw Graphfile table, `data$plot_data` the selected long rows, and `data$diagnostics` the probability-sum checks.

Value

A `mfrm_facets_results` list with `table` (all imported rows, source tokens, IDs, status and per-quantity availability), `sources` (file paths, MD5 checksums and encoding; unavailable for data frames), `metadata` (source declarations) and `notes`. When supplied, `graph` contains the original character `table`, long `curves` with raw tokens, precision and availability, row-level `diagnostics`, `categories` and detected `layout`. Graph `metadata` identifies its separate coordinate system; its units remain unknown rather than inheriting element `measure_units`. `summary()` counts available rows by facet. `AnchorFlagged` counts `SourceStatus = "anchored"` rows, not individually fixed parameters; it replaces the earlier candidate's ambiguous `Anchored` summary column. `as.data.frame()` exports the element table. Use `component = "curves"` with either method to inspect Graphfile curves. Use `saveRDS()` to retain the full object.

For `plot()`: Invisibly an `mfrm_plot_data` object. `data$plot_data` retains every selected row with `X`, `Y`, `Available` and `Reason`; element plots also have `Symbol`, and curves have `Series`, `Color`, `LineType` and `Segment`. Unavailable numerical coordinates are `NA` (measure plots retain label positions). Original reported values/status remain in the same rows. `data$notes`, `data$sources` and `data$metadata` describe interpretation and origin. `data$label_map` retains a supplied display map. Filtering and display labels do not change the saved results.

References

FACETS Scorefile specification: https://www.winsteps.com/facetman64/scoreoutput.htm. FACETS Graphfile specification: https://www.winsteps.com/facetman64/graphfile.htm. FACETS

element and Table 7 anchor annotations: https://www.winsteps.com/facetman64/elements.htm and https://www.winsteps.com/facetman64/table7.htm.

Examples

```
# Small synthetic binary-rating analysis exported by FACETS.
# Each row in result$table is one Person, Rater or Task element.
path <- system.file("extdata", "facets-example", package = "mfrmr")
result <- read_facets_results(path,
  facet_map = c("Person", "Rater", "Task"), source_version = "4.5.0")
summary(result) # Availability, not a model-adequacy assessment.
result$table[, c("Facet", "ElementID", "Label", "Estimate", "SourceStatus")]
plot(result, type = "measure", facets = "Rater", draw = FALSE)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_facets_results", run.dontrun = TRUE)
## Not run:
path <- system.file("extdata", "facets-example", package = "mfrmr")
result <- read_facets_results(path,
  facet_map = c("Person", "Rater", "Task"), source_version = "4.5.0")
plot(result, type = "measure", facets = "Person")
p <- plot(result, type = "infit_outfit", facets = "Rater", draw = FALSE)
p$data$plot_data[, c("Label", "X", "Y", "Available", "Reason")]
p$data$notes
# Display names are joined by facet and exported ID, never by row order.
labels <- data.frame(Facet = "Person", ElementID = c("1", "2"),
  DisplayLabel = c("Participant A", "Participant B"))
labelled <- plot(result, facets = "Person", label_map = labels, draw = FALSE)
labelled$data$plot_data[, c("ElementID", "Label", "DisplayLabel")]
# A separate synthetic PCM analysis, with four exported scale curves.
graph_path <- system.file("extdata", "facets-graph-example", "graph.txt",
  package = "mfrmr")
curves <- read_facets_results(graph_file = graph_path, source_version = "4.5.0")
summary(curves, component = "curves")
plot(curves, type = "ccc", scales = 1)
expected <- plot_data(curves, "plot_data", type = "expected", scales = 1)
head(expected[, c("Scale", "Measure", "Value", "Available")])

## End(Not run)
```

---

read_mfrm_posterior *Read an exported Bayesian measurement summary*

Description

Bring a saved Stan/brms summary back into mfrmr for figures and reports, without loading either backend or running an estimator.

Usage

```
read_mfrm_posterior(path)

## S3 method for class 'mfrm_posterior_summary'
plot(
  x,
  quantity = c("location", "ability", "step_offset", "slope", "log_slope",
    "population_mean", "population_sd"),
  facet = NULL,
  draw = TRUE,
  reference = NULL,
  preset = "standard",
  show_legend = TRUE,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ...
)

## S3 method for class 'mfrm_posterior_summary'
print(x, ...)

## S3 method for class 'mfrm_posterior_summary'
summary(object, ...)

## S3 method for class 'mfrm_posterior_summary'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

path	An export directory containing `posterior-summary.rds` and its matching `model-data.rds`, or the path to that summary file.
x	Output from `read_mfrm_posterior()`. Also for `print()`: A saved `mfrm_posterior_summary`.
quantity	One of `"location"` (default), `"ability"`, `"step_offset"`, `"slope"`, `"log_slope"`, `"population_mean"` or `"population_sd"`. Population quantities require an estimated-population export. Quantities with different units are not mixed.
facet	Optional facet/owner names to retain; `NULL` retains all for the selected quantity. Step rows use `facet = "Step owner"` and retain their owner labels and category transitions in the plotted table.
draw	Logical; `FALSE` returns plot data without opening a device.
reference	Optional finite vertical reference; `NULL` omits it. Zero for locations and one for relative slopes are references, not standards of performance or rater quality.
preset	A package plot preset, such as `"standard"` or `"monochrome"`.
show_legend	Logical; show interval/fixed-value symbols.

`title`, `subtitle`, `caption`
: Optional returned text. Defaults omit it. Text is not drawn by default; retrieve it with `plot_data()`.

`...`
: Unused.

`object`
: A saved `mfrm_posterior_summary`.

`parm`
: Optional saved parameter names (for example `"locations[1]"`) or row indices in `$parameters`. `NULL` returns mapped model parameters.

`level`
: The stored level, currently `0.95`. Other levels require posterior draws and are rejected rather than approximated from an SD.

Details

Run the exported bundle's `Rscript run.R --summarize` first. The reader checks the recorded model-data checksum, parameter mappings, fixed-value semantics and consistency of the saved sampling checks. Older summaries without a checksum must be regenerated with the current bundled `run.R` and `summarize.R`; no new sampling is needed. Keep the original `model-data.rds` bytes with the fit when moving an export directory.

These checks detect mismatched files and inconsistent summaries; they do not independently verify draws, convergence, prior adequacy or model fit. The object contains marginal summaries, not joint posterior draws or a native fitted model. It supports `mfrm_report()`, `apa_table()`, `build_summary_table_bundle()` and `export_summary_appendix()`. Use the backend's saved fit for new credible levels, contrasts, predictions, trace plots or posterior predictive checks. Native scoring and MML/JML diagnostics are not available from this object. Estimated-population one-family Stan summaries retain the population mean and SD as reported parameters. Their schema-2 files require this newer reader; existing fixed-population schema-1 summaries remain supported. Use `plot(x, quantity = "population_sd")` to inspect between-Person variation on the native metric. This SD is not an individual Person standard error.

`confint()` retrieves the stored equal-tailed 95% *posterior credible* intervals, retaining fixed/unavailable rows and review flags. It does not calculate frequentist confidence intervals or change the saved level. Fixed anchors and constraint-determined values have no estimated SD or interval. External calibration uncertainty has not been propagated. `summary()` returns the complete validated summary; printing is compact. The bundled synthetic example uses short chains and requires sampling review. Its numbers are workflow illustrations, not assessment findings.

`plot()` method: Plot saved posterior estimates and credible intervals

For `plot()`: These are saved equal-tailed 95% posterior credible intervals, not frequentist confidence intervals. Dashed intervals and open circles mark estimates whose sampling screen requires review; solid intervals mean that no sampling flag was detected, not that inference is validated. Open diamonds mark fixed anchors/constraint-determined values without an interval. Crosses mark estimated points without an available interval. Missing points retain their labels and are never replaced by zero. Original row order, labels and endpoints are preserved. The plot never refits, changes the level, selects significant effects or validates priors.

Value

An `mfrm_posterior_summary` containing unrounded parameter and diagnostic tables, source settings, priors, mappings and constraints.

For `plot()`: Invisibly, an `mfrm_plot_data` with the exact long `table`, source settings, parameter identities, interval statuses and interpretation notes. `as_ggplot()` returns an editable ggplot; `plot_data()` supplies the table for custom graphics. No posterior draws, densities or raincloud samples are reconstructed from marginal summaries.

Session plot defaults

Set `options(mfrmr.plot_preset = "publication")` to choose a session default for plotting functions that expose the common `preset` argument. The supported values are `"standard"`, `"publication"`, `"compact"` and `"monochrome"`. Precedence is an explicit call argument, then the session option, then `"standard"`. For example, `preset = "standard"` overrides a session set to `"monochrome"`. Explicit `preset = NULL` retains the earlier package-default behavior; it does not read the session option. Invalid session values cause an error only when that option is needed.

The category-curve, data-quality, fit-review, connectivity and network routes of `plot()` for report bundles use the same option through `....` Plots without a common `preset` argument, including extended-model plots with their own `palette` controls, keep their own settings. This option selects a preset, not a universal theme or a guarantee that all renderers implement every appearance control identically.

New plot payloads retain the resolved preset for supported saved-data rendering. Converting an existing payload with `as_ggplot()` uses its saved appearance, even after the session option changes. A call that creates a new plot from a fit or statistical result uses the current default. For a reproducible script, supply `preset` explicitly or set the option in that script. Saving only the fitted model does not save a session option. No global ggplot theme is changed.

Restore previous settings with `old <- options(mfrmr.plot_preset = "monochrome")` followed by `options(old)`. Use `options(mfrmr.plot_preset = NULL)` to remove the option. The preset changes appearance, not estimates, confidence levels or diagnostic thresholds.

See Also

`export_mfrm_model()`, `plot.mfrm_posterior_summary()`

`read_mfrm_posterior()`, `mfrm_report()`

Examples

```
directory <- system.file("extdata", "posterior-example", package = "mfrmr")
posterior <- read_mfrm_posterior(directory)
summary(posterior) # Read the decision and next action before the estimates.
intervals <- confint(posterior)
head(intervals[, c("facet", "level", "value_role", "lower", "upper", "interval_status")])
report <- mfrm_report(posterior, style = "apa", study_context = c(
  score_direction = "Higher coded scores indicate more favorable performance"))
summary(report, view = "reader")
posterior <- read_mfrm_posterior(system.file(
  "extdata", "posterior-example", package = "mfrmr"))
p <- plot(posterior, quantity = "location", draw = FALSE)
plot_data(p, "table")
if (requireNamespace("ggplot2", quietly = TRUE)) as_ggplot(p)
```

recode_missing_codes *Recode common missing-value sentinels to* NA

Description

Convenience helper that replaces the standard non-NA missing-code sentinels used in SPSS / SAS / FACETS exports (99, 999, -1, "N", "NA", "n/a", ".", "") with NA across the columns you select. It is useful before calling fit_mfrm() on data exported with those conventions. A sentinel is a value used to mean "missing" instead of an actual score. Only recode values that your data documentation defines as missing. For custom score markers, specify both columns = "Score" and codes; this preserves a person or rater identifier such as 99.

Usage

```
recode_missing_codes(
  data,
  columns = NULL,
  codes = c("99", "999", "-1", "N", "NA", "n/a", ".", ""),
  numeric_codes = TRUE,
  verbose = FALSE
)
```

Arguments

data	A data frame.
columns	Character vector of column names to recode. Defaults to NULL, in which case all columns are scanned.
codes	Character vector of code values to convert to NA. Defaults to the FACETS / SPSS / SAS conventions; override when your instrument uses different sentinels.
numeric_codes	Logical; if TRUE (default), numeric columns are also compared against the numeric conversion of codes.
verbose	Logical; if TRUE, emits a message() summary of per-column replacement counts.

Details

Save the returned data, for example cleaned <- recode_missing_codes(...). The original object is unchanged. This helper replaces cells and retains every row; later, describe_mfrm_data() and fit_mfrm() exclude rating rows with missing scores or required identifiers. Inspect the replacement counts with attr(cleaned, "mfrm_missing_recoding"), then review and fit cleaned.

The defaults differ across entry points: this helper scans all columns when columns is omitted. In describe_mfrm_data() and fit_mfrm(), missing_codes = TRUE uses the conventional code set on the score column only; an explicit missing_codes vector applies to the person, facet, and score columns. Use this helper with an explicit score column when your custom code could also be a legitimate identifier.

Value

The input `data` with the specified missing sentinels replaced by `NA`. A `mfrm_missing_recoding` attribute records the per-column replacement counts for traceability logs.

See Also

`describe_mfrm_data()`, `fit_mfrm()`.

Examples

```
library(mfrmr)

# A small input example: 99 and . mean missing only in the Score column
ratings <- data.frame(
  Person = c("001", "001", "002", "002"),
  Rater = c("R1", "99", "R1", "99"),
  Score = c("3", "99", ".", "2")
)
cleaned <- recode_missing_codes(
  ratings,
  columns = "Score",
  codes = c("99", ".")
)
cleaned # Two scores become NA; rater ID 99 and all four rows remain
attr(cleaned, "mfrm_missing_recoding") # Score: Replaced = 2
# Use the returned cleaned data for subsequent data review and fitting
```

---

recommend_mfrm_design *Recommend a design condition from simulation results*

Description

Recommend a design condition from simulation results

Usage

```
recommend_mfrm_design(
  x,
  facets = c("Rater", "Criterion"),
  min_separation = 2,
  min_reliability = 0.8,
  max_severity_rmse = 0.5,
  max_misfit_rate = 0.1,
  min_convergence_rate = 1,
  prefer = c("n_person", "raters_per_person", "n_rater", "n_criterion"),
  max_ratings = NULL,
  max_ratings_per_rater = NULL,
  require_connected = TRUE
```

```
)

## S3 method for class 'mfrm_design_recommendation'
summary(object, ...)

## S3 method for class 'summary.mfrm_design_recommendation'
print(x, ...)

## S3 method for class 'mfrm_design_recommendation'
print(x, ...)

## S3 method for class 'mfrm_design_recommendation'
plot(x, draw = TRUE, ...)
```

Arguments

x	For recommend_mfrm_design(): Output from evaluate_mfrm_design() or summary.mfrm_design_evaluation(). For plot(): Output from recommend_mfrm_design().
facets	Non-empty vector of facets that must satisfy the planning thresholds. By default, use the stored non-person facet names; for older objects without these names, use the non-person facets in the summary.
min_separation	Minimum acceptable mean separation.
min_reliability	Minimum acceptable mean reliability.
max_severity_rmse	Maximum acceptable severity recovery RMSE.
max_misfit_rate	Maximum acceptable mean misfit rate.
min_convergence_rate	Minimum acceptable convergence rate.
prefer	Ranking priority among design variables. Earlier entries are optimized first when multiple designs pass. Custom public aliases from sim_spec are also accepted, as are the role keywords person, rater, criterion, and assignment.
max_ratings	Optional upper bound on total rating rows in each evaluated replication. Supply a non-negative whole number, or NULL (default) to impose no limit.
max_ratings_per_rater	Optional upper bound on rating rows assigned to any one rater-like facet level in each evaluated replication. Supply a non-negative whole number, or NULL (default) to impose no limit.
require_connected	Logical; require both generated Person-facet assignment graphs to be connected in every recorded replication (default TRUE). This covers both non-person facets regardless of which performance metrics are selected through facets. Set FALSE only to explicitly omit this structural screen; the recorded status is retained.
object	Output from recommend_mfrm_design().

...	Unused; extra arguments are rejected.
draw	Logical; draw the criterion-status matrix. Use FALSE to retrieve every criterion and source table without drawing.

Details

This helper screens simulated designs under user-chosen planning criteria. The default targets non-Person facet calibration: high separation reliability means that facet-level differences can be distinguished, not that raters agree. Use facets = "Person" to target Person differentiation instead. Neither use establishes classification accuracy or a universal sample size. Inspect summary(result)$target and $thresholds before interpreting the recommendation. Defaults are planning choices, not validated standards. summary.mfrm_design_recommendation() explains the criterion outcomes; plot.mfrm_design_recommendation() and mfrm_report() preserve them without repeating the simulation. No plan-selection confidence interval is computed.

A design is marked as recommended when all requested facets satisfy all selected thresholds simultaneously. Each design must have results for every requested facet. Missing facets are reported in FacetsMissing and prevent that design from passing; FacetsRequired always counts the requested facets, not the available rows. Designs with none of the requested facets have no facet-check rows and cannot be recommended. A requested facet absent from the entire summary produces an error. If multiple designs pass, the helper returns the smallest one according to prefer (by default: fewer persons first, then fewer ratings per person, then fewer raters, then fewer criteria). The convergence threshold uses all recorded replications, including failures that returned no facet metrics, as summarized by summary.mfrm_design_evaluation(). Threshold checks use unrounded metrics. Summaries saved by earlier versions may contain only rounded values; rebuild them with summary(original_evaluation) before requesting a recommendation. The original evaluation can be reused without generating or fitting new data. Summaries also need the current recovery-coordinate review. Rebuilding an old PCM/GPCM summary can leave its recovery fields unavailable, preventing a recommendation; a new evaluation is needed for corrected severity recovery. An unavailable group minimum/maximum is NA, not infinite apparent success.

The connectivity screen uses generated assignments, including failed fits and diagnostic runs. ConnectivityStatus is "disconnected" if any recorded graph is disconnected, "connected" if both graphs are connected in every replication, and "not_assessed" otherwise. With the default require_connected = TRUE, only "connected" passes. Older saved objects without component counts cannot pass this screen; re-summarizing alone cannot recover the missing assignment evidence.

This is a conservative screen for comparisons supported by shared-person assignments, not a test of full model identification or adequate precision. MML population assumptions can supply information beyond these connections; disabling the screen does not validate that assumption-based comparison. Connections may be indirect, so a rater pair without common persons is not by itself evidence of a disconnected design. LinkReviewStatus and LinkReviewReason carry the separate sparse-design overlap review: "review" flags a missing overlap count, a pair without common persons, or a pair below its recorded target; "ok" means those recorded checks pass, and "not_assessed" means no sparse overlap target was assessed. They do not change Pass; overlap counts alone do not establish precision or a universal minimum.

Workload limits apply to the maxima across all recorded replications, including failed fits and diagnostic runs. One rating is one generated person-rater-criterion row, not a bundle of criteria or a weighted count. For example, two raters each scoring three criteria for ten persons use 60 ratings,

with 30 per rater. Counts use the actual assignment, including linking persons and uneven or incomplete skeletons. A design cannot pass an active limit when its corresponding count is missing. Older evaluation objects can be re-summarized for total counts, but per-rater counts require an evaluation that recorded `MaxRatingsPerRater`.

These checks describe the evaluated assignments. With randomized assignments, an observed maximum does not guarantee that future assignments will respect the same limit. The limits constrain workload, not monetary cost or scoring time. Ranking among passing designs still follows `prefer`.

`summary()` method: Review a simulation-based design recommendation

For `summary()`: Recommendations screen the saved simulation aggregates under the requested criteria. The default targets non-Person facet calibration. For Person differentiation, request `facets = "Person"` when recommending the design. Neither target establishes classification accuracy, inter-rater agreement, a universal minimum sample size or precision outside the simulated population.

`checks` has one row per design, target facet and criterion, including unavailable metrics and inactive constraints. `DesignRow` links these rows to the complete, ranked `design_table`. `Selected` marks the first passing plan under the stored `prefer` order; it does not establish an optimum over unevaluated plans. `Pass` and selected plans are reused without recalculation. Threshold sources are recorded for new results; older saved results show `not_recorded`. No confidence intervals or selection-adjusted uncertainty are calculated. Retain the simulation's generating assumptions and replication evidence with the recommendation.

`plot()` method: Plot criteria behind a design recommendation

For `plot()`: Columns are ranked candidates, labelled by `DesignRow`; an asterisk marks the selected candidate. Rows combine the facet and criterion. Grey means unavailable and white means not requested, neither a passing result. Selection follows the stored ordering, not a new utility function. For custom ggplot/raincloud graphics, use `plot_data(x, "table")` or obtain full design metrics with `plot_data(x, "designs", format = "long", value_cols = c("MinReliability", "MaxRatings"))`. These aggregate metrics are not replication-level distributions for a raincloud plot.

Value

A list of class `mfrm_design_recommendation` with:

- `facet_table`: facet-level threshold checks, including design-variable alias columns when applicable
- `design_table`: design-level aggregated checks, including design-variable alias columns when applicable, `MaxRatings`, `MaxRatingsPerRater`, and workload checks `RatingsPass` and `RaterWorkloadPass`. Its `Pass` requires the facet-level checks, active workload limits, and `ConnectivityPass`. Component maxima, `DisconnectedReps`, `ConnectivityStatus`, and the separate `LinkReviewStatus` / `LinkReviewReason` explain structural checks.
- `recommended`: the first passing design after ranking
- `thresholds`: thresholds used in the recommendation
- `design_variable_aliases`: accepted public aliases for design variables
- `design_descriptor`: role-based design-variable metadata
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of mutable/locked design variables
- `planning_schema`: structured planning metadata

- caveats: fixed-effects and connectivity interpretations and suggested post-fit reviews

For summary(): summary() returns a summary.mfrm_design_recommendation list containing target, thresholds, checks, design_table, facet_table, recommended, and notes. Print methods return their input invisibly.

For plot(): An invisible mfrm_plot_data with table (long criterion outcomes), designs, targets, thresholds and interpretation notes.

Typical workflow

1. Run evaluate_mfrm_design().
2. Review summary.mfrm_design_evaluation() and plot.mfrm_design_evaluation().
3. Use recommend_mfrm_design(...) to identify the smallest acceptable design.

See Also

evaluate_mfrm_design(), summary.mfrm_design_evaluation, plot.mfrm_design_evaluation

recommend_mfrm_design(), plot.mfrm_design_recommendation(), mfrm_report(), export_summary_appendix()

summary.mfrm_design_recommendation(), as_ggplot(), plot_data()

Examples

```
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-design.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
sim_eval <- saved$result
rec <- recommend_mfrm_design(
  sim_eval, max_ratings = 48, max_ratings_per_rater = 24
)
rec$recommended
# Saved synthetic simulation; one replication illustrates structure only.
saved <- readRDS(system.file("examples", "simulation-design.rds",
  package = "mfrmr", mustWork = TRUE))
rec <- recommend_mfrm_design(saved$result, facets = "Person")
summary(rec)
report <- mfrm_report(rec)
report$tables$checks
saved <- readRDS(system.file("examples", "simulation-design.rds",
  package = "mfrmr", mustWork = TRUE))
rec <- recommend_mfrm_design(saved$result, facets = "Person")
plot(rec)
head(plot_data(rec, "table"))
```

`reference_case_benchmark` *Benchmark packaged reference cases*

Description

Benchmark packaged reference cases

Usage

```
reference_case_benchmark(
 cases = c("synthetic_truth", "synthetic_latent_regression", "synthetic_bias_contract",
    "study1_itercal_pair", "study2_itercal_pair", "combined_itercal_pair"),
  method = "MML",
  model = "RSM",
  quad_points = 7,
  maxit = 40,
  reltol = 1e-06,
  mml_engine = c("direct", "em", "hybrid")
)
```

Arguments

`cases`	Reference cases to run. Defaults to the standard `RSM`-compatible reference suite. Specialized `GPCM` and ConQuest-overlap package-side cases can be requested explicitly.
`method`	Estimation method passed to `fit_mfrm()`. Defaults to `"MML"`.
`model`	Model family passed to `fit_mfrm()`. Defaults to `"RSM"`.
`quad_points`	Quadrature points for `method = "MML"`.
`maxit`	Maximum optimizer iterations passed to `fit_mfrm()`.
`reltol`	Convergence tolerance passed to `fit_mfrm()`.
`mml_engine`	MML optimization engine passed to `fit_mfrm()`. Applies only when `method = "MML"`.

Details

This function checks `mfrmr` against the package's curated reference case families:

- `synthetic_truth`: checks whether recovered facet measures align with the known generating values from the package's synthetic design.
- `synthetic_latent_regression`: checks whether the latent-regression `MML` branch recovers known population coefficients, residual latent variance, criterion ordering, and posterior-shift direction from a synthetic overlap case.
- `synthetic_latent_regression_omit`: checks whether the population-model complete-case omission policy is reflected in the fitted metadata, response-row review, active person estimates, and replay provenance.

- synthetic_conquest_overlap_dry_run: builds the narrow ConQuest-overlap bundle for the latent-regression synthetic case, round-trips package tables through the normalization/review helpers, and confirms the package-side workflow without claiming that ConQuest itself was executed.
- synthetic_gpcm: checks whether the GPCM branch recovers known criterion-specific slopes, row-centered step parameters, and criterion ordering from a synthetic overlap case. This case currently requires model = "GPCM" and is intended for method = "MML".
- synthetic_bias_contract: checks whether package bias tables and pairwise local comparisons satisfy the identities documented in the bias help workflow.
- *_itercal_pair: compares a baseline packaged dataset with its iterative recalibration counterpart to review fit stability, facet-measure alignment, and linking coverage together.

The resulting object is intended as a reference-case check for package behavior. It does not by itself establish external validity against FACETS, ConQuest, or published calibration studies, and it does not assume any familiarity with external table numbering or printer layouts. When specialized latent-regression omission or ConQuest-overlap package-side cases are requested, summary(bench) prints preview rows from population_policy_checks and conquest_overlap_checks alongside the reference notes so the package-versus-external validation boundary remains visible.

Value

An object of class mfrm_reference_benchmark.

Interpreting output

- overview: one-row reference-case summary.
- case_summary: pass/warn/fail triage by reference case.
- fit_runs: fitted-run metadata (fit, precision tier, convergence, and latent-regression population-model/posterior-basis fields, including categorical-coding details when present).
- design_checks: exact design recovery checks for each dataset.
- recovery_checks: known-truth recovery metrics for the synthetic cases, including the latent-regression reference case. If a benchmark fit is not scoring-ready, its posterior-shift row is retained as Warn and is not calculated from a review-only score.
- bias_checks: source-backed bias/local-measure identity checks.
- pair_checks: paired-dataset stability screens for the iterated cases.
- linking_checks: common-element reviews for paired calibration datasets.
- conquest_overlap_checks: package-side checks for the ConQuest-overlap bundle/normalization/review workflow; this remains a package-side check until actual ConQuest output tables are supplied.
- population_policy_checks: complete-case omission checks for population model reference data.
- source_profile: source-backed rules used by the reference checks.

Examples

```
# Long-running workflow; run explicitly when needed.
# example("reference_case_benchmark", run.dontrun = TRUE)
## Not run:
bench <- reference_case_benchmark(
  cases = "synthetic_truth",
  method = "JML",
  maxit = 30
)
summary(bench)

## End(Not run)
```

`reference_case_review` *Build a package-native reference review for report completeness*

Description

Build a package-native reference review for report completeness

Usage

```
reference_case_review(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  reference_profile = c("core", "compatibility"),
  include_metrics = TRUE,
  top_n_attention = 15L
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. If omitted, diagnostics are computed internally with `residual_pca = "none"`.
`bias_results`	Optional output from `estimate_bias()`. If omitted and at least two facets exist, a 2-way interaction screen is computed internally.
`reference_profile`	Review profile. `"core"` emphasizes package-native report contracts. `"compatibility"` exposes the manual-aligned compatibility layer used by `facets_output_contract_review(branch = "facets")`.
`include_metrics`	If `TRUE`, run numerical consistency checks in addition to schema coverage checks.
`top_n_attention`	Number of lowest-coverage components to keep in `attention_items`.

Details

This function repackages the output-contract review into package-native terminology so users can review output completeness without needing external manual/table numbering. It reports:

- component-level schema coverage
- numerical consistency checks for derived report tables
- the highest-priority attention items for follow-up

It is a package-output completeness review, not an external validation study.

Use `reference_profile = "core"` for ordinary `mfrmr` workflows. Use `reference_profile = "compatibility"` only when you explicitly want to inspect the compatibility layer.

Value

An object of class `mfrm_reference_review`.

Interpreting output

- `overall`: one-row review summary with schema coverage and metric pass rate.
- `component_summary`: per-component coverage summary.
- `attention_items`: direct list of components needing review.
- `metric_summary` / `metric_checks`: numerical consistency status.

See Also

`facets_output_contract_review()`, `diagnose_mfrm()`, `build_fixed_reports()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("reference_case_review", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
review <- reference_case_review(fit, diagnostics = diag)
summary(review)

## End(Not run)
```

reporting_checklist *Build an auto-filled MFRM reporting checklist*

Description

Build an auto-filled MFRM reporting checklist

Usage

```
reporting_checklist(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  hierarchical_structure = NULL,
  include_references = TRUE
)

## S3 method for class 'mfrm_reporting_checklist'
summary(object, top_n = 10, ...)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are computed with residual_pca = "none".
bias_results	Optional output from estimate_bias() or a named list of such outputs.
hierarchical_structure	Optional output from analyze_hierarchical_structure() for matching retained rating rows. The reviewed facets may be a subset of the fitted facets. After checking the data identity, the "Hierarchical structure review" item is marked DraftReady = TRUE; Detail names the reviewed columns, row count, nested / crossed pairs and ICC availability. This indicates available material within that scope, not a complete design assessment.
include_references	If TRUE, include a compact reference table in the returned bundle.
object	Output from reporting_checklist().
top_n	Maximum number of draft-action rows shown in the compact action table.
...	Reserved for generic compatibility.

Details

This helper builds a package-native reporting checklist. It does not try to judge substantive reporting quality; instead, it checks whether the fitted object and related diagnostics contain the evidence typically reported in MFRM write-ups.

Checklist items are grouped into seven core sections:

- Method section
- Global fit
- Facet-level statistics
- Element-level statistics
- Rating scale diagnostics
- Bias/interaction analysis
- Visual displays

When a fit uses the latent-regression population-model branch, the checklist also adds a `Population Model` section covering coefficient reporting, categorical model-matrix coding, complete-case omissions, posterior-basis wording, and ConQuest scope wording.

The output is designed for manuscript preparation, reproducibility records, and reproducible reporting workflows.

`summary()` method: Summarize a reporting-checklist bundle for manuscript work

Value

A named list with checklist tables. Class: `mfrm_reporting_checklist`.

For `summary()`: An object of class `summary.mfrm_reporting_checklist` with:

- `overview`: run-level counts of available and draft-ready items
- `section_summary`: section-level checklist coverage
- `software_scope`: external-software relationship summary
- `facets_positioning`: report-ready FACETS relationship wording
- `visual_scope`: available plots and data for custom rendering, including the main `InterpretationCheck` caveat for each type of plot
- `priority_summary`: counts by priority/severity
- `action_items`: highest-priority rows that still need draft work
- `settings`: checklist settings rendered as a compact table
- `notes`: interpretation notes

What this checklist means

`reporting_checklist()` is a manuscript-preparation guide. It tells you which reporting elements are already present in the current analysis objects and which still need to be generated or documented. The primary draft-status column is `DraftReady`; `ReadyForAPA` is retained as a backward-compatible alias.

What this checklist does not justify

- It is not a single run-level pass/fail decision for publication.
- `DraftReady = TRUE` / `ReadyForAPA = TRUE` does not certify formal inferential adequacy.
- Missing bias rows may simply mean `bias_results` were not supplied.

- Study questions, recruitment, rater training, the assignment process, missingness reasons, ethics, and substantive interpretation require the author's study record. Available output does not verify those facts. The "Manuscript coverage map" in `vignette("mfrmr-reporting-and-apa", package = "mfrmr")` pairs reporting topics with numerical evidence and information to supply manually.

Interpreting output

- `checklist`: one row per reporting item with `Available = TRUE/FALSE`. `DraftReady = TRUE` means the item can be drafted into a report with the package's documented caveats. `ReadyForAPA` is a backward-compatible alias of the same flag; neither field certifies formal inferential adequacy.
- `fit_readiness`, `fit_readiness_components`, and `fit_readiness_parameters`: the fitted model's recorded numerical checks and restrictions on inference, overall and by component or parameter. Checklist completeness does not change these records.
- `section_summary`: available items by section.
- The Global Fit section includes a "Fit/separation reporting boundary" row that points to `precision_review_report()`, `fit_measures_table()`, and `facets_fit_review()` before users phrase fit, ZSTD, separation, or reliability claims.
- `software_scope`: external-software relationship summary for `mfrmr`, FACETS, ConQuest, and SPSS-style tabular handoffs.
- `facets_positioning`: report-ready wording that states `mfrmr` is not a FACETS numerical clone and separates native estimation from FACETS-style handoff or external-table review.
- `visual_scope`: available plots and their interpretation limits. It distinguishes 2D figures from category-surface data for custom rendering; data availability does not mean an interactive 3D renderer is provided. `InterpretationCheck` gives the main caveat for each type of plot.
- `references`: abbreviated background citations and topics when requested, not a complete bibliography. Verify full records for the methods used; use `citation("mfrmr")` for the installed software's citation.

Recommended next step

Review the rows with `Available = FALSE` or `DraftReady = FALSE`, then add the missing diagnostics, bias results, or narrative context before calling `build_apa_outputs()` for draft text generation. For `RSM` / `PCM` reporting runs where the MML population assumptions are defensible, the most complete package-native route is an `MML` fit plus `diagnose_mfrm(..., diagnostic_mode = "both")` so the checklist can see the legacy and strict marginal screens together. A JML route remains available when its estimand and incidental-parameter limitations better match the analysis purpose.

How this differs from operational review

`reporting_checklist()` is the manuscript/reporting branch of the package. Use it when the question is "what is still missing from the report?" rather than "which observations or links need follow-up?" For operational review:

- Use `build_misfit_casebook()` after `diagnose_mfrm()` when you need ranked misfit cases and grouping views for local follow-up.

- Use build_linking_review() after anchor/drift/chain helpers when you need operational linking triage rather than manuscript-oriented reporting tables.

Typical workflow

1. Fit with fit_mfrm(). For RSM / PCM reporting runs, prefer method = "MML".
2. Compute diagnostics with diagnose_mfrm(). For RSM / PCM, prefer diagnostic_mode = "both".
3. Run reporting_checklist() to see which reporting elements are already available from the current analysis objects.
4. If the issue is operational rather than manuscript-facing, branch to build_misfit_casebook() or build_linking_review() instead of treating reporting_checklist() as the single review hub.

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model. Structural reviews are checked against the reviewed data, not the model's estimates. Row order and unused columns do not affect this match. Reviews of additional input rows or unrecorded grouping columns remain standalone analyses; they cannot establish availability for this fit's checklist.

See Also

build_apa_outputs(), build_visual_summaries(), specifications_report(), data_quality_report(), build_misfit_casebook(), build_linking_review()

reporting_checklist(), summary.mfrm_apa_outputs

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("reporting_checklist", run.dontrun = TRUE)
## Not run:
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Use the diagnostics saved with this fit
diagnostics <- example_fit$diagnostics

# Which reporting items still need evidence or explanation?
checklist <- reporting_checklist(fit, diagnostics = diagnostics)
checklist$section_summary

# Review the missing items and their suggested next actions
subset(checklist$checklist, !DraftReady,
```

```
       c("Section", "Item", "DraftReady", "NextAction"))
# DraftReady is TRUE/FALSE; TRUE means draft material is available with caveats
# Choose follow-up analyses for your question, not merely to make every row TRUE

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("summary.mfrm_reporting_checklist", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
chk <- reporting_checklist(fit, diagnostics = diag)
summary(chk)

## End(Not run)
```

`response_time_review` *Review response-time patterns outside the MFRM likelihood*

Description

Build a descriptive response-time review table from the same long-format rating-event data used by `fit_mfrm()`. This helper does not fit a joint response-time model and does not change MFRM estimates. It summarizes response-time distributions, distributional rapid/slow flags, and person / facet / score-level response-time patterns for screening and reporting context.

Usage

```
response_time_review(
  data,
  person,
  facets = NULL,
  time,
  score = NULL,
  time_unit = "seconds",
  min_time = 0,
  rapid_threshold = NULL,
  slow_threshold = NULL,
  rapid_quantile = 0.05,
  slow_quantile = 0.95,
  rapid_rate_warn = 0.25,
  slow_rate_warn = 0.25,
  min_n_flag = 3L
)

## S3 method for class 'mfrm_response_time_review'
summary(object, top_n = 10L, ...)
```

```
## S3 method for class 'mfrm_response_time_review'
plot(x, type = c("distribution", "person", "facet", "score"), ...)
```

Arguments

data	A data.frame in long format with one row per observed rating event.
person	Column name for the person identifier.
facets	Optional character vector of facet columns to summarize.
time	Column name containing positive response times.
score	Optional ordered-score column. When supplied, score-level response-time summaries are returned.
time_unit	Label for the response-time unit, such as "seconds".
min_time	Minimum valid response time. Values must be strictly greater than this threshold; default 0.
rapid_threshold	Optional numeric response-time cutoff for rapid responses. When NULL, it is estimated from rapid_quantile.
slow_threshold	Optional numeric response-time cutoff for slow responses. When NULL, it is estimated from slow_quantile.
rapid_quantile	Quantile used when rapid_threshold = NULL.
slow_quantile	Quantile used when slow_threshold = NULL.
rapid_rate_warn	Group-level rapid-response rate that creates a descriptive flag; default 0.25.
slow_rate_warn	Group-level slow-response rate that creates a descriptive flag; default 0.25.
min_n_flag	Minimum group size before rapid/slow rates are flagged; default 3.
object	An object returned by response_time_review().
top_n	Number of top groups to retain in summary previews.
...	For plot(): View-specific controls from plot_response_time_review(). For summary(): Unused.
x	A mfrm_response_time_review object.
type	Plot type: "distribution", "person", "facet", or "score".

Details

Supply one row per timed event. A respondent's production time must not be repeated for every rater or criterion that scores that response; rater scoring time is a different observation. This helper does not model repeated events, censoring or a speed-accuracy relationship.

Numeric factor labels are interpreted as times, not factor codes. Group summaries retain InputRows, valid timed rows (N) and ExcludedRows. Rates describe valid rows only; groups with no valid times remain present with unavailable statistics. Missing facet or score labels are omitted from that grouping. Sample quantiles and user-specified cutoffs are descriptive; they do not diagnose rapid guessing, effort or rater quality. Equal cutoffs can flag the same event in both tails when times

are tied. FlaggedGroups counts distinct groups, while Flags counts rapid/slow rule crossings. Recreate older reviews from the original timing data before summary or plotting; no MFRM refit is needed.

summary() method: Summarize a response-time review

plot() method: Plot a saved response-time review

Value

An object of class mfrm_response_time_review, a list with overview, thresholds, observations, person_summary, facet_summary, score_summary, flags, and notes.

For summary(): A list of class summary.mfrm_response_time_review.

For plot(): The existing response-time plot payload. These are descriptive timing summaries.

See Also

plot(x), fit_mfrm(), mfrmr_visual_diagnostics, mfrmr_output_guide()

response_time_review(), mfrmr_plot_migration

Examples

```
toy <- load_mfrmr_data("example_core")
toy$ResponseTime <- 12 + as.numeric(factor(toy$Person)) * 0.4 +
  as.numeric(toy$Score)
rt <- response_time_review(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  time = "ResponseTime"
)
summary(rt)
plot(rt, draw = FALSE)
```

---

review_accessors *Extract canonical review components*

Description

Extract canonical review components

Usage

```
anchor_review(x, required = TRUE)

precision_review(x, required = TRUE)
```

Arguments

x	A fitted mfrm_fit, diagnostics object, summary object, or compatible list containing the requested review component.
required	Logical. If TRUE, error when no compatible review component is found. If FALSE, return NULL instead.

Details

anchor_review() returns the fitted object's config$anchor_review component. precision_review() returns the diagnostics precision_review table. These helpers intentionally do not search older field names.

Value

anchor_review() returns an mfrm_anchor_review-like object when available. precision_review() returns the precision-review table when available. If required = FALSE and no component is available, both helpers return NULL.

Examples

```
fit_like <- list(config = list(
  anchor_review = structure(list(issue_counts = data.frame()),
                            class = "mfrm_anchor_review")
))
anchor_review(fit_like)
diag_like <- list(
  precision_review = data.frame(Check = "SE", Status = "ok", Detail = "Model-based")
)
precision_review(diag_like)
```

---

review_conquest_overlap

Review a ConQuest comparison within the documented mfrmr *overlap scope*

Description

Review a ConQuest comparison within the documented mfrmr overlap scope

Usage

```
review_conquest_overlap(
  bundle,
  conquest_population = NULL,
  conquest_item_estimates = NULL,
  conquest_case_eap = NULL,
  conquest_population_term = "auto",
```

```
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
  conquest_item_estimate = "auto",
  item_id_source = c("auto", "response_var", "level"),
  conquest_case_person = "auto",
  conquest_case_estimate = "auto",
  conquest_location_shift = NULL
)
```

Arguments

bundle	Output from build_conquest_overlap_bundle().
conquest_population	Normalized ConQuest population-parameter table as a data.frame, or output from normalize_conquest_overlap_exports(), normalize_conquest_overlap_files(), or normalize_conquest_overlap_tables().
conquest_item_estimates	Normalized ConQuest item-estimate table as a data.frame. Leave NULL when conquest_population is an object from one of the ConQuest normalization helpers.
conquest_case_eap	Normalized ConQuest case-level EAP table as a data.frame. Leave NULL when conquest_population is an object from one of the ConQuest normalization helpers.
conquest_population_term	Column in conquest_population that stores parameter names. "auto" tries conservative aliases such as Parameter and Term.
conquest_population_estimate	Column in conquest_population that stores parameter estimates. "auto" tries aliases such as Estimate and Est.
conquest_item_id	Column in conquest_item_estimates that stores the item identifier. This may be the exported response variable (for example I001) or the original item/facet level. "auto" tries aliases such as ResponseVar, ItemID, Item, and Label.
conquest_item_estimate	Column in conquest_item_estimates that stores the item estimate. "auto" tries Estimate and Est. Other columns must be specified explicitly and converted to difficulty logits before review.
item_id_source	How conquest_item_id should be matched. "auto" chooses the larger overlap between exported response variables and original item levels, with ties resolved toward exported response variables.
conquest_case_person	Column in conquest_case_eap that stores person IDs. "auto" tries conservative aliases such as Person, PID, and Sequence ID.
conquest_case_estimate	Column in conquest_case_eap that stores case EAP estimates. "auto" tries conservative aliases such as Estimate, EAP_1, and EAP.

conquest_location_shift
: Known additive location offset, ConQuest minus mfrmr. Use `0` only after confirming a common origin. `NULL` uses zero when the supplied generated report confirms the common ITEMS constraint; otherwise aligned EAP/intercept comparisons are unavailable. This argument does not change scale units or reverse the item sign.

Details

This helper compares normalized ConQuest output tables against the exact- overlap bundle produced by `build_conquest_overlap_bundle()`. It is intentionally conservative:

- it does **not** parse raw ConQuest text output automatically;
- its primary route expects output from `normalize_conquest_overlap_exports()`, which reads the four native comparison CSV files requested by the generated command; the additional matrixout-history CSV is retained for a separate objective/free-dimension verification; custom extracted tables may use `normalize_conquest_overlap_files()` or `normalize_conquest_overlap_tables`
- and it reports numerical differences and missing elements without claiming that any fixed tolerance implies software equivalence.

This is the package's external-table review path. It is distinct from `reference_case_benchmark(cases = "synthetic_conquest_overlap_dry_run")`, which only round-trips package-native tables through the same normalization and review contract without executing ConQuest.

The intended workflow is:

1. export a bundle for the documented comparison scope with `build_conquest_overlap_bundle()`;
2. run the narrow matching case in ConQuest;
3. normalize the four comparison CSV files with `normalize_conquest_overlap_exports()`, supplying the generated review text as `report_file` for stopping/model metadata;
4. pass those tables here to inspect direct differences, centered item agreement, and case-level EAP agreement.

Value

A named list with class `mfrm_conquest_overlap_review`.

Output

The returned object has class `mfrm_conquest_overlap_review` and includes:

- `overall`: one-row comparison summary with missing/duplicate/non-numeric attention-item counts and worst-row labels
- `population_comparison`: parameter-by-parameter comparison table
- `item_comparison`: centered item-estimate comparison table
- `case_comparison`: case-level EAP comparison table
- `attention_items`: missing, malformed, or unmatched elements
- `settings`: review settings
- `notes`: interpretation notes

Interpretation

- Read `summary(review)$review_scope` first to confirm that the result is a supplied-table review, not raw ConQuest text parsing or a software- equivalence claim.
- Population slopes and `sigma2` are intended for direct comparison.
- Both item tables are centered using the same matched finite items. `MfrmrCentered` is this comparison coordinate; `CenteredEstimate` retains the native bundle's original full-set coordinate. `ItemCenteringRows` records the reference-set size. At least two items are needed; otherwise centered differences and summary metrics are unavailable. Missing items remain flagged, and partial-set agreement does not imply full-set agreement. Recompute older reviews from the saved bundle and external tables to apply this common reference set; printing an old review does not change its values.
- Case estimates are posterior EAP summaries under the fitted population model. `Difference` retains the raw external-minus-native value; `AlignedDifference` subtracts the declared location shift. `CaseMae` and `CaseMaxAbsDifference` use aligned differences and are unavailable when the origin is unknown. `CaseRawMae` and `CaseRawMaxAbsDifference` remain descriptive. Do not infer a common origin from partial centered items.
- Population summary errors cover slopes and variance; the intercept has separate `InterceptCompared` and `InterceptAlignedDifference` fields.
- `ExternalEstimationStatus` is unknown without a supplied report. `criterion_met` does not certify precision; `ExternalPrecisionStatus` remains `not_assessed`. Failed stopping and surplus external IDs are attention items. Recreate old reviews to apply these rules.
- The `overall` table reports both mean and maximum absolute differences for compared population, centered item, and case rows. The `PopulationMaxAbsParameter`, `ItemCenteredMaxAbsItem`, and `CaseMaxAbsPerson` columns identify the row where each maximum absolute difference occurs.
- Missing or non-numeric rows in `attention_items` indicate that the external tables do not align cleanly with the exported overlap bundle. Non-finite estimates also use the existing `NonNumericInConQuest` status and are excluded from comparison metrics.
- This review does not compare MnSq, ZSTD, threshold crossings or full parameter covariance. Native MML row residuals evaluated at EAP are not ConQuest parameter-fit statistics: those aggregate through the response design and average over posterior distributions. Adjacent step parameters are not ConQuest cumulative-probability (Thurstonian) thresholds. See the ConQuest technical manual.

See Also

`build_conquest_overlap_bundle()`, `normalize_conquest_overlap_files()`, `normalize_conquest_overlap_tables`
`reference_case_benchmark()`

Examples

```
## Not run:
bundle <- build_conquest_overlap_bundle()
normalized <- normalize_conquest_overlap_exports(
  bundle,
  parameter_file = "conquest_overlap_conquest_parameters.csv",
```

```
  regression_file = "conquest_overlap_conquest_reg_coefficients.csv",
  covariance_file = "conquest_overlap_conquest_covariance.csv",
  case_file = "conquest_overlap_conquest_cases_eap.csv",
  report_file = "conquest_overlap_conquest_parameters_review.txt"
)
review <- review_conquest_overlap(bundle, normalized)
summary(review)$summary

## End(Not run)
```

`review_mfrm_anchors` *Review and normalize anchor/group-anchor tables*

Description

Review and normalize anchor/group-anchor tables

Usage

```
review_mfrm_anchors(
  data,
  person,
  facets,
  score,
  anchors = NULL,
  group_anchors = NULL,
  weight = NULL,
  rating_min = NULL,
  rating_max = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  min_common_anchors = 5L,
  min_obs_per_element = 30,
  min_obs_per_category = 10,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  category_policy = NULL,
  repeated_ratings = c("review", "independent"),
  centering_scope = c("facet", "remainder", "anchor_origin")
)

## S3 method for class 'mfrm_anchor_review'
summary(object, digits = 3, top_n = 10, ...)

## S3 method for class 'mfrm_anchor_review'
plot(
  x,
  y = NULL,
```

```
  type = c("issue_counts", "facet_constraints", "level_observations"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

data	A data.frame in long format (one row per rating event).
person	Column name for person IDs.
facets	Character vector of facet column names.
score	Column name for observed score.
anchors	Optional direct-anchor table (Facet, Level, Anchor). Retained values are fixed during estimation.
group_anchors	Optional group-anchor table (Facet, Level, Group, GroupValue) defining group-mean constraints.
weight	Optional weight/frequency column name.
rating_min	Optional minimum category value.
rating_max	Optional maximum category value.
keep_original	Keep original category values. New code can instead use category_policy = "preserve".
missing_codes	Optional. NULL (default) is a no-op; TRUE or "default" converts the FACETS / SPSS / SAS sentinel set to NA on the score column before review while preserving person and facet identifiers. Supply a character vector to apply a custom code set across the person, facet, and score columns.
min_common_anchors	Minimum directly anchored levels per non-Person facet used in the package count recommendation (default 5). The function cannot verify that those levels have invariant cross-run identity.
min_obs_per_element	Minimum weighted observations per facet level used in recommendations (default 30).
min_obs_per_category	Minimum weighted observations per score category used in recommendations (default 10).
noncenter_facet	One facet to leave non-centered. Other facets use centering_scope, as in fit_mfrm().
dummy_facets	Facets to fix at zero.
category_policy	Optional explicit category choice: "collapse" maps gaps in the observed categories to consecutive scores; "preserve" keeps the intended ladder, declared

with rating_min and rating_max. This changes the fitted category steps, not just labels. NULL (default) uses keep_original, whose default is FALSE ("collapse"). Supplying both choices is allowed only when they agree. Preservation does not estimate unsupported steps: fitting stops if a retained internal category has no observations. Use the same policy in describe_mfrm_data() and review_mfrm_anchors().

repeated_ratings
: Treatment of repeated Person-by-facet cells. "review" (default) retains the duplicate-cell warning and ordinary MML/JML input-review state. Use "independent" only to declare that the rows are distinct, conditionally independent rating events under the specified model. Every row and its weight are retained; this argument does not aggregate scores, add a facet, change the likelihood or establish independence from the data. Other input, numerical and inference checks still apply. The declaration is saved in preparation notes, configuration and replay settings. Corrected JML already requires conditionally independent repeated ratings; the argument records that assumption explicitly there. It is not supported by the two-family GPCM route.

centering_scope
: "facet" (default) centers each declared facet across all its levels, including direct anchors and group totals. "remainder" reproduces the earlier convention that centers only levels outside direct and group constraints. For ordinary additive RSM/PCM JML, "anchor_origin" lets direct or group anchors replace centering on their facet; select an unanchored, active noncenter_facet. Other facets remain centered. This option excludes comparison-only fitting. None of these conventions changes the population model or category steps. Unsupported specialized routes reject an explicit setting. See **Anchors and scale constraints**.

object
: Output from review_mfrm_anchors().

digits
: Number of digits for numeric rounding.

top_n
: Maximum rows shown in issue previews.

...
: Reserved for generic compatibility.

x
: Output from review_mfrm_anchors().

y
: Reserved for generic compatibility.

type
: Plot type: "issue_counts", "facet_constraints", or "level_observations".

main
: Optional title override.

palette
: Optional named colors.

label_angle
: X-axis label angle for bar plots.

draw
: If TRUE, draw using base graphics.

Details

This helper reviews computational constraints. A *direct anchor* fixes an individual parameter to a supplied value. A *group anchor* constrains the mean of declared elements to a supplied target. A *common-element link* is different: it is observed overlap among administrations or subsets. Direct or group constraints can transfer coordinates from a defensible reference, but they do not manufacture empirical overlap.

This function applies the same preprocessing and key-resolution rules as `fit_mfrm()`, but returns a review object so constraints can be checked *before* estimation. Running the review first helps avoid estimation failures caused by misspecified or data-incompatible anchors. It does not inspect source-fit readiness or establish that labels denote the same invariant elements across runs. Those substantive checks remain the caller's responsibility. This review has no estimation-method argument. A clean table can still request an unsupported model combination: in particular, `fit_mfrm()` rejects Person direct/group anchors with MML. See the scale-constraint section of `fit_mfrm()` before using partial anchors.

Anchor types:

- *Direct anchors* fix individual element measures to specific logit values (e.g., Rater R01 anchored at 0.35 logits). Use internal coordinates, not values transformed for display with `Umean` or `Uscale`.
- *Group anchors* constrain the mean of a set of elements to a target value, allowing individual elements to vary freely around that mean. The target requires an external justification such as a defensible equal-mean or known-scale assumption. The mean is unweighted over declared levels, not over rating rows. `GroupValue` repeats one target per group. It is not a FACETS per-element contribution: values 0.2 and 1.0 in a native two-member group translate to 0.6 for both rows. See `fit_mfrm()` for group-origin centering and FACETS group 0 / extreme-exclusion differences.
- When both types include the same element, both constraints are retained: the direct anchor fixes that element and the group constraint still fixes the declared group mean. Incompatible combinations are rejected by the constraint/estimability checks.

Design checks report whether each observed facet level has at least `min_obs_per_element` weighted observations (default 30) and each score category has at least `min_obs_per_category` (default 10). These user-configurable counts are package screening recommendations, not universal adequacy thresholds and not tests of connectedness or invariance.

`summary()` method: Summarize an anchor-review object

For `summary()`: This summary provides a compact pre-estimation review of anchor and group-anchor specifications.

`plot()` method: Plot an anchor-review object

For `plot()`: Base-R visualization helper for anchor-review outputs.

Value

A list of class `mfrm_anchor_review` with:

- `anchors`: cleaned direct constraints used by estimation
- `group_anchors`: cleaned group-mean constraints used by estimation
- `centering_scope`: the reviewed `"facet"`, `"remainder"` or `"anchor_origin"` convention
- `facet_summary`: counts of levels inside/outside declared constraints, `Centered` indicating the declared centering constraint, and `OriginBasis` describing how the facet's origin is set
- `design_checks`: observation-count checks by level/category
- `thresholds`: active threshold settings used for recommendations
- `issue_counts`: issue-type counts

- issues: list of issue tables
- recommendations: package-native anchor guidance strings

For summary(): An object of class summary.mfrm_anchor_review.

For plot(): A plotting-data object of class mfrm_plot_data.

Interpreting output

- issue_counts/issues: concrete data or specification problems.
- facet_summary: constraint coverage by facet. FreeLevels counts levels outside direct and group constraints, not free parameters. A group member may still vary, whereas centering removes a degree of freedom outside groups.
- design_checks: observation counts against the requested screening values.
- recommendations: action items before estimation.

For summary():

Recommended order:

- issue_counts: primary triage table (non-zero issues first).
- facet_summary: anchored/grouped/free-level balance by facet.
- level_observation_summary and category_counts: sparse-cell diagnostics.
- recommendations: concrete remediation suggestions.

If issue_counts is non-empty, treat anchor constraints as provisional and resolve issues before final estimation.

For plot():

- "issue_counts": volume of each issue class.
- "facet_constraints": anchored/grouped/free mix by facet.
- "level_observations": observation support across levels.

Typical workflow

1. Build candidate anchors (e.g., with make_anchor_table()).
2. Run review_mfrm_anchors(...).
3. Resolve issues, then fit with fit_mfrm().

For summary():

1. Run review_mfrm_anchors() with intended anchors/group anchors.
2. Review summary(review) and recommendations.
3. Revise anchor tables, then call fit_mfrm().

For plot():

1. Run review_mfrm_anchors().
2. Start with plot(review, type = "issue_counts").
3. Inspect constraint and support plots before fitting.

See Also

fit_mfrm(), describe_mfrm_data(), make_anchor_table(), mfrmr_linking_and_dff

review_mfrm_anchors(), fit_mfrm()

review_mfrm_anchors(), make_anchor_table()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("review_mfrm_anchors", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_operational")
# Illustrative external value, not an estimate supplied with this dataset.
anchors <- data.frame(Facet = "Rater", Level = "R01", Anchor = 0.35)
review <- review_mfrm_anchors(
  data = ratings,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  anchors = anchors,
  rating_min = 1, rating_max = 4, category_policy = "preserve"
)
review$anchors                 # Check the retained value and matching ID.
review$issue_counts
review$recommendations         # Explains the selected centering convention.

# A duplicate is a separate input problem: resolve it before fitting.
duplicate <- rbind(anchors, transform(anchors, Anchor = 0.10))
problem <- review_mfrm_anchors(
  ratings, "Person", c("Rater", "Criterion"), "Score",
  anchors = duplicate
)
problem$issues$duplicate_anchors
# Normalization retains the last value; it does not decide which is correct.

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_anchor_review", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
review <- review_mfrm_anchors(toy, "Person", c("Rater", "Criterion"), "Score")
summary(review)

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("plot.mfrm_anchor_review", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
review <- review_mfrm_anchors(toy, "Person", c("Rater", "Criterion"), "Score")
p <- plot(review, draw = FALSE)

## End(Not run)
```

`review_peer_design` *Review a peer-assessment assignment design*

Description

Review existing assignment metadata for self-review, reviewer/submission load and shared submissions. This function does not create assignments, simulate scores or fit a measurement model. In 0.2.4 it replaces `build_peer_review_design_review()` with the same arguments and result class.

Usage

```
review_peer_design(peer_review_design, top_n = 10)

## S3 method for class 'mfrm_peer_review_design_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`peer_review_design`	A generated data frame carrying the `mfrm_peer_review_design` attribute, the attribute itself, or its `overview` data frame.
`top_n`	For `review_peer_design()`: Number of reviewer-load, submission-load, common-link, and reciprocal-pair rows to keep in compact summary tables. For `summary()`: Number of rows to keep in compact follow-up tables.
`object`	Output from `review_peer_design()`.
`digits`	Number of digits for printed numeric values.
`...`	Reserved for generic compatibility.

Details

`review_peer_design()` converts peer-review simulation metadata into a reportable design-review object. The review summarizes self-review checks, reviewer and submission load, common submissions per reviewer pair, and reciprocal review pairs. These rows are assignment-design diagnostics: they do not replace MFRM estimates, fit, separation, reliability, or substantive review-quality evidence.

Peer-review use of MFRM follows studies that model peer/self/teacher rater severity and leniency. Common-link anchor interpretation follows sparse rater-mediated design work; the review status is therefore descriptive and conservative rather than a literature-derived universal adequacy cutoff.

`summary()` method: Summarize a peer-review design review

Value

A bundle of class `mfrm_peer_review_design_review`.

For `summary()`: An object of class `summary.mfrm_peer_review_design_review`.

References

- Farrokhi, F., Esfandiari, R., & Schaefer, E. (2012). A many-facet Rasch measurement of differential rater severity/leniency in three types of assessment. *JALT Journal*, 34(1), 79-102. doi:10.37546/JALTJJ34.1-3.
- Uto, M., & Ueno, M. (2020). A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo. *Behaviormetrika*, 47, 469-496. doi:10.1007/s41237-020-00115-7.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

See Also

`build_peer_review_sim_spec()`, `simulate_mfrm_data()`, `build_mfrm_network_review()`, `build_summary_table_bu`

`review_peer_design()`

Examples

```
peer_spec <- build_peer_review_sim_spec(
  n_submission = 12,
  n_criterion = 3,
  reviewers_per_submission = 2,
  anchor_submissions = 2
)
peer_sim <- simulate_mfrm_data(sim_spec = peer_spec, seed = 123)
review <- review_peer_design(peer_sim)
summary(review)$overview
```

`run_mfrm_facets` *Run a legacy-compatible estimation workflow wrapper*

Description

This helper mirrors `mfrmRFacets.R` behavior as a package API and keeps legacy-compatible defaults (`model = "RSM"`, `method = "JML"`), while allowing users to choose compatible estimation options.

Usage

```
run_mfrm_facets(
  data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  keep_original = FALSE,
  model = c("RSM", "PCM"),
  method = c("JML", "JMLE", "MML"),
```

```
  step_facet = NULL,
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  positive_facets = NULL,
  quad_points = 15,
  maxit = 400,
  reltol = 1e-09,
  optimizer = c("auto", "BFGS", "L-BFGS-B"),
  mml_engine = c("direct", "em", "hybrid"),
  top_n_interactions = 20L,
  mml_integration = c("fixed", "adaptive"),
  centering_scope = c("facet", "remainder", "anchor_origin")
)

mfrmRFacets(
  data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  keep_original = FALSE,
  model = c("RSM", "PCM"),
  method = c("JML", "JMLE", "MML"),
  step_facet = NULL,
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  positive_facets = NULL,
  quad_points = 15,
  maxit = 400,
  reltol = 1e-09,
  mml_engine = c("direct", "em", "hybrid"),
  top_n_interactions = 20L,
  mml_integration = c("fixed", "adaptive"),
  centering_scope = c("facet", "remainder", "anchor_origin")
)

## S3 method for class 'mfrm_facets_run'
plot(x, y = NULL, type = c("fit", "qc"), ...)
```

Arguments

data	A data.frame in long format.
person	Optional person column name. If NULL, guessed from names.
facets	Optional facet column names. If NULL, inferred from remaining columns after

	person/score/weight mapping.
score	Optional score column name. If NULL, guessed from names.
weight	Optional weight column name.
keep_original	Passed to fit_mfrm().
model	MFRM model ("RSM" default, or "PCM").
method	Estimation method ("JML" default; "JMLE" and "MML" also supported).
step_facet	Step facet for PCM mode; passed to fit_mfrm().
anchors	Optional anchor table (data.frame).
group_anchors	Optional group-anchor table (data.frame).
noncenter_facet	Non-centered facet passed to fit_mfrm().
dummy_facets	Optional dummy facets fixed at zero.
positive_facets	Optional facets with positive orientation.
quad_points	Quadrature points for MML; passed to fit_mfrm().
maxit	Maximum optimizer iterations.
reltol	Relative optimizer tolerance passed to fit_mfrm(). The default is 1e-9.
optimizer	Direct optimizer passed to fit_mfrm(). "auto" selects L-BFGS-B for MML and JML fits with at least 200 free parameters, and BFGS for smaller JML fits.
mml_engine	MML optimization engine passed to fit_mfrm(). Applies only when method = "MML".
top_n_interactions	Number of rows for interaction diagnostics.
mml_integration	MML integration mode passed to fit_mfrm(): "fixed" (default) or "adaptive". Adaptive integration requires the direct MML engine.
centering_scope	Whole-facet (default), legacy remainder, or explicit direct/group-anchor-origin centering for ordinary additive RSM/PCM JML; passed to fit_mfrm().
x	A mfrm_facets_run object from run_mfrm_facets().
y	Unused.
type	Plot route: "fit" delegates to plot.mfrm_fit() and "qc" delegates to plot(x, type = "qc").
...	Additional arguments passed to the selected plot function.

Details

run_mfrm_facets() is intended as a one-shot workflow helper: fit -> diagnostics -> key report tables. Returned objects can be inspected with summary() and plot().

plot() method: Plot outputs from a legacy-compatible workflow run

For plot(): This method is a router for fast visualization from a one-shot workflow result:

- type = "fit" for model-level displays.
- type = "qc" for multi-panel quality-control diagnostics.

Value

A list with components:

- `fit`: `fit_mfrm()` result
- `diagnostics`: `diagnose_mfrm()` result
- `iteration`: `estimation_iteration_report()` result
- `fair_average`: `fair_average_table()` result
- `rating_scale`: `rating_scale_table()` result
- `run_info`: run metadata table
- `mapping`: resolved column mapping

For `plot()`: A plotting object from the delegated plot route.

Estimation-method notes

- `method = "JML"` (default): legacy-compatible joint estimation route; the default preserves the FACETS-style output continuity that existing one-shot scripts expect. For new analysis scripts, prefer `fit_mfrm(..., method = "MML")` – MML is the package-wide recommended route because person parameters are integrated out under an N(0, 1) prior and per-person posterior SEs are available.
- `method = "JMLE"`: backward-compatible input alias for the JML route; fitted objects and generated outputs use the canonical `"JML"` label.
- `method = "MML"`: marginal maximum likelihood route using `quad_points`. Use `mml_engine = "em"` or `"hybrid"` only for `RSM` / `PCM` fits when you want the staged MML alternatives.

`model = "PCM"` is supported; set `step_facet` when facet-specific step structure is needed.

Visualization

- `plot(out, type = "fit")` delegates to `plot.mfrm_fit()` and returns fit-level visual bundles (e.g., Wright/pathway/CCC).
- `plot(out, type = "qc")` delegates to plot(x, type = "qc") and returns a QC dashboard plot object.

Interpreting output

Start with `summary(out)`:

- check convergence and iteration count in `overview`.
- confirm resolved columns in `mapping`.

Then inspect:

- `out$rating_scale` for category/threshold behavior.
- `out$fair_average` for observed-vs-model scoring tendencies.
- `out$diagnostics` for misfit/reliability/interactions.

For `plot()`:

Returns the plotting object produced by the delegated route: `plot.mfrm_fit()` for `"fit"` and plot(x, type = "qc") for `"qc"`.

Typical workflow

1. Run `run_mfrm_facets()` with explicit column mapping.
2. Check `summary(out)` and `summary(out$diagnostics)`.
3. Visualize with `plot(out, type = "fit")` and `plot(out, type = "qc")`.
4. Export selected tables for reporting (`out$rating_scale`, `out$fair_average`).

For `plot()`:

1. Run `run_mfrm_facets()`.
2. Start with `plot(out, type = "fit", draw = FALSE)`.
3. Continue with `plot(out, type = "qc", draw = FALSE)` for diagnostics.

Preferred route for new analyses

For new scripts, prefer the package-native route: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()` -> `build_apa_outputs()`. Use `run_mfrm_facets()` when you specifically need the legacy-compatible one-shot wrapper.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `estimation_iteration_report()`, `fair_average_table()`, `rating_scale_table()`, mfrmr_visual_diagnostics, mfrmr_workflow_methods, mfrmr_compatibility_layer

`run_mfrm_facets()`, `plot.mfrm_fit()`, plot(x, type = "qc"), mfrmr_visual_diagnostics, mfrmr_workflow_methods

Examples

```
# Long-running workflow; run explicitly when needed.
# example("run_mfrm_facets", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]

# Legacy-compatible default: RSM + JML
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
out$fit$summary[, c("Model", "Method", "MethodUsed")]
s <- summary(out)
s$overview[, c("Model", "Method", "Converged")]
p_fit <- plot(out, type = "fit", draw = FALSE)
p_fit$wright_map$data$plot

# Optional: MML route
if (interactive()) {
  out_mml <- run_mfrm_facets(
    data = toy_small,
```

```
    person = "Person",
    facets = c("Rater", "Criterion"),
    score = "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
  out_mml$fit$summary[, c("Model", "Method", "MethodUsed")]
}

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_facets_run", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
p_fit <- plot(out, type = "fit", draw = FALSE)
p_fit$wright_map$data$plot
p_qc <- plot(out, type = "qc", draw = FALSE)
p_qc$data$plot


## End(Not run)
```

`run_qc_pipeline` *Run automated quality control pipeline*

Description

Integrates convergence, model fit, reliability, separation, element misfit, unexpected responses, category structure, connectivity, inter-rater agreement, and DIF/bias into a single pass/warn/fail report.

Usage

```
run_qc_pipeline(
  fit,
  diagnostics = NULL,
  threshold_profile = "standard",
  thresholds = NULL,
  rater_facet = NULL,
  include_bias = TRUE,
  bias_results = NULL,
```

```
  separation_facets = NULL
)

## S3 method for class 'mfrm_qc_pipeline'
plot(x, type = c("traffic_light", "detail"), draw = TRUE, ...)
```

Arguments

fit	Native output from fit_mfrm(). Imported measurement tables do not contain the response-level information required by this pipeline.
diagnostics	Output from diagnose_mfrm(). Computed automatically if NULL.
threshold_profile	Threshold preset: "strict", "standard" (default), or "lenient".
thresholds	Named list to override individual thresholds.
rater_facet	Character name of the rater facet for inter-rater check (detected from rater-like facet names if NULL; otherwise supply it explicitly).
include_bias	If TRUE and bias available in diagnostics, check DIF/bias.
bias_results	Optional pre-computed bias results from estimate_bias().
separation_facets	Character names of non-Person facets whose levels you intend to distinguish. Reliability/separation thresholds apply only to these facets. With the default NULL, these two checks are not requested; they do not affect the overall verdict. For example, use "Criterion" only when distinguishing criterion difficulties is a substantive goal.
x	Output from run_qc_pipeline().
type	Plot type: "traffic_light" (default) or "detail".
draw	If FALSE, return plot data invisibly without drawing.
...	Additional base-graphics arguments for the selected view.

Details

The pipeline evaluates 10 quality checks and assigns a verdict (Pass / Warn / Fail / Skip) to each. The overall status is the most severe verdict among checks marked AffectsOverall. Unrequested differentiation and bias checks, and inapplicable rater-agreement checks, are excluded. Missing information for an applicable check cannot produce Pass. Diagnostics are computed automatically via diagnose_mfrm() if not supplied.

All thresholds are screening rules, not universal statistical acceptance criteria. High rater separation indicates distinguishable rater measures, not high agreement; low separation may reflect similar rater severity. Select separation_facets only when differentiation is the intended target. A Pass does not establish model validity, adequate SEs, or anchor invariance.

Reliability and separation are used here as QC signals. In mfrmr, Reliability / Separation are model-based facet indices and RealReliability / RealSeparation use fit-inflated SEs. They are smaller or equal on the same levels; they are not statistical confidence bounds. For MML, these rely on model-based ModelSE values for non-person facets; for JML, they remain exploratory approximations. Reliability and separation come from the same variance decomposition; they are not independent evidence.

Category counts are sums of observation weights when weights are used. Threshold ordering compares adjacent numbered steps within each threshold family; missing steps or families require review. Equal estimates are nondecreasing within numerical tolerance. A binary scale has no adjacent threshold comparison, so its category screen evaluates counts only. These checks do not establish category adequacy or justify automatic merging.

Three threshold presets are available via `threshold_profile`:

Aspect	strict	standard	lenient
Global fit warn	1.3	1.5	1.7
Global fit fail	1.5	2.0	2.5
Reliability pass	0.90	0.80	0.70
Separation pass	3.0	2.0	1.5
Misfit warn (pct)	3	5	10
Unexpected fail	3	5	10
Min cat count	15	10	5
Agreement pass	60	50	40
Bias fail (pct)	5	10	15

Individual thresholds can be overridden via the `thresholds` argument (a named list using entries such as `global_fit_warn` or `reliability_pass`).

For `GPCM`, this pipeline is available as caveated operational triage over supported diagnostics. Its pass/warn/fail labels remain package QC policy overlays; they are not FACETS score-side equivalence, operational scoring decisions, design-forecasting evidence, or automatic fairness / validity decisions.

`plot()` method: Plot a saved QC pipeline

Display the checks already returned by `run_qc_pipeline()`. Use the colored verdicts for triage, then read the returned detail for the reason and next review.

For `plot()`: Two plot types are provided for visual triage of QC results:

- `"traffic_light"` (default): A horizontal bar chart with one row per QC check. Bars are coloured green (Pass), amber (Warn), or red (Fail). Provides an at-a-glance summary of the current QC review state.
- `"detail"`: A panel showing the saved `Detail` text beside each verdict. Read this explanation and the returned table to understand why a check was flagged or unavailable.

Value

Object of class `mfrm_qc_pipeline` with verdicts, overall status, details, and recommendations.

For `plot()`: Invisible verdicts tibble from the QC pipeline.

QC checks

The 10 checks are:

1. **Convergence**: Did the model converge?
2. **Global fit**: Infit/Outfit MnSq within the current review band.

3. **Reliability**: Minimum index across explicitly selected differentiation facets.
4. **Separation**: Minimum index across those same selected facets.
5. **Element misfit**: Percentage of elements with Infit/Outfit outside the current review band.
6. **Unexpected responses**: Percentage of observations with large standardized residuals.
7. **Category structure**: Minimum category count and threshold ordering.
8. **Connectivity**: All observations in a single connected subset.
9. **Inter-rater agreement**: Exact agreement percentage for the rater facet (if applicable).
10. **Functioning/Bias screen**: Percentage of interaction cells that cross the screening threshold (if interaction results are available).

Interpreting output

- `$overall`: character string `"Pass"`, `"Warn"`, or `"Fail"`.
- `$verdicts`: tibble with columns `Check`, `Verdict`, `Value`, and `Threshold` for each of the 10 checks.
- `$verdicts$Detail`: human-readable explanation of each assessment.
- `$verdicts$AffectsOverall`: whether a check contributes to the overall result.
- `$details`: named list of per-check numeric details for programmatic access.
- `$recommendations`: character vector of actionable suggestions for checks that need review; suggestions do not prescribe deleting raters or collapsing categories automatically.
- `$config`: records the threshold profile and effective thresholds.

Typical workflow

1. Fit a model: `fit <- fit_mfrm(...)`.
2. Optionally compute diagnostics and bias: `diag <- diagnose_mfrm(fit)`; `bias <- estimate_bias(fit, diag, ...)`.
3. Run the pipeline: `qc <- run_qc_pipeline(fit, diag, bias_results = bias)`.
4. Check `qc$overall` for the headline verdict.
5. Review `qc$verdicts` for per-check details.
6. Follow `qc$recommendations` for remediation.
7. Visualize with plot(x).

Saved-result reuse

Supplied screening results must describe this fitted analysis. Results from another fit or older results without identity metadata are refused. See mfrmr_compatibility_layer to recreate them from the saved fit and matching diagnostics without refitting the model.

Interpretation

The rows are the actual checks returned by run_qc_pipeline(): convergence, global fit, selected-facet reliability and separation, element misfit, unexpected responses, category structure, connectivity, inter-rater agreement, and the functioning/bias screen. Element misfit uses the configured mean-square band; it is not a ZSTD test. PCA and displacement are separate diagnostics and are not checks in this pipeline.

Green (Pass), amber (Warn), and red (Fail) describe the selected screening rules. Grey (Skip) marks unrequested or unavailable checks; `AffectsOverall` in the returned data distinguishes their contribution to the overall result. A missing result is not evidence of acceptable fit. High rater separation is not high agreement, and no colour establishes statistical validity or justifies deleting raters or collapsing categories without further review.

See Also

diagnose_mfrm(), estimate_bias(), mfrm_threshold_profiles(), plot(x), plot(x, type = "qc"), build_visual_summaries()

run_qc_pipeline(), mfrmr_plot_migration

Examples

```
# Long-running workflow; run explicitly when needed.
# example("run_qc_pipeline", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("study1")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
  method = "JML", maxit = 300
)
qc <- run_qc_pipeline(fit)
qc
summary(qc)
qc$verdicts

## End(Not run)
# Long-running workflow; run explicitly when needed.
# example("plot.mfrm_qc_pipeline", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("study1")
# These observed data trigger a category-support warning: the QC plot
# displays that review requirement; convergence does not remove it.
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 300)
qc <- run_qc_pipeline(fit)
verdicts <- plot(qc, draw = FALSE)
head(verdicts)

## End(Not run)
```

sample_mfrm_plausible_values *Sample approximate plausible values under fitted posterior scoring*

Description

Sample approximate plausible values under fitted posterior scoring

Usage

```
sample_mfrm_plausible_values(
  fit,
  new_data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
  n_draws = 5,
  interval_level = 0.95,
  scoring_quad_points = 31L,
  readiness_policy = c("error", "review"),
  seed = NULL,
  scoring_prior = NULL
)

## S3 method for class 'mfrm_plausible_values'
summary(object, digits = 3, ...)
```

Arguments

fit	Output from fit_mfrm() estimated with method = "MML" or method = "JML".
new_data	Long-format data for the future or partially observed units to be scored.
person	Optional person column in new_data. Defaults to the person column recorded in fit.
facets	Optional facet-column mapping for new_data. Supply either an unnamed character vector in the calibrated facet order or a named vector whose names are the calibrated facet names and whose values are the column names in new_data.
score	Optional score column in new_data. Defaults to the score column recorded in fit.
weight	Optional weight column in new_data. Defaults to the weight column recorded in fit, if any.

person_data — Optional one-row-per-person data.frame with the background variables required by a latent-regression fit. Ignored for ordinary fitted-object posterior scoring. Intercept-only latent-regression fits can reconstruct the minimal scored-person table internally. This is the scoring-time table for new_data, not the fit object's replay/export provenance table. For categorical background variables, supply values on the same coding scale used at fit time; the fitted factor levels and contrasts are reused when building the scoring design matrix. Fitted standardization and polynomial/spline bases are also reused; they are not re-estimated from the scoring cohort.

person_id — Optional person-ID column in person_data.

population_policy — How missing background data are handled when fit uses the latent-regression branch. "error" (default) requires complete person-level covariates; "omit" drops scored persons lacking complete covariates and records that omission in population_review.

n_draws — Number of posterior draws per person. Must be a positive integer.

interval_level — Posterior interval level passed to predict_mfrm_units() for the accompanying EAP summary table. The same level selects the empirical draw quantiles reported by summary().

scoring_quad_points — Number of Gauss-Hermite nodes used only for this scoring call. Passed to predict_mfrm_units() and independent of the fit-time quadrature order. Fixed or adaptive integration is inherited from fit.

readiness_policy — Source-fit readiness policy passed to predict_mfrm_units(). Conditional calibration and scoring-integration checks are shared with that function and retained in the returned settings.

seed — Optional seed for reproducible posterior draws.

scoring_prior — Optional normal scoring prior, list(mean = ..., sd = ...), passed to predict_mfrm_units(). NULL retains the fitted scoring prior. Estimates and draws retain both the original and the supplied prior; checks of the fitted calibration and numerical accuracy are unchanged.

object — Output from sample_mfrm_plausible_values().

digits — Number of digits used in numeric summaries.

... — Reserved for generic compatibility.

Details

sample_mfrm_plausible_values() uses predict_mfrm_units() to return draws from each Person's posterior distribution as a standalone object. It is useful when downstream workflows want repeated latent-value imputations rather than just one posterior EAP summary.

In the current mfrmr implementation these are **approximate plausible values** drawn from the fitted quadrature-grid posterior under the scoring basis implied by fit, unless scoring_prior explicitly supplies another normal prior. With the default prior, for ordinary MML fits this is the fitted marginal calibration; for latent-regression MML fits it is the fitted conditional normal population model for the

scored persons; for JML fits it is the fixed facet/step calibration together with a standard normal reference prior on the quadrature grid. They should be interpreted as posterior uncertainty summaries for the scored persons, not as deterministic future truth values and not as a claim of full many-facet plausible-values equivalence with population-model software.

In other words, the JML path here is a practical scoring approximation layered on top of the fitted joint-likelihood calibration, whereas the latent-regression MML path uses the fitted one-dimensional conditional normal population model. Neither path should be described as a full many-facet plausible-values system with all ConQuest-style extensions.

summary() method: Summarize approximate plausible values from posterior scoring

Value

An object of class mfrm_plausible_values with components:

- values: one row per person per draw
- fit_identity: source calibration, fitted population coding and scoring status retained from predict_mfrm_units()
- estimates: companion posterior EAP summaries
- row_review: row-preparation review
- population_review: optional person-level omission review for latent-regression scoring
- input_data: cleaned canonical scoring rows retained from new_data
- person_data: cleaned or supplied person-level background data used for latent-regression scoring; NULL otherwise
- settings: scoring settings
- notes: interpretation notes

For summary(): An object of class summary.mfrm_plausible_values with:

- draw_summary: empirical summaries of the sampled values by person
- estimates: companion posterior EAP summaries
- row_review: row-preparation review
- population_review: optional person-level omission review for latent-regression scoring
- settings: scoring settings
- notes: interpretation notes

Interpreting output

- values contains one row per person per draw.
- estimates contains the companion posterior EAP summaries from predict_mfrm_units().
- summary() reports draw counts and empirical draw summaries by person. LowerValue and UpperValue use the requested interval_level and are labelled with IntervalLevel and DrawSummaryBasis. They are empirical quantiles of a finite sample of discrete draws, not the continuous posterior limits in the companion estimates table. With few draws they are coarse; one draw has an unavailable empirical SD. Recreate older summaries from the original plausible-values object to update these limits and labels; no refitting or resampling is needed.

What this does not justify

This helper does not update the calibration, estimate new non-person facet levels, or provide exact future true values. It samples from the quadrature- grid posterior implied by the existing fitted-model scoring basis, using fixed or Person-specific adaptive nodes according to the fit's setting. Calibration and population parameters remain fixed. These draws alone do not validate downstream group comparisons or regressions; those analyses require a compatible conditioning model and sampling design.

References

The underlying posterior scoring follows the usual quadrature-based EAP framework of Bock and Aitkin (1981). The interpretation of multiple posterior draws as plausible-value-style summaries follows the general logic discussed by Mislevy (1991), while the current implementation remains a practical fitted-object posterior approximation rather than a full published many-facet plausible-values method. For `JML` source fits, the quadrature posterior uses a package-level standard normal reference prior for this post hoc scoring layer.

- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Mislevy, R. J. (1991). *Randomization-based inference about latent variables from complex samples*. Psychometrika, 56(2), 177-196.

Saved-result reuse

Fit-level export and replay helpers check the source calibration and scoring status of supplied draws. Recreate older results from the saved fit and original scoring inputs/settings, including the seed. This repeats drawing values without refitting the calibration. See mfrmr_compatibility_layer.

See Also

`predict_mfrm_units()`, summary.mfrm_plausible_values

`sample_mfrm_plausible_values()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("sample_mfrm_plausible_values", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- fit_mfrm(
  toy[toy$Person %in% keep_people, , drop = FALSE],
  "Person", c("Rater", "Criterion"), "Score",
  method = "MML",
  quad_points = 5,
  maxit = 30
)
new_units <- data.frame(
  Person = c("NEW01", "NEW01"),
  Rater = unique(toy$Rater)[1],
```

```
  Criterion = unique(toy$Criterion)[1:2],
  Score = c(2, 3)
)
pv <- sample_mfrm_plausible_values(toy_fit, new_units, n_draws = 3, seed = 1)
summary(pv)$draw_summary

## End(Not run)
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_plausible_values", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- fit_mfrm(
  toy[toy$Person %in% keep_people, , drop = FALSE],
  "Person", c("Rater", "Criterion"), "Score",
  method = "MML",
  quad_points = 5,
  maxit = 30
)
new_units <- data.frame(
  Person = c("NEW01", "NEW01"),
  Rater = unique(toy$Rater)[1],
  Criterion = unique(toy$Criterion)[1:2],
  Score = c(2, 3)
)
pv <- sample_mfrm_plausible_values(toy_fit, new_units, n_draws = 3, seed = 1)
summary(pv)

## End(Not run)
```

`score_comparison_table` *Compare aligned scores against reference ratings*

Description

Summarize agreement, association and mean differences for two score sources, with an optional second-reference baseline. Sources can be human raters, scoring systems or other scoring procedures using the same rubric. No measurement-model fit is needed for the score comparisons.

Usage

```
score_comparison_table(
  scores,
  reference,
  reference2 = NULL,
  categories,
  score_categories = scores,
  smd = c("pooled", "reference"),
```

```
  screen = c("none", "williamson2012"),
  qwk_min = NULL,
  correlation_min = NULL,
  max_degradation = NULL,
  max_abs_smd = NULL,
  fit = NULL,
  diagnostics = NULL,
  rater_facet = NULL,
  rater = NULL
)
```

Arguments

`scores`	Numeric scores from the source being evaluated, before rounding.
`reference`	Numeric reference ratings on the declared categories. A reference is a comparison source, not necessarily an error-free criterion.
`reference2`	Optional second reference rating of each response. Its agreement with `reference` supplies the baseline for degradation.
`categories`	All permitted numeric score categories, in increasing order, including categories not observed in this sample.
`score_categories`	Scores expressed on `categories`. Defaults to `scores`. For continuous scores, supply an explicitly categorized vector. No rounding, clipping, category inference or averaging is performed.
`smd`	SMD denominator: `"pooled"` uses the square root of the mean of the two sample variances; `"reference"` uses the reference sample SD. Both SMDs are returned. SDs use the sample divisor `N - 1`.
`screen`	`"none"` (default) applies no automatic thresholds. `"williamson2012"` selects the automated-scoring examples discussed by Williamson, Xi, and Breyer (2012): QWK and correlation >= .70, agreement degradation <= .10, and absolute SMD <= .15.
`qwk_min, correlation_min`	Optional minimum QWK and Pearson correlation.
`max_degradation`	Optional maximum baseline-minus-score agreement, applied separately to QWK and correlation. Negative degradation indicates higher agreement of `scores` with `reference` than the baseline.
`max_abs_smd`	Optional maximum absolute selected SMD. Non-NULL threshold arguments override the selected screen. With `screen = "none"`, only explicitly supplied thresholds are applied.
`fit`	Optional native additive RSM/PCM `fit_mfrm()` result containing the selected scorer as a rater. This function does not fit a model.
`diagnostics`	Optional matching `diagnose_mfrm()` result, supplying saved Infit/Outfit. No diagnostics are computed here.
`rater_facet, rater`	With `fit`, the explicit facet name and level identifying the scorer being evaluated.

Details

Vectors must already be aligned: one entry per same response in the same order, on one common rubric/score scale. Match Person, task, criterion and occasion before calling. Repeated ratings are not collapsed. All statistics use the same complete cases across supplied vectors, including `score_categories` and `reference2`. Every row's inclusion and missing-input flags are retained in `cases`; missingness may affect representativeness.

QWK is $1 - D_o/D_e$, where D_o is observed mean squared category disagreement and D_e uses independent draws from empirical marginals. Distances use numeric category values, not their ranks. The denominator is not MFRM model-implied expected agreement. Constant identical ratings have undefined QWK. Correlation uses unrounded scores. SMD is signed `scores` minus `reference`; numerical screens use its absolute value.

`ReferenceQWK` and `ReferenceCorrelation` compare `reference2` with `reference`; degradation subtracts the score-reference statistic from that baseline. These columns are unavailable without a second reference. `criteria` retains all five statistics, even when no thresholds are requested. Its status is `"not_requested"` for a finite statistic without a threshold, `"unavailable"` for an undefined statistic, or `"within_threshold"` / `"review"` when an inclusive threshold is applied. No combined approval, significance test or sampling uncertainty is inferred.

The optional Williamson screen concerns joint human/automated scoring; it is not a universal standard for all score comparisons. It evaluates unrounded statistics and does not reproduce the paper's unspecified rounding convention or operational exceptions. Its overall SMD example is .15; the subgroup discussion uses .10. The source's p. 8 prose uses human-SD SMD, whereas p. 12 note 4 uses the pooled formula. The default follows the formula and retains `SMDReference` for sensitivity review. Evaluation data/tasks should be separate from model development as required by the intended use. Construct relevance, reference quality, external validity, generalizability, subgroup differences and consequences require additional evidence.

Optional MFRM columns describe **all retained source ratings for the selected rater**, which need not be the comparison cohort. Source counts, readiness, model settings and scope are retained separately. Severity is in logits, oriented so positive values imply lower expected ratings; its zero is the fitted model reference. Severity and Infit/Outfit do not measure criterion accuracy and receive no automatic quality threshold.

Value

A list with `table` (one comparison row), `criteria` (five statistics, optional thresholds, availability and reasons), `cases` (row disposition), `counts` (`scores_reference` and `reference2_reference` category tables), `settings`, `mfrm` (optional saved source metadata), and `notes`. Save with `base::saveRDS()` and reopen with `base::readRDS()`. Export the data frames with `utils::write.csv()`. No ICC, alpha or AC2 is computed.

References

Williamson, D. M., Xi, X., & Breyer, F. J. (2012). A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice, 31*(1), 2-13. doi:10.1111/j.1745-3992.2011.00223.x Especially pp. 7-10 and p. 12, notes 1 and 4.

See Also

interrater_agreement_table(), mfrm_report(), automated_scoring_table() for the earlier automated-scoring interface.

Examples

```
comparison <- score_comparison_table(
  scores = c(0, 1, 2, 2, 3, 3), reference = c(0, 1, 1, 2, 3, 3),
  categories = 0:3
)
comparison$table
# The automated-scoring screen is an explicit application choice.
auto_comparison <- score_comparison_table(
  scores = c(0.2, 0.9, 1.3, 2.1, 2.7, 2.8),
  reference = c(0, 1, 1, 2, 3, 3), reference2 = c(0, 1, 2, 2, 2, 3),
  categories = 0:3, score_categories = c(0, 1, 1, 2, 3, 3),
  screen = "williamson2012"
)
auto_comparison$criteria
```

---

score_mfrm_persons *Estimate abilities for people already included in a fitted RSM*

Description

Use the existing ratings and fitted model to estimate each person's ability. This is the common scoring entry for supported ordinary, shared-rater and testlet rating-scale models (RSMs). It reuses the fitted calibration, such as rater severity and category thresholds; it does not fit the model again.

Usage

```
score_mfrm_persons(
  fit,
  persons = NULL,
  level = 0.95,
  quad_points = NULL,
  cut_scores = NULL
)

## S3 method for class 'mfrm_person_scores'
summary(object, ...)

## S3 method for class 'mfrm_person_scores'
print(x, ...)

## S3 method for class 'mfrm_person_scores'
plot(x, ...)
```

Arguments

fit	A numerically ready ordinary RSM MML, testlet or shared-rater fit.
persons	Distinct source Person IDs; NULL returns all source Persons, including Persons whose assigned scores are all missing. For a shared-rater fit, start with a few actual IDs: scoring everyone can be slow. Selecting IDs changes output rows, not the data used to account for shared raters.
level	Conditional equal-tail interval probability, between zero and one.
quad_points	Extension quadrature order; NULL uses its fit's order. Ordinary ability integration is continuous and does not use quadrature; leave this argument NULL for ordinary fits.
cut_scores	Optional finite, strictly increasing numeric ability cuts, with optional distinct names. Ordinary RSM scoring only. Returns a decisions table of probabilities above each cut and in successive bands (Lower < ability <= Upper). Calibration, population parameters and cuts are held fixed. Labels do not establish CEFR alignment or a pass/fail rule.
object, x	An ordinary mfrm_person_scores result.
...	Display controls passed to plot.mfrm_testlet_scores(); unused by print and summary.

Details

The estimate is the mean of the conditional ability distribution (EAP). Posterior SDs and continuous equal-tail intervals describe its uncertainty with calibration held fixed. Selecting persons changes the returned rows, not the ratings used to condition shared effects.

Ordinary fits use the bounded model/population specification of mfrm_response_diagnostics(): additive unanchored severity facets and known N(0,1) or an estimated intercept-only normal population. Both the fitted population mean and variance are retained. EAP, posterior SD and interval endpoints integrate the continuous conditional density; endpoints are not quadrature-grid quantiles. Ordinary plug-in scoring is unchanged.

For extensions this calls predict.mfrm_testlet() or score_mfrm_random_rater() on the complete source roster. Their numerical checks, approximation limits and uncertainty definitions apply unchanged. Use those functions directly for a different complete scoring roster.

Intervals condition on fitted calibration and the assumed normal population. They exclude calibration-estimation uncertainty, do not test Person differences and do not guarantee frequentist coverage for each fixed ability. Missing-only Persons remain prior_only; zero ability variance and failed integration return unavailable, not zero-width intervals. No missing scores are imputed.

Value

Ordinary fits return mfrm_person_scores with table, scoring_data, data usage, settings and source metadata. Extensions retain their existing scoring classes. If cut_scores is supplied, decisions retains probabilities, boundaries, labels, priors and availability; summary(x)$decisions and apa_table(x, which = "decisions") reuse it. Supply saved scores to compare_mfrm() as person_scores; extension scores also support model-aware maps through mfrm_results(). Saved plots/reports do not recompute scores. In table, Status and Reason distinguish available_conditional (a computed conditional posterior), prior_only (no observed responses; the estimate and interval

describe the assumed population alone), and `unavailable` (the estimate, SD and bounds are withheld, with the cause). Keep these columns with exported scores. More integration points cannot supply missing responses or resolve a zero ability-variance restriction.

Conditional ability decisions

Supply `cut_scores` on the calibration's ability scale to obtain `decisions`, with one row per Person and requested event. `Type = "above_cut"` gives $P(\theta > c \mid y, \widehat{\psi})$; `Type = "ability_band"` gives the probability between successive cuts, including the two outer bands. `Lower < ability <= Upper` defines every row. `Label` uses the supplied cut names (or `Cut1`, ...) and `Band1`, ... in increasing order. These events overlap: only the ability-band probabilities sum to one.

`PriorMean` and `PriorSD` record the actual Person-specific normal prior; `Status` and `Reason` retain scoring availability. `available_conditional` means the applicable numerical checks passed, not that the model or the cut's substantive meaning has been validated. `review_only` requires inspection before interpretation. Source-roster scoring may return `prior_only` for missing-only Persons; portable scoring instead retains them as `unavailable` with `NA` probabilities. Fitted-object scoring omits Persons with no retained rows; inspect its row and population reviews. A Person absent from the input is never invented.

Probabilities integrate the continuous posterior, rather than counting `sample_mfrm_plausible_values()` draws or applying a normal approximation to EAP/SD. They condition on point calibration, population/prior parameters and supplied cuts, as recorded in `Uncertainty`. They do not propagate estimation error, compare two Persons or predict passing a future test. CEFR labels require externally justified cuts on this same scale. Very small tails have absolute, not relative, numerical accuracy; a returned zero or one is not a claim of impossibility or certainty.

Review the scoring population as well as the calibration. An estimated normal prior is treated as known during scoring; its sampling uncertainty can matter especially with a small calibration cohort. A larger calibration cohort can improve parameter precision without correcting a prior that mismatches a new population's mean, spread or shape. Passing numerical checks does not establish that probabilities are accurate for that new population. A fixed reference prior is not a universal remedy: justify it for the target population on the calibration scale. If a prior override is supported, comparing externally justified alternatives is a sensitivity analysis, not evidence that the alternative giving the preferred classification is valid. Population-averaged accuracy also does not guarantee accuracy near each cut, within every subgroup or for each individual Person.

Use `score_mfrm_persons()` for eligible ordinary RSM MML source Persons, `predict_mfrm_units()` for ordinary RSM/PCM MML or JML fitted-object scoring, and `score_mfrm_calibration()` for eligible continuous-posterior RSM/PCM artifacts. Each route retains its existing source and population restrictions. Unit weights are required; GPCM, shared-rater and testlet cut probabilities are not supported. JML uses a separate scoring prior, not a population estimated by JML. `NULL` leaves the existing output unchanged.

`summary(x)$decisions` and `apa_table()` with `which = "decisions"` reuse saved probabilities without rounding them. Fitted-object and portable score batches also support `build_summary_table_bundle()` and `export_summary_appendix()`. Save the complete result with `saveRDS()` to retain cuts, priors and review states for later reporting. Existing score plots display EAPs and intervals, not these event probabilities.

See Also

`score_mfrm_calibration()` for new people under an eligible saved ordinary RSM/PCM calibra-

tion; predict_mfrm_units() for supplied response rows under a fitted calibration; plot.mfrm_testlet_scores(), mfrm_response_diagnostics()

Examples

```
ratings <- load_mfrmr_data("example_core")
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score")
scores <- score_mfrm_persons(fit, persons = unique(ratings$Person)[1:4],
  cut_scores = c(Basic = -0.5, Advanced = 0.8))
scores$table
scores$decisions
apa_table(scores, which = "decisions")
plot(scores, draw = FALSE)
```

---

score_mfrm_random_rater

Score Persons while integrating shared random raters

Description

Compute conditional Person scores using the complete joint scoring roster, with calibration held fixed and shared raters integrated jointly.

Usage

```
score_mfrm_random_rater(
  object,
  newdata = NULL,
  persons = NULL,
  level = 0.95,
  quad_points = object$settings$quad_points,
  missing = if (is.null(newdata)) object$settings$missing else "fail"
)

## S3 method for class 'mfrm_random_rater_scores'
summary(object, ...)

## S3 method for class 'mfrm_random_rater_scores'
print(x, ...)

## S3 method for class 'mfrm_random_rater_scores'
plot(
  x,
  draw = TRUE,
  style = c("interval", "precision", "distribution"),
  sort = c("input", "estimate", "uncertainty"),
  palette = c("accessible", "mono"),
  title = NULL,
```

```
  caption = NULL,
  show_title = FALSE,
  show_notes = FALSE,
  show_labels = TRUE,
  reference = 0,
  text_scale = 1,
  point_size = 2.5,
  ...
)
```

Arguments

object	A numerically ready fit_mfrm_random_rater() result.
newdata	Complete long-format scoring roster, with the fitted columns. NULL uses the source roster. A supplied table replaces the source roster; it is not appended to it. New Person and rater IDs are allowed, but fixed facet levels must be known. Include all responses intended to inform the shared raters, including responses of Persons not selected for display.
persons	Character vector of distinct Person IDs to return; NULL returns all Persons in the scoring roster. This selects outputs, not data.
level	Conditional equal-tail interval probability; default 0.95.
quad_points	Other-Person quadrature order, 7 to 241; default uses the fit's order. Scores are checked at 2 * quad_points + 1 points.
missing	"fail" or "omit". Source replay uses the fit's policy; omission in a supplied roster must be requested explicitly.
x	For score_mfrm_random_rater(): A scoring result. For plot(): A result from score_mfrm_random_rater().
...	Unused.
draw	FALSE returns plot data without opening a device.
style	"interval" keeps every labeled row; "precision" plots estimates against interval width (upper minus lower bound); "distribution" shows the empirical cumulative distribution of finite point estimates, excluding prior-only and unavailable scores. No density is reconstructed.
sort	Row order for interval plots: input order, increasing estimate, or increasing interval width. Missing values come last; ties retain input order. Sorting is descriptive, not a test of differences.
palette	"accessible" uses blue and dark orange with filled/open symbols; "mono" uses dark grey with the same symbol distinctions.
title, caption	NULL uses the default text; a string replaces it; "" removes it. Interpretation notes remain in the saved plot data.
show_title, show_notes	Show the title and caption, respectively. FALSE hides text without deleting its metadata.

show_labels	Show IDs on interval and precision plots. Distribution plots summarize estimates and do not label individual IDs. For crowded precision plots use FALSE and consult the retained table.
reference	Vertical reference line in logits; NULL omits it. This is an orientation aid, not a quality threshold.
text_scale	Positive multiplier for text size.
point_size	Positive point size in ggplot millimetres; base graphics use the corresponding relative size (2.5 is the default).

Details

For each requested Person, the other Persons' abilities are integrated by quadrature. At each value of the requested ability, the full shared-rater vector is integrated by a conditional Laplace approximation. Multiplying this likelihood by the normal ability density and normalizing gives an approximate continuous marginal posterior. EAP, posterior SD and equal-tail endpoints are calculated from that density. Neither independent rater marginals nor rater modes substituted as known values define this calculation.

Writing $L_j(u)$ for Person j's likelihood integrated over that Person's normal ability population, the target for Person p is

$$q_p(t) \propto \phi_p(t) \int \phi_r(u) P(y_p \mid t, u) \prod_{j \neq p} L_j(u)\, du.$$

Calibration arguments are suppressed in this expression. The rater integral is approximated separately at each t, and the resulting density is normalized over t; a rater mode is not held fixed across t.

InterpolationDifference checks refinement of the continuous density, and IntegrationDifference compares other-Person quadrature orders. Small values do not bound the separate rater Laplace approximation error or establish interval coverage. Endpoints invert a continuous CDF, not the steps of a quadrature-grid CDF. Computation grows with roster size and requested Persons. Start with a few IDs when checking a large roster; persons = NULL computes every Person's posterior and can be slow. Selecting outputs keeps all scoring responses. No fixed effects or population variances are re-estimated.

With newdata, reusing a source rater ID does not import that rater's saved conditional distribution. Shared-rater distributions are calculated from the supplied responses and fitted population parameters. To retain earlier responses, include them in the complete scoring table, with each intended rating event appearing once. By contrast, predict.mfrm_random_rater() with rater = "observed" uses the rater's conditional distribution from the calibration responses at abilities you supply; it does not estimate abilities from new responses.

Intervals condition on all supplied observed responses, fitted calibration and the fitted or specified normal ability population. They exclude calibration/population estimation uncertainty, are not fixed-ability frequentist confidence intervals and do not test Person differences. They do not qualify few-rater uncertainty or the normality and assignment assumptions. With an estimated zero rater variance they describe that fitted submodel, not certainty that rater differences are absent.

An entirely missing Person explicitly retained with missing = "omit" returns the normal prior with status "prior_only", not a measured average ability. Zero fitted ability variance gives "unavailable"

scores rather than zero-width intervals. Errors or unresolved numerical checks preserve the requested Person's row and reason. No missing scores are imputed. Older saved fits keep their original known N(0,1) ability population. If an older fit omitted scores without saving the original roster, supply that roster explicitly to retain its unobserved Persons and row accounting.

`plot()` method: Plot Person scores from a shared-rater model

For `plot()`: In the default interval view, all requested Persons remain labeled. Open circles indicate prior-only results; unavailable scores have no point or interval. Intervals are conditional on calibration and use joint rater Laplace integration. They are not tests of Person differences. `as_ggplot()` preserves these meanings without rerunning scoring.

Value

An `mfrm_random_rater_scores` object with `table`, complete scoring roster, data accounting, settings and source metadata for `mfrm_results()`. Save it with `saveRDS()`. Its summary and plots need no live optimizer. Keep `table$Status` and `table$Reason` with exported scores: `prior_only` describes the assumed population without observed responses for that Person; `unavailable` withholds the estimate, SD and bounds with a recorded cause.

For `plot()`: Invisibly, an `mfrm_plot_data` object; use `plot_data()`.

Numerical evidence

An independent joint-posterior comparison at fixed calibration covered three-category RSMs with six or 24 raters and rotating or weakly linked assignments. Eight scoring rosters contained 240 Persons and 1,440 responses; four reduced rosters contained 48 Persons and 96 responses. The 48 selected EAPs and posterior SDs met a 0.05-logit numerical tolerance, including Monte Carlo uncertainty. Conditional-CDF calculations using the saved joint rater draws supported the stated 0.10-logit tolerance for all 96 interval endpoints, allowing for Monte Carlo and numerical error.

These are bounded numerical comparisons, not universal accuracy bounds or performance cutoffs. They do not establish repeated-sampling 95 percent coverage when calibration is estimated, qualify the calibration likelihood, or test Person differences. See `vignette("mfrmr-random-raters")` for the reference-precision limitations and interpretation.

See Also

`predict.mfrm_random_rater()`, `plot.mfrm_random_rater_scores()`

`plot.mfrm_testlet_scores()` for view selection, accessible output and the meaning of the empirical distribution.

Examples

```
example <- readRDS(system.file("examples", "extended-models.rds", package = "mfrmr"))
fit <- example$random_rater$fit
scores <- example$random_rater$scores
# To recompute (requires RTMB >= 2.0 and can take several minutes):
# scores <- score_mfrm_random_rater(fit, persons = c("P001", "P002"))
# The complete source roster still informs both selected Persons.
scores$table
plot(scores)
mfrm_results(fit, scores = scores, compute = "never")
```

shrinkage_report *Extract the shrinkage report from a fitted mfrm_fit*

Description

Lightweight accessor that returns the per-facet empirical-Bayes shrinkage table stored on a fit when `facet_shrinkage != "none"`. Returns `NULL` (with a message) when no shrinkage has been applied so callers can probe without error.

Usage

```
shrinkage_report(fit)
```

Arguments

`fit`	An `mfrm_fit` object.

Value

A data.frame with one row per facet (and optionally `"Person"`) or `NULL` when shrinkage has not been applied.

See Also

`apply_empirical_bayes_shrinkage()`, `fit_mfrm()`.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("shrinkage_report", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 300,
                facet_shrinkage = "empirical_bayes")
shrinkage_report(fit)

## End(Not run)
```

simulate_mfrm_data *Simulate long-format ordered many-facet data for design studies*

Description

Simulate long-format ordered many-facet data for design studies

Usage

```
simulate_mfrm_data(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL,
  seed = NULL,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = "Criterion",
  slope_facet = NULL,
  thresholds = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  sim_spec = NULL
)
```

Arguments

n_person	Number of persons/respondents.
n_rater	Number of rater facet levels.
n_criterion	Number of criterion/item facet levels.
raters_per_person	Number of raters assigned to each person.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. When sim_spec = NULL, names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person) or role keywords (person, rater, criterion, assignment). A nested named-facet form,

design$facets = c(person = ..., rater = ..., criterion = ...), is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both design and the scalar count arguments.

score_levels — Number of ordered score categories.

theta_sd — Standard deviation of simulated person measures.

rater_sd — Standard deviation of simulated rater severities.

criterion_sd — Standard deviation of simulated criterion difficulties.

noise_sd — Optional observation-level noise added to the linear predictor.

step_span — Spread of step thresholds on the logit scale.

group_levels — Optional character vector of group labels. When supplied, a balanced Group column is added to the simulated data.

dif_effects — Optional data.frame describing true group-linked DIF effects. Must include Group, at least one design column such as Criterion, and numeric Effect.

interaction_effects — Optional data.frame describing true non-group interaction effects. Must include at least one design column such as Rater or Criterion, plus numeric Effect.

seed — Optional random seed.

model — Measurement model recorded in the simulation setup. The current public generator supports RSM, PCM, and GPCM.

step_facet — Step facet used when model = "PCM" and threshold values vary across levels. Currently "Criterion" and "Rater" are supported.

slope_facet — Slope facet used when model = "GPCM". The current GPCM branch requires slope_facet == step_facet.

thresholds — Optional threshold specification. Use a numeric vector of common thresholds; a named list such as list(C01 = c(-1, 0, 1)); a numeric matrix with one row per StepFacet and one column per step; or a long data frame with columns StepFacet, Step/StepIndex, and Estimate.

slopes — Optional slope specification used when model = "GPCM". Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns SlopeFacet and Estimate. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean-one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact PCM reduction.

assignment — Assignment design. "crossed" means every person sees every rater; "rotating" uses a balanced rotating subset; "resampled" reuses person-level rater-assignment profiles stored in sim_spec; "sparse_linked" uses an incomplete rating design with optional linking persons; "skeleton" reuses an observed response skeleton stored in sim_spec, including optional Group/Weight columns when available. When omitted, the function chooses "crossed" if raters_per_person == n_rater, otherwise "rotating".

sparse_controls — Optional named list used when assignment = "sparse_linked". Supported entries are link_fraction, link_persons, link_raters_per_person, assignment_mode, and min_common_persons_per_rater_pair. See build_mfrm_sim_spec() for the same contract.

`sim_spec` Optional output from `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()`. When supplied, it defines the generator setup; direct scalar arguments are treated as legacy inputs and should generally be left at their defaults except for `seed`. Any custom public two-facet names recorded in `sim_spec$facet_names` are also carried into the simulated output and downstream planning helpers. If `sim_spec` stores an active latent-regression population generator, the returned object also carries the generated one-row-per-person background-data table needed to refit that population model later.

Details

This function generates synthetic ordered many-facet data under `RSM`, `PCM`, or the package's `GPCM` branch. The data-generating process is:

1. Draw person abilities: $\theta_n \sim N(0, \texttt{theta_sd}^2)$
2. Draw rater severities: $\delta_j \sim N(0, \texttt{rater_sd}^2)$
3. Draw criterion difficulties: $\beta_i \sim N(0, \texttt{criterion_sd}^2)$
4. Unless thresholds are supplied, generate evenly spaced thresholds from `-step_span` to `step_span`; the single binary threshold is zero
5. For each observation, compute the linear predictor $\eta = \theta_n - \delta_j - \beta_i + \epsilon$ where $\epsilon \sim N(0, \texttt{noise_sd}^2)$ (optional)
6. Compute category probabilities under the recorded measurement model (`RSM`, `PCM`, or `GPCM`) and sample the response

Latent-value generation is explicit:

- `latent_distribution = "normal"` draws centered normal person/rater/ criterion values using the supplied standard deviations. Values are assigned independently to level IDs, without sorting them by severity or difficulty; zero is the distribution mean, not a forced sample mean.
- `latent_distribution = "empirical"` resamples centered support values recorded in `sim_spec$empirical_support`
- if `sim_spec$population$active = TRUE`, person measures are generated from the stored latent-regression population model and template person covariates rather than from `theta_sd`

Latent-regression transformations and factor coding retain their template basis during generation and refitting. `mfrm_population_data$population_formula` is a formula-compatible `terms` object carrying this basis, factor levels and contrasts. Pass it together with the returned `person_data` when fitting generated data manually. Old specifications containing transformed predictors but no stored basis must be rebuilt from the original template or source fit. Previously saved generated data/results are not silently reinterpreted.

Earlier versions sorted normal rater/criterion draws before assigning IDs, imposing an association with ID order and any ID-specific steps/slopes. Correcting that ordering changes seeded datasets. Preserve old generated data and results for historical reproduction; new runs use the corrected generator and a matching saved specification/software environment.

When `dif_effects` is supplied, the specified linear-predictor shift is added to η for the focal group on the target facet level, creating a known DIF signal. Similarly, `interaction_effects` injects a known bias into specific facet-level combinations. For RSM/PCM this is also the adjacent-category log-odds shift. For GPCM, the shift in adjacent-category log odds is the specified effect multiplied

by the observation's discrimination. A common injected effect therefore need not imply equal response-probability changes across facet levels.

The generator targets the common two-facet rating design (persons × raters × criteria). `raters_per_person` controls the incomplete-block structure: when less than `n_rater`, each person is assigned a rotating subset of raters to keep coverage balanced and reproducible.

Threshold handling is intentionally explicit:

- if `thresholds = NULL`, common equally spaced thresholds are generated from `step_span`
- if `thresholds` is a numeric vector, it is used as one common threshold set
- if `thresholds` is a named list, numeric matrix, or data frame, threshold values may vary by `StepFacet` (currently `Criterion` or `Rater`)

For `GPCM`, the generator requires an explicit slope contract in parallel with the threshold table. The supported route keeps `slope_facet == step_facet`, normalizes supplied slopes to the same geometric-mean-one log-slope identification used by `fit_mfrm()`, and samples responses from the corresponding GPCM category probabilities. Independent arbitrary slope-facet planning is not supported by these design, population-forecasting, diagnostic-screening, or signal-detection helpers.

Assignment handling is also explicit:

- `"crossed"` uses the full person x rater x criterion design
- `"rotating"` assigns a deterministic rotating subset of raters per person
- `"sparse_linked"` assigns most persons to an incomplete rater subset and assigns a configurable set of linking persons to a larger rater set
- `"resampled"` reuses empirical person-level rater profiles stored in `sim_spec$assignment_profiles`, optionally carrying over person-level `Group`
- `"skeleton"` reuses an observed person-by-rater-by-criterion response skeleton stored in `sim_spec$design_skeleton`, optionally carrying over `Group` and `Weight`

With an active population model, `population_assignment = "matched"` in `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()` draws covariates and skeleton/profile assignments from the same template Person. Independent draws require `population_assignment = "independent"`. A population `Group` column supplies the response `Group`; it is not drawn a second time. Conflicting group labels or unmatched template IDs fail with an explanation. Stored specifications without `population_assignment` retain independent sampling; rebuild them to request matched sampling. The ambiguous case of independently drawn tables both defining `Group` now errors.

Skeleton rows retain observation multiplicity. Repeated cells receive new, conditionally independent responses given the generated latent values and signals. Study runners declare this assumption via `repeated_ratings = "independent"` when refitting these generated data. This does not establish independence of repeated ratings in the source data, or simulate serial, occasion or peer-network dependence. `Weight` is passed through to fitting, not interpreted as additional independently generated rows.

Sparse linked simulation is intended for planned-missing rating designs in which connectivity is maintained through common linking persons. The returned `mfrm_sparse_design` attribute summarizes design density, planned missingness, rater coverage, and rater-pair common-person counts. These summaries are design diagnostics, not model-fit statistics or universal adequacy thresholds. This branch follows sparse rater-mediated assessment design work by Wind, Jones, and Grajeda

(2023, doi:10.1177/01466216231182148), Wind and Jones (2018, doi:10.1177/0013164417703733), and DeMars, Shapovalov, and Hathcoat (2023).

For more controlled workflows, build a reusable simulation specification first via `build_mfrm_sim_spec()` or derive one from an observed fit with `extract_mfrm_sim_spec()`, then pass it through `sim_spec`.

Returned data include attributes:

- `mfrm_truth`: simulated true parameters (for parameter-recovery checks)
- `mfrm_truth$signals`: injected DIF and interaction signal tables
- `mfrm_truth$population$template_person` and `mfrm_truth$design$template_person`: source template IDs keyed by generated Person, where population or assignment templates are used
- `mfrm_truth$slope_table`: simulated discrimination table for `GPCM`
- `mfrm_population_data`: generated one-row-per-person background data when the simulation specification stores an active latent-regression generator, including model-matrix xlevel and contrast provenance for categorical covariates
- `mfrm_simulation_spec`: generation settings (for reproducibility) including the effective `population_assignment` (`"matched"`, `"independent"` or `"none"`) and the conditional response-independence rule
- `mfrm_sparse_design`: sparse-design diagnostics when `assignment = "sparse_linked"`, including design density, planned missing rate, rater coverage, and rater-pair common-person counts

Value

A long-format `data.frame` with core columns `Study`, `Person`, two simulated non-person facet columns, and `Score`. By default those facet columns are `Rater` and `Criterion`; when `sim_spec` records custom public names, those names are used instead. If group labels are simulated or reused from an observed response skeleton, a `Group` column is included. If a weighted response skeleton is reused, a `Weight` column is also included.

Interpreting output

- Higher `theta` values in `mfrm_truth$person` indicate higher person measures.
- Higher values in `mfrm_truth$facets$Rater` indicate more severe raters.
- Higher values in `mfrm_truth$facets$Criterion` indicate more difficult criteria.
- `mfrm_truth$signals$dif_effects` and `mfrm_truth$signals$interaction_effects` record any injected detection targets.

Typical workflow

1. Generate one design with `simulate_mfrm_data()`.
2. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`.
3. For repeated design studies, use `evaluate_mfrm_design()`.

See Also

`evaluate_mfrm_design()`, `fit_mfrm()`, `diagnose_mfrm()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("simulate_mfrm_data", run.dontrun = TRUE)
## Not run:
sim <- simulate_mfrm_data(
  n_person = 40,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = 2,
  seed = 123
)
head(sim)
names(attr(sim, "mfrm_truth"))

## End(Not run)
```

`simulate_mfrm_recovery` *Simulate parameter recovery by repeated generation and refitting*

Description

Runs a compact parameter-recovery simulation study: generate data from a known ordered many-facet data-generating setup, refit the requested model, align estimates to the known truth where location indeterminacy requires it, and summarize bias, RMSE, MAE, correlation, and coverage of available centered structural facet intervals. This function runs the simulation and fits models. To review an already saved simulation result without new data generation or fitting, use `assess_mfrm_recovery()`. In 0.2.4 this replaces `evaluate_mfrm_recovery()`; the old name remains an alias. See the details for corrected recovery targets, Monte Carlo uncertainty and migration of saved results.

Usage

```
simulate_mfrm_recovery(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
```

```
  step_facet = NULL,
  slope_facet = NULL,
  thresholds = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  sim_spec = NULL,
  fit_method = c("JML", "MML"),
  maxit = 25,
  quad_points = 7,
  include_person = TRUE,
  include_diagnostics = FALSE,
  diagnostic_fit_df_method = c("both", "engine", "facets"),
  seed = NULL,
  file = NULL,
  file_refit = c("never", "on_change", "always")
)
```

Arguments

n_person	Number of persons/respondents.
n_rater	Number of rater facet levels.
n_criterion	Number of criterion/item facet levels.
raters_per_person	Number of raters assigned to each person.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. When sim_spec = NULL, names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person) or role keywords (person, rater, criterion, assignment). A nested named-facet form, design$facets = c(person = ..., rater = ..., criterion = ...), is also accepted for the supported person/rater/criterion design. Do not specify the same variable through both design and the scalar count arguments.
reps	Number of Monte Carlo replications.
score_levels	Number of ordered score categories.
theta_sd	Standard deviation of simulated person measures.
rater_sd	Standard deviation of simulated rater severities.
criterion_sd	Standard deviation of simulated criterion difficulties.
noise_sd	Optional observation-level noise added to the linear predictor.
step_span	Spread of step thresholds on the logit scale.
model	Measurement model recorded in the simulation setup. The current public generator supports RSM, PCM, and GPCM.
step_facet	Step facet used when model = "PCM" and threshold values vary across levels. Currently "Criterion" and "Rater" are supported.
slope_facet	Slope facet used when model = "GPCM". The current GPCM branch requires slope_facet == step_facet.

`thresholds` Optional threshold specification. Use a numeric vector of common thresholds; a named list such as `list(C01 = c(-1, 0, 1))`; a numeric matrix with one row per `StepFacet` and one column per step; or a long data frame with columns `StepFacet`, `Step`/`StepIndex`, and `Estimate`.

`slopes` Optional slope specification used when `model = "GPCM"`. Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns `SlopeFacet` and `Estimate`. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean-one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact `PCM` reduction.

`assignment` Assignment design. `"crossed"` means every person sees every rater; `"rotating"` uses a balanced rotating subset; `"resampled"` reuses person-level rater-assignment profiles stored in `sim_spec`; `"sparse_linked"` uses an incomplete rating design with optional linking persons; `"skeleton"` reuses an observed response skeleton stored in `sim_spec`, including optional `Group`/`Weight` columns when available. When omitted, the function chooses `"crossed"` if `raters_per_person == n_rater`, otherwise `"rotating"`.

`sparse_controls` Optional named list used when `assignment = "sparse_linked"`. Supported entries are `link_fraction`, `link_persons`, `link_raters_per_person`, `assignment_mode`, and `min_common_persons_per_rater_pair`. See `build_mfrm_sim_spec()` for the same contract.

`sim_spec` Optional output from `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()`. When supplied, it defines the generator setup; direct scalar arguments are treated as legacy inputs and should generally be left at their defaults except for `seed`. Any custom public two-facet names recorded in `sim_spec$facet_names` are also carried into the simulated output and downstream planning helpers. If `sim_spec` stores an active latent-regression population generator, the returned object also carries the generated one-row-per-person background-data table needed to refit that population model later.

`fit_method` `"JML"` or `"MML"`, passed to `fit_mfrm()`.

`maxit` Maximum optimizer iterations passed to `fit_mfrm()`.

`quad_points` Quadrature points used when `fit_method = "MML"`.

`include_person` Logical. When `TRUE`, include person-measure recovery rows when the fitted object exposes person estimates.

`include_diagnostics` Logical. When `TRUE`, run `diagnose_mfrm()` after each successful refit and retain facet-level fit/separation operating characteristics. These diagnostics are reported separately from recovery metrics and are not parameter-recovery criteria by themselves.

`diagnostic_fit_df_method` Fit-ZSTD degrees-of-freedom convention used for optional diagnostic operating-characteristic summaries. Use `"both"` when reviewing FACETS-style df sensitivity.

`seed` Optional random seed.

file	Optional RDS path for saving and reusing the simulation. A missing .rds extension is appended. Its parent directory must already exist. NULL (default) performs no file reads or writes.
file_refit	File reuse policy. "never" (default) loads a matching completed result or resumes matching saved trials; incompatible files produce an error. "on_change" restarts when inputs or implementation/ environment differ. "always" replaces the file with a new run. Unlike brms::brm(file_refit = "never"), this function checks compatibility even under "never"; it does not silently return a different analysis.

Details

This helper is deliberately narrower than evaluate_mfrm_design(). Design evaluation asks which design condition is operationally adequate; recovery simulation asks whether the fitted model recovers the known parameters under one explicit data-generating setup.

Location-like parameters (Person, non-person facets, and steps) are summarized after mean alignment within each replication and parameter group. These errors describe relative locations within the matched group. Their average Bias is zero by construction, so it cannot establish absence of bias for individual parameters. BiasBasis identifies this restriction; assess_mfrm_recovery() leaves its bias check not_assessed. Inspect RMSE and row-level errors for relative recovery instead.

PCM/GPCM step vectors are centered within their owning facet level. Their generating step means are therefore added to the corresponding facet truth before comparison. A generated population intercept, when fitted, is shifted consistently with the fitted facet/step identification. RawTruth retains the original generator parameter. Truth is the comparison-coordinate truth; ErrorRaw, RawBias and RawRMSE use this truth before the additional within-group mean alignment, not necessarily RawTruth.

For GPCM, supplied generator slopes are treated as relative discriminations and normalized to the same geometric-mean-one log-slope identification used by the fitter. Slope recovery is therefore summarized on the identified log-slope scale without an additional mean-alignment step. Average bias over a complete log-slope block is also zero by identification and is not an automatic adequacy criterion. Direct data generation and refitting are supported, but broader GPCM design- planning claims remain outside the current package boundary.

Sparse linked generators are supported through sim_spec or direct assignment = "sparse_linked" plus sparse_controls. Their design-density and rater-link diagnostics are retained in rep_overview; recovery metrics remain parameter-recovery summaries and should not be read as evidence that a sparse linking design is adequate by itself.

The returned ademp component follows the simulation-study framing of Morris, White, and Crowther (2019) and the ADEMP planning/reporting template used in later simulation-study guidance.

Value

An object of class mfrm_recovery_simulation with components:

- recovery: row-level truth/estimate comparisons and interval availability by replication.
- recovery_summary: parameter-group metrics, trial MCSEs, denominators, availability rates and interval reasons.
- recovery_targets: target counts by replication and parameter group, recorded before fitting.

- `rep_overview`: replication-level convergence, timing, error status, and sparse-design diagnostics when applicable.
- `diagnostic_oc`: optional replication-by-facet fit/separation operating characteristics when `include_diagnostics = TRUE`.
- `diagnostic_oc_summary`: optional facet-level diagnostic operating- characteristic summary.
- `settings`: fitting and simulation settings.
- `ademp`: simulation-study metadata.
- `reproducibility`: file identity, original inputs and per-trial seeds when `file` is supplied.

Saving and resuming simulations

Pass the same `file` and analysis arguments on a subsequent call to reuse the completed result without generation or fitting, or to resume an interrupted run. The identity includes generator inputs/specification, fitting controls, requested replications, seed, RNG kind, package R code and native library fingerprints, relevant numerical/diagnostic R options, R/platform, numerical libraries and dependency versions. Changing `reps` starts a different study; extension of an existing study is not supported. `progress` and `parallel` do not change the identity. Cached results retain the settings of their original run.

Recovery runs and serial design evaluations save after every finished trial. Future design evaluations save after a whole design row returns; an interrupted row may need to run again. A partially completed optimizer is not resumed. Fit failures are completed attempts and are saved too; resuming does not redraw failures until they succeed. An unexpected error outside the recorded fit/diagnostic failure routes stops the call and leaves earlier checkpoints available. These files do not change the interpretation or eligibility of the underlying estimates and intervals.

Per-trial seeds are allocated before computation and retained even when `seed = NULL`; an explicit seed is preferable for reproducing a new run in another file. Each trial controls generation, fitting and diagnostics with its own seed. A matching environment is required for automatic reuse; recorded seeds alone do not guarantee identical floating-point results on other systems. Keep the analysis script and software environment as well.

After completion, `readRDS(file)` returns the usual result object with an additional `reproducibility` component containing identity, inputs and trial seeds. Before completion the file is an internal checkpoint; resume through the same public function. Writes use a temporary file in the same directory and checked replacement. A `.lock` directory prevents concurrent writers. Normal returns/errors release it; after process termination, remove a remaining lock only after checking that no writer is active. Unreadable files are never silently reused or replaced by `"on_change"`.

Monte Carlo uncertainty and missing results

Replications, not parameter rows within a replication, are independent. For a pooled available-row mean, let S_r be the sum of values in trial r, n_r its available-row count and $m = \sum_r S_r / \sum_r n_r$. With R recorded trials, the ratio-linearization MCSE is

$$\mathrm{MCSE}(m) = \frac{\sqrt{R/(R-1)\sum_r (S_r - m n_r)^2}}{\sum_r n_r}.$$

Failed trials contribute zero sums and counts. RMSE uses squared errors and the delta-method factor $1/(2\sqrt{m})$; `McseCoverage95` applies the same ratio calculation to coverage indicators. Fewer

than two contributing trials gives NA, as does the RMSE delta MCSE at zero RMSE. These are large-replication approximations; small examples do not qualify performance.

recovery_targets records expected target counts after generation, before fitting. Summaries retain AttemptedReps, FailedReps, ExpectedRows, EstimateRows, SEAvailableRows, CoverageRows and CoverageReps. Coverage95 is conditional on an available interval: report it with CoverageAvailabilityRate and failures. Unavailable intervals are not counted as noncoverage. Availability rates use all expected targets, including failed fits. A generation failure or legacy result can leave the roster unknown; then ExpectedRows and full-denominator availability rates are NA, while KnownTargetRows and KnownTargetReps remain visible.

Interval targets

For non-Person facets, coverage concerns each level minus its facet mean. With $P = I - 11^T/m$, its covariance is PVP^T; the original marginal SE cannot be reused after alignment. The runner obtains model-based 95\ its numerical/source checks. A complete matched facet roster is required. The row table retains CoverageEstimate, CoverageTruth, SE, Lower95, Upper95, CIEligible and CIReason. RawSE separately preserves the source uncertainty. A valid target SE can remain available when its interval bounds cannot be represented.

JML remains a point-recovery route. Person-score uncertainty is conditional or posterior score uncertainty, not the centered structural interval above. Their coverage is unavailable with an explicit reason; step, slope and population recovery also retain point comparisons without a matching interval implementation here. This does not alter uncertainty available through other model-specific APIs. Numerical eligibility alone does not establish finite-sample coverage or robustness to model misspecification.

Previously saved recovery results

summary(), assess_mfrm_recovery() and summary/coverage plot() routes recompute aggregates from saved rows without fitting. They correct trial MCSEs and withhold old coverage whose centered covariance was not retained. They cannot repair the identified PCM/population truth from old row tables; start a new study to obtain those corrections and matched interval evidence. Read summary(x)$notes and the new availability/reason columns. Directly reading an old x$recovery_summary does not update that stored table. The default file reuse policy rejects the changed implementation identity; use a new file path to preserve the old study.

Typical workflow

1. Build a simulation specification with build_mfrm_sim_spec() or pass scalar generator arguments directly.
2. Run simulate_mfrm_recovery(...) with a modest reps value for an initial end-to-end review, then increase reps for stable Monte Carlo summaries.
3. Inspect summary(x)$recovery_summary and the row-level x$recovery table.

References

Morris, T. P., White, I. R., and Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11), 2074–2102. doi:10.1002/sim.8086.

See Also

simulate_mfrm_data(), evaluate_mfrm_design(), fit_mfrm(), assess_mfrm_recovery(), mfrmr_compatibility_layer

Examples

```
# Long-running workflow; run explicitly when needed.
# example("simulate_mfrm_recovery", run.dontrun = TRUE)
## Not run:
saved <- tempfile(fileext = ".rds")
rec <- simulate_mfrm_recovery(
  n_person = 12,
  n_rater = 2,
  n_criterion = 2,
  reps = 1,
  maxit = 30,
  seed = 123,
  file = saved
)
summary(rec)$recovery_summary[, c("ParameterType", "Facet", "RMSE",
  "McseRMSE", "ExpectedRows", "CoverageRows", "Coverage95")]

# A second identical call reads the result without fitting again.
again <- simulate_mfrm_recovery(n_person = 12, n_rater = 2, n_criterion = 2,
  reps = 1, maxit = 30, seed = 123, file = saved)
identical(rec, again)
unlink(saved)

## End(Not run)
```

---

specifications_report *Build a specification summary report (preferred alias)*

Description

Build a specification summary report (preferred alias)

Usage

```
specifications_report(
  fit,
  title = NULL,
  data_file = NULL,
  output_file = NULL,
  include_fixed = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
title	Optional analysis title.
data_file	Optional data-file label (for reporting only).
output_file	Optional output-file label (for reporting only).
include_fixed	If TRUE, include a legacy-compatible fixed-width text block.

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_specifications (type = "facet_elements", "anchor_constraints", "convergence").

Value

A named list with specification-report components. Class: mfrm_specifications.

Interpreting output

- header / data_spec: run identity and model settings. CenteringScope is "facet" for whole-facet centering or "remainder" for the legacy constraint on ungrouped, unanchored levels. It applies only to centered facets; NonCenteredFacet remains separate. Older native fits without convention metadata retain "remainder"; an input without a stored constraint specification is "not_recorded".
- facet_labels: facet sizes and labels.
- convergence_control: optimizer configuration and status.

Typical workflow

1. Generate specifications_report(fit).
2. Verify model settings and convergence metadata.
3. Use the output as methods and run-documentation support in reports.

See Also

fit_mfrm(), data_quality_report(), estimation_iteration_report(), mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("specifications_report", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_operational")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 7, maxit = 30
)
```

```
out <- specifications_report(fit, title = "Toy run")
summary(out)
p_spec <- plot(out, draw = FALSE)
p_spec$data$plot

## End(Not run)
```

`subset_connectivity_report` *Build a subset connectivity report (preferred alias)*

Description

Build a subset connectivity report (preferred alias)

Usage

```
subset_connectivity_report(
  fit = NULL,
  diagnostics = NULL,
  top_n_subsets = NULL,
  min_observations = 0,
  data = NULL,
  person = NULL,
  facets = NULL,
  score = NULL,
  basis = c("observed", "planned"),
  weight = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`top_n_subsets`	Optional maximum number of subset rows to keep.
`min_observations`	Minimum observations required to keep a subset row.
`data`	Optional raw assignment/rating data, supplied instead of `fit`.
`person, facets`	Names of the person column and one or more facet columns in `data`. Include only Rater in `facets` to inspect person-rater links without a common Criterion node obscuring separate panels.
`score`	Numeric score column for `basis = "observed"`. Recode study-specific missing markers to `NA` first; no category recoding or estimation is done.
`basis`	`"observed"` includes rows with complete identifiers and nonmissing finite scores; `"planned"` includes complete assignment rows without scores. Planned networks require `score = NULL` and `weight = NULL`.

`weight` Optional numeric weight column for observed raw data. Missing and zero-weight rows do not create edges. Edge weights still count rows, not sampling weights. Negative or infinite weights are refused.

Details

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_subset_connectivity` (`type = "subset_observations"`, `"facet_levels"`, or `"linking_matrix"` / `"coverage_matrix"` / `"design_matrix"` / `"network"`). The network route returns reusable node and edge tables with `draw = FALSE`; drawing uses `igraph` when available.

Value

A named list with subset-connectivity components. Class: `mfrm_subset_connectivity`.

Interpreting output

- `summary`: number and size of connected subsets.
- subset table: whether data are fragmented into disconnected components.
- facet-level columns: where connectivity bottlenecks occur.

Typical workflow

1. Run `subset_connectivity_report(fit)`.
2. Inspect graph coverage alongside `identification_audit`. A common criterion can connect panels whose person/rater locations remain confounded.
3. Use the model rank and population assumptions to assess linking; a single co-observation component is not a FACETS identification subset.

Before fitting

Supply `data`, `person`, `facets` and `score` instead of `fit`. Known levels from incomplete/unscored rows remain as nodes, including isolates. A planned assignment roster can be examined with `basis = "planned"`; absent assignments cannot be reconstructed from an observed-only file. `row_usage` records each input row's inclusion and first exclusion reason. With a fit, only its retained rows are available here; review original missing assignments before fitting. No prefit result supplies an identification audit or model-readiness decision. `settings$is_disjoint` describes the source graph, even when subset filters omit components; compare source/retained subset counts. Filters do not establish linking. See `vignette("mfrmr-networks-and-reporting")` for a panel example.

See Also

`diagnose_mfrm()`, `mfrm_network_analysis()`, `measurable_summary_table()`, `data_quality_report()`, mfrmr_linking_and_dff, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("subset_connectivity_report", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
out <- subset_connectivity_report(fit)
summary(out)
p_sub <- plot(out, draw = FALSE)
p_design <- plot(out, type = "design_matrix", draw = FALSE)
p_net <- plot(out, type = "network", draw = FALSE)
p_sub$data$plot
p_design$data$plot
p_net$data$edges
out$summary[, c("Subset", "Observations", "ObservationPercent")]

## End(Not run)
```

`summary.mfrm_bundle` *Summarize report/table bundles in a user-friendly format*

Description

Summarize report/table bundles in a user-friendly format

Usage

```
## S3 method for class 'mfrm_bundle'
summary(object, digits = 3, top_n = 10, include_person = FALSE, ...)
```

Arguments

`object`	Any report bundle produced by `mfrmr` table/report helpers.
`digits`	Number of digits for printed numeric values.
`top_n`	Number of preview rows shown from the main table component.
`include_person`	If `TRUE`, person-level identifiers may appear in supported summary and preview tables. The default suppresses identifier columns, person-facet level labels, and ConQuest case labels from the returned summary object and its console output. This does not alter the source bundle.
`...`	Reserved for generic compatibility.

Details

This method provides a compact summary for bundle-like outputs (for example: unexpected-response, fair-average, chi-square, and category report objects). It extracts:

- object class and available components
- one-row summary table when available
- preview rows from the main data component
- resolved settings/options

Branch-aware summaries are provided for:

- `mfrm_bias_count` (`branch = "original" / "facets"`)
- `mfrm_fixed_reports` (`branch = "original" / "facets"`)
- `mfrm_visual_summaries` (`branch = "original" / "facets"`)

Additional class-aware summaries are provided for:

- `mfrm_unexpected`, `mfrm_fair_average`, `mfrm_displacement`
- `mfrm_interrater`, `mfrm_facets_chisq`, `mfrm_bias_interaction`
- `mfrm_rating_scale`, `mfrm_category_structure`, `mfrm_category_curves`
- `mfrm_measurable`, `mfrm_unexpected_after_bias`, `mfrm_output_bundle`
- `mfrm_residual_pca`, `mfrm_specifications`, `mfrm_data_quality`, `mfrm_fit_measures`
- `mfrm_iteration_report`, `mfrm_subset_connectivity`, `mfrm_facet_statistics`
- `mfrm_facets_contract_review`, `mfrm_facets_fit_review`, `mfrm_facets_fit_df_guide`, `mfrm_reference_benchmark`

Value

An object of class `summary.mfrm_bundle`.

Interpreting output

- `overview`: class, component count, and selected preview component.
- `summary`: one-row aggregate block when supplied by the bundle.
- `preview`: first `top_n` rows from the main table-like component.
- `settings`: resolved option values if available.
- `validation_scope`: package-generated versus independently supplied evidence scope when summarizing `mfrm_reference_benchmark`.
- `conquest_command_scope`: ConQuest command-template scope when summarizing `mfrm_conquest_overlap_bundl`
- `conquest_output_contract`: requested ConQuest outputs and review handoff when summarizing `mfrm_conquest_overlap_bundle`.
- `mfrmr_fit_status`: actual optimizer controls, MML engine, convergence evidence, and inference-readiness state for an `mfrm_conquest_overlap_bundle`.
- `normalization_scope`: extracted-table normalization scope when summarizing `mfrm_conquest_overlap_tables`.
- `review_scope`: supplied-table review scope when summarizing `mfrm_conquest_overlap_review`.
- `conquest_overlap_checks` / `population_policy_checks`: specialized benchmark check previews when summarizing `mfrm_reference_benchmark`.

Typical workflow

1. Generate a bundle table/report helper output.
2. Run `summary(bundle)` for compact QA.
3. Drill into specific components via `$` and visualize with `plot(bundle, ...)`.

See Also

`unexpected_response_table()`, `fair_average_table()`, `plot()`

Examples

```
# Long-running workflow; run explicitly when needed.
# example("summary.mfrm_bundle", run.dontrun = TRUE)
## Not run:
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
t4 <- unexpected_response_table(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 5)
summary(t4)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t11 <- bias_count_table(bias, branch = "facets")
summary(t11)

## End(Not run)
```

---

summary.mfrm_design_evaluation

Summarize a design-simulation study

Description

Summarize a design-simulation study

Usage

```
## S3 method for class 'mfrm_design_evaluation'
summary(object, digits = 3, ...)
```

Arguments

`object`	Output from `evaluate_mfrm_design()`.
`digits`	Number of digits used when printing the summary. Returned numeric tables retain full precision for plotting and design decisions.
`...`	Reserved for generic compatibility.

Details

The summary emphasizes condition-level averages that are useful for practical design planning, especially:

- convergence rate
- separation and reliability by facet
- severity recovery RMSE
- mean misfit rate

`Reps`, `ConvergenceRate`, and `McseConvergenceRate` use all recorded replications for each design, including fit and diagnostic failures. Only a recorded `Converged = TRUE` counts as converged. `AvailableReps` counts the replications with returned results for that facet; performance means still use the available metric values. Designs with no returned facet results have no performance-summary rows and cannot be recommended. To update an older saved summary, call `summary()` again on the original evaluation object; no simulation or refitting is needed. Rebuilding also restores full precision when an older summary stored rounded metrics. It also withholds legacy PCM/GPCM recovery metrics whose step-location truth coordinates were not verified; those require a new evaluation for corrected recovery, while other diagnostic metrics remain. `RecoveryAvailableReps` reports the number of finite trial RMSEs. Mean-aligned bias is zero by construction; see `evaluate_mfrm_design()` for the comparison target.

`MaxRatings` and `MaxRatingsPerRater` are the largest total rating count and individual rater workload across all recorded replications, including failed runs. A rating is one generated person-rater-criterion row; weights do not multiply workload. A maximum is `NA` when any replication lacks that count, including older evaluations without per-rater workload records.

`MaxRaterComponents` and `MaxCriterionComponents` give the largest Person-facet component count across all recorded assignments; a missing count makes its maximum `NA`. `DisconnectedReps` counts replications with a recorded disconnection in either graph. Zero known disconnections does not establish connectedness when records are missing. Sparse-design summaries also use all run records, including failures.

Value

An object of class `summary.mfrm_design_evaluation` with components:

- `overview`: run-level overview
- `design_summary`: aggregated design-by-facet metrics, with design-variable alias columns when applicable
- `sparse_review`: compact planned-missingness and rater-link review counts when sparse linked designs are active
- `ademp`: simulation-study metadata carried forward from the original object
- `facet_names`: public facet labels carried from the simulation specification
- `design_variable_aliases`: accepted public aliases for design variables
- `design_descriptor`: role-based design-variable metadata
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of mutable/locked design variables

- planning_schema: structured planning metadata
- structural_design_review: deterministic structural review of the named-facet design grid; it reports design bookkeeping rather than simulation performance
- notes: short interpretation notes

See Also

evaluate_mfrm_design(), plot.mfrm_design_evaluation

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_design_evaluation", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-design.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
sim_eval <- saved$result
s <- summary(sim_eval)
s$overview
head(s$design_summary)

## End(Not run)
```

---

summary.mfrm_diagnostics

Summarize an mfrm_diagnostics *object in a user-friendly format*

Description

Summarize an mfrm_diagnostics object in a user-friendly format

Usage

```
## S3 method for class 'mfrm_diagnostics'
summary(
  object,
  digits = 3,
  top_n = 10,
  detail = c("brief", "full"),
  include_person = FALSE,
  ...
)
```

Arguments

object	Output from diagnose_mfrm().
digits	Number of digits for printed numeric values.
top_n	Number of highest-absolute-Z fit rows to keep.
detail	Console detail: "brief" (default) prints the first-screen review; "full" prints the additional structured tables.
include_person	If TRUE, person-level identifiers may be printed in fit-review tables. The default keeps identifiers out of console output; person-level rows remain available in the returned object.
...	Reserved for generic compatibility.

Details

This method returns a compact diagnostics summary designed for quick review:

- design overview (observations, persons, facets, categories, subsets)
- diagnostic-basis guide for legacy versus strict fit paths
- global fit statistics
- approximate reliability/separation by facet
- top facet/person fit rows by absolute ZSTD
- counts of flagged diagnostics (unexpected, displacement, interactions)

Value

An object of class summary.mfrm_diagnostics with:

- overview: design-level counts and residual-PCA mode
- decision: the same plain-language fit-readiness decision used by summary(fit), retained ahead of diagnostic screening results
- fit_readiness, fit_readiness_components, and fit_readiness_parameters: readiness provenance inherited from the source fit and retained separately from diagnostic-screening status
- status: concise front-door status block for quick review
- key_warnings: highest-priority warnings to review first
- next_actions: recommended follow-up helpers
- diagnostic_basis: guide to legacy versus strict diagnostic targets
- fit_standardization: guide to the df convention used for fit ZSTD
- overall_fit: global fit block
- precision_profile: design-weighted precision summary across the information curve at decile theta points
- precision_review: separation / reliability / strata review for the sample- and population-basis modes (paired with precision_profile)
- reliability: facet-level separation/reliability summary

- facets_chisq: facets-style fixed-effect chi-square heterogeneity screen across non-person facets
- interrater: inter-rater agreement / pairwise correlation / rater separation overview when a Rater facet is present
- misfit_flagged: rows flagged by the Infit / Outfit mean-square band
- fit_screening: counts of all, classified and unclassified elements for the mean-square band and each ZSTD cutoff. Either available statistic crossing a cutoff flags the element; otherwise a missing statistic leaves it unclassified. Rates require every element to be classified. Nonfinite statistics and negative mean squares are unavailable, not passing values.
- misfit_thresholds: named numeric vector with the misfit lower / upper thresholds used to populate misfit_flagged
- category_usage: per-category response-frequency summary used to flag empty / collapsed categories
- top_fit: top maximum |ZSTD| rows with both statistics available
- marginal_coverage: classified, unclassified and flagged category cells, groups and level pairs; counts of available flags do not describe missing results
- marginal_fit: optional strict marginal-fit overview when requested
- top_marginal_cells: largest strict marginal residual cells when requested
- marginal_pairwise: optional strict pairwise local-dependence overview
- top_marginal_pairs: largest strict pairwise residual summaries
- marginal_guidance: interpretation labels for strict marginal diagnostics
- reporting_map: manuscript-oriented guide to what is covered here versus which companion outputs should be consulted
- flags: compact flag counts for major diagnostics
- notes: short interpretation notes
- digits: numeric-print precision threaded through to print.summary.mfrm_diagnostics()

Interpreting output

- overview: analysis scale, subset count, and residual-PCA mode.
- fit_readiness: the source fit's versioned readiness row. Diagnostics do not promote a blocked or review-only fit to inferential use.
- diagnostic_basis: plain-language map of which fit path was computed and what each path means statistically.
- overall_fit: global fit indices.
- reliability: facet separation/reliability block, including model and real bounds when available.
- top_fit: highest |ZSTD| elements for immediate inspection.
- flags: compact counts for key warning domains.
- fit_screening: available classifications and unclassified elements for each mean-square/ZSTD rule. A known threshold crossing remains flagged even if the other statistic is missing. Recreate older summaries and reports from existing diagnostics; no MFRM refit is required.

Typical workflow

1. Run diagnostics with diagnose_mfrm(), using diagnostic_mode = "both" for RSM / PCM when you want legacy continuity plus strict marginal screening.
2. Review summary(diag) for major warnings and inspect diagnostic_basis before comparing legacy and strict outputs.
3. Follow up with dedicated tables/plots for flagged domains.

See Also

diagnose_mfrm(), summary.mfrm_fit()

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit

# Check model fit and the support for standard errors and intervals
diagnostics <- example_fit$diagnostics
diagnostic_summary <- summary(diagnostics)
diagnostic_summary$decision

diagnostic_summary$key_warnings # Issues to investigate, if present
diagnostic_summary$top_fit      # Most unusual residual-based fit statistics

# Distinguish residual-based checks from marginal model checks
diagnostic_summary$diagnostic_basis[, c("DiagnosticPath", "Status", "Basis")]
```

---

summary.mfrm_diagnostic_screening

Summarize a diagnostic-screening simulation study

Description

Summarizes output from evaluate_mfrm_diagnostic_screening() for reporting, appendix export, and draw-free visualization handoff. The summary keeps simulation operating characteristics separate from inferential conclusions: fit, marginal, pairwise, and report-review signals are screening readouts rather than pass/fail evidence.

Usage

```
## S3 method for class 'mfrm_diagnostic_screening'
summary(object, digits = 3, ...)
```

Arguments

object	Output from evaluate_mfrm_diagnostic_screening().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_diagnostic_screening with:

- overview: run-level design, replication, convergence, and report-review metadata
- reading_order: recommended order for reading the summary tables
- next_actions: action-oriented triage for interpreting and exporting the summary
- reporting_notes: report-facing boundaries and recommended wording safeguards
- figure_recipes: recommended figure/display recipes for the draw-free plot-data tables
- scenario_summary: aggregated scenario-by-design screening summaries
- performance_summary: operating-characteristic rates and runtime summaries
- report_signal_summary: optional mfrm_report() readiness/review signals
- scenario_contrast: misspecification-minus-well-specified contrasts
- plot_*: long-form draw-free plot tables for overview, report, contrast, and runtime views
- planning metadata, settings, ADEMP metadata, and interpretation notes

See Also

evaluate_mfrm_diagnostic_screening(), plot.mfrm_diagnostic_screening, plot_data()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_diagnostic_screening", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-diagnostic-screening.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
diag_eval <- saved$result
summary(diag_eval)

## End(Not run)
```

summary.mfrm_facets_run *Summarize a legacy-compatible workflow run*

Description

Summarize a legacy-compatible workflow run

Usage

```
## S3 method for class 'mfrm_facets_run'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from run_mfrm_facets().
digits	Number of digits for numeric rounding in summaries.
top_n	Maximum rows shown in nested preview tables.
...	Passed through to nested summary methods.

Details

This method returns a compact cross-object summary that combines:

- model overview (objectfitsummary)
- resolved column mapping
- run settings (run_info)
- nested summaries of fit and diagnostics

Value

An object of class summary.mfrm_facets_run.

Interpreting output

- overview: convergence, scale size, and the canonical/legacy information-criterion contract status.
- mapping: sanity check for auto/explicit column mapping.
- fit / diagnostics: drill-down summaries for reporting decisions.

Typical workflow

1. Run run_mfrm_facets() to execute a one-shot pipeline.
2. Inspect with summary(out) for mapping and convergence checks.
3. Review nested objects (outfit, outdiagnostics) as needed.

See Also

run_mfrm_facets(), summary.mfrm_fit(), mfrmr_workflow_methods, summary()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_facets_run", run.dontrun = TRUE)
## Not run:
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:8], , drop = FALSE]
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
s <- summary(out)
s$overview[, c("Model", "Method", "Converged")]
s$mapping

## End(Not run)
```

---

summary.mfrm_fit *Summarize an* mfrm_fit *object in a user-friendly format*

Description

Summarize an mfrm_fit object in a user-friendly format

Usage

```
## S3 method for class 'mfrm_fit'
summary(
  object,
  digits = 3,
  top_n = 5,
  ...,
  profile = c("fit", "facets", "reporting"),
  detail = NULL,
  diagnostics = NULL,
  compute = c("auto", "never"),
  include_person = FALSE,
  comparison_review = NULL
)
```

Arguments

object	Output from fit_mfrm().
digits	Number of digits for printed numeric values.
top_n	Number of extreme facet/person rows shown in summaries.
...	Reserved for generic compatibility. The workflow arguments that follow ... must be supplied by name.
profile	Summary profile. "fit" preserves the lightweight fit-only contract and does not compute diagnostics. "facets" adds a comprehensive measurement review using familiar FACETS-style section organization; it does not require FACETS knowledge or software. "reporting" adds the reporting-oriented results profile.
detail	Printed detail. When NULL (the default), the lightweight "fit" profile retains the legacy "full" print while expanded profiles use "brief". Neither mode prints person identifiers unless include_person = TRUE; "brief" also reduces the number of fit-level sections shown in the console.
diagnostics	Optional matching output from diagnose_mfrm(). It is reused by the "facets" and "reporting" profiles without recomputation.
compute	Diagnostic computation policy for the expanded profiles. "auto" computes diagnostics once when they were not supplied; "never" returns the available fit-only portions and marks every requested dependent section as "not_computed". The "fit" profile never computes diagnostics.
include_person	Logical. Whether person identifiers may be printed in extreme-person tables and requested by the fit-pathway route. The default is FALSE for privacy-safe console output.
comparison_review	Optional prospective interaction review from describe_mfrm_data(). Explicitly requests fitted contrasts and retained- point numerical checks through interaction_effect_table() saved as $interaction_comparisons. It does not fit a model or supply intervals.

Details

Start with results <- summary(fit). Use results$person_overview for the distribution of person ability estimates and results$facet_overview for the distribution of estimates within each non-person facet. These are aggregate summaries; use as.data.frame(fit) for individual person and facet-level estimates. Read results$decision before interpreting results. Assignment with <- saves the summary without printing it; enter results to print the full summary or use $ to display a selected table. In the example, person_overview has one row for all persons and facet_overview has one row for raters and one for criteria. Each non-person facet's mean is constrained to zero in this fit; use its SD and range or individual estimates to inspect differences among its levels.

This method provides a compact, human-readable summary oriented to reporting. The expanded profiles use FACETS-style organization for navigation, but do not claim that FACETS was executed or that estimates are numerically equivalent to FACETS output. It returns a structured object and prints:

- model fit overview (N, LogLik, the canonical/legacy IC status, convergence)

- estimation settings that affect identification/scoring interpretation
- facet-level estimate distribution (mean/SD/range)
- person measure distribution
- step/threshold checks
- a reporting map showing which companion summaries/tables should be used for manuscript-oriented data description, diagnostics, category checks, and draft reporting
- extreme facet levels and, when explicitly requested, high/low person measures

Value

An object of class `summary.mfrm_fit` with:

- `overview`: global model/fit indicators. For MML, `MMLEngineUsed` records the actual algorithm, `IterationsBasis` explains what is counted, and `ConvergenceBasis` identifies the numerical stopping rule. Numerical convergence does not establish identification or interval eligibility
- `status`: concise front-door status block for quick review
- `decision`: plain-language interpretation, formal-inference status, reason, and highest-priority next action derived from the stored readiness contract plus supplied precision evidence; fit readiness alone never yields `FormalInference = "Yes"`
- `readiness`: the stored fit-level state plus numerical, data, design, stability, diagnostic, and reporting workflow states
- `data_review`: structured connectivity and facet-support evidence used by the non-numerical readiness requirements
- `key_warnings`: highest-priority warnings to review first
- `next_actions`: recommended follow-up helpers
- `population_overview`: current population-model basis, residual variance, and omission review
- `population_coefficients`: fitted latent-regression coefficients when a population model is active
- `population_design`: latent-regression design-matrix column check when a population model is active
- `population_coding`: categorical covariate levels and contrast provenance when a population model uses model-matrix coding
- `facet_overview`: per-facet estimate distribution summary
- `person_overview`: person-measure distribution summary with the actual aggregation denominator, blocked extreme-EAP exclusion count, and estimate use
- `targeting`: person-versus-non-person facet targeting overview (Wright-map-style mean/SD comparison)
- `step_overview`: threshold/step diagnostics by PCM/GPCM `StepFacet` ladder, or for the common RSM ladder

- slope_overview: parameter-readiness and explicitly labelled optimizer- trace summary for GPCM discriminations. SlopeOwner identifies the facet whose levels carry slopes. Min, Max and GeometricMean summarize primary estimates; the Optimizer* fields retain numerical iterates when primary estimates are unavailable. The scale reference is an identification constraint, not evidence that estimates or intervals are reliable
- inference_evidence: for GPCM MML, a compact separation of optimizer stationarity, retained-point local rank, observed-information curvature, slope-boundary screening, and the final readiness decision. Supportive local evidence does not override an inconclusive boundary audit
- interaction_overview: model-estimated facet-interaction summary when the fit was specified with facet_interactions
- interaction_comparisons: explicitly requested joint-fit contrasts, retaining unavailable rows, the prospective review and numerical checks; absent unless comparison_review is supplied
- settings_overview: estimation-settings overview that pins the configuration that affects identification/scoring. CenteringScope distinguishes whole-facet ("facet") from legacy remainder centering ("remainder"), separately from NoncenterFacet. Native fits saved before convention metadata retain the legacy interpretation. Recreate this summary from the fit to obtain new fields; reading an old summary does not update it or refit the model
- numerical_review: separate optimization, curvature, starting-value, integration and reproducible-refit evidence with specific next actions. Missing checks remain unassessed; this table does not change readiness
- starting_values, starting_value_notes: all attempted starts, numerical outcomes, selection and interpretation limits when fit_mfrm(mml_starts = ...) was requested. These describe numerical sensitivity, not sampling uncertainty
- attached_diagnostics: logical flag indicating whether the mfrm_fit was returned with diagnostics already attached
- attached_diagnostics_cols: character vector of diagnostic columns attached to fit$facets$person when attached_diagnostics = TRUE
- row_retention: row counts before and after preparation filters
- preparation_notes: structured preparation notes retained from fit$prep
- reporting_map: routing map showing which companion summaries and tables should be used for the four manuscript-oriented reporting sections (data description, diagnostics, category checks, draft reporting)
- person_high / person_low: highest and lowest person measures
- facet_extremes: extreme facet-level estimates
- facet_support_boundaries: observed boundary-constant non-person facet levels, kept distinct from parameter-level recession conclusions
- facet_recession_review: certified JML additive facet recession directions, including review scope and completeness
- caveats: structured warning/review rows for score-support and latent-regression population-model issues
- notes: short interpretation notes
- digits: numeric-print precision threaded through to print.summary.mfrm_fit()

- section_status: availability and explicit non-computation boundaries
- required_visual: ordered Wright-map and Infit-pathway routes
- provenance: profile, diagnostic source, computation policy, and the FACETS-organization interpretation boundary
- analysis: compact fit/results indexes used for first-screen review
- results: the reused mfrm_results backend for expanded profiles, or NULL for the lightweight "fit" profile

Interpreting output

Corrected JML has a separate summary of saved tables and numerical attempts, with the correction order and the meaning of RootSE. All profiles show the same saved summary and never compute ordinary diagnostics. Use include_person = TRUE for conditional Person profiles without Person SEs. See the **Corrected JML** section of fit_mfrm(). The entries below describe the ordinary, uncorrected fitting routes.

- overview: convergence plus the versioned information-criterion contract. For eligible fixed-facet MML fits, BIC/SABIC use unique Persons, not response rows; JML, non-unit observation weights, and legacy objects do not enter the common MML ranking panel. ICSelectable additionally distinguishes raw screening/review criteria at q<31 from criteria that may enter automatic same-grid comparison at q>=31; close decisions still require a denser common-grid sensitivity check.
- readiness: the stored fit-readiness result followed by Numerical, Data, Design, Stability, Diagnostics, and Reporting workflow states. InferenceReady is a conservative compatibility scalar and is TRUE only when the stored FitReadiness is ready; numerical convergence cannot override input, estimability, category, or boundary review.
- decision: separates fit readiness from formal precision support. A fit-only summary returns FormalInference = "No" until a matching mfrm_diagnostics object is supplied through diagnostics =; use summary(diagnostics)$decision for the equivalent precision-aware view. The console uses this single decision throughout; a converged optimizer or a passed fit check does not independently authorize formal inference. Printed workflow, population and GPCM descriptions use readable labels. The returned tables retain their numerical values and structured status fields for programmatic use. Reprinting an existing summary updates its display without refitting or changing its stored results.
- data_review: overall multi-facet connectivity, facet-level score support, boundary-constant levels, single-level facets, and retained preparation notes behind the readiness rows.
- facet_overview: per-facet spread and range of estimates.
- person_overview: distribution of person measures. For a blocked source fit, a prior-regularized extreme MML EAP is retained in person_high / person_low but excluded from this aggregate and targeting; the table records its distribution denominator, exclusion count, and estimate use.
- step_overview: threshold spread and monotonicity checks, reported by StepFacet ladder for PCM/GPCM fits and as one common ladder for RSM fits.
- settings_overview: estimation settings that affect interpretation. CategoryPolicy records "preserve" or "collapse"; ScoreRecoded states whether the stored original-to-internal score map changes values. A collapse policy need not recode a complete contiguous scale.

For older saved fits, absent policy or mapping records give "not_recorded" or NA, respectively; an unchanged score map does not identify the chosen policy. For MML fits, the printed fit and summary also state the engine, fixed or adaptive Gauss–Hermite rule and order, one-dimensional latent structure, population identification, and discrimination constraint.

- population_coding: fitted categorical levels and contrasts that must be reused when scoring new persons under the population-model posterior.
- key_warnings / notes: short triage subset of retained zero-count score categories and latent-regression population-model caveats such as complete-case omissions, zero-variance design columns, missing coefficients, or unstable residual variance when present. Incomplete or non-finite covariates are normally handled before fitting as input errors or complete-case omissions; they appear here only if retained in a population-design check row.
- caveats: structured rows behind those warnings for appendix/export use; print(summary(fit)) shows a compact Caveats block when rows are present.
- reporting_map: where to get companion outputs for manuscript reporting.
- person_high / person_low (opt-in for printing) and facet_extremes: extreme estimates for focused review.
- facet_support_boundaries: observed non-person facet levels whose retained responses are constant at the minimum or maximum score. This is a data- support warning, not by itself a proof that a parameter MLE is infinite.
- facet_recession_review: non-person facet directions certified as unbounded in an evaluated JML additive recession subspace. Joint rows are relative directions under the fitted identification constraints.

Typical workflow

1. Review data and score support with describe_mfrm_data().
2. Fit with fit_mfrm() and read summary(fit, profile = "fit").
3. Request summary(fit, profile = "facets") for the comprehensive measurement review. The historical profile name does not mean that FACETS is run.
4. Draw the required native Wright map with plot(fit, type = "wright", show_ci = TRUE); add the FACETS renderer or Infit pathway only when they answer a specific follow-up question.
5. For RSM / PCM, continue with diagnose_mfrm() for element-level fit checks. For GPCM, continue with compute_information() / plot(x) or the fitted-object posterior scoring helpers.

See Also

fit_mfrm(), diagnose_mfrm()

Examples

```
library(mfrmr)

# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
```

```
fit <- example_fit$fit

# Save the summary, then display its tables
results <- summary(fit)
results$person_overview # One row summarizing person ability estimates
results$facet_overview  # One row per facet: number of levels, mean, SD, range

# Check the interpretation status and recommended next step
results$decision

# Extract estimates and select the rows to display
estimates <- as.data.frame(fit)
head(subset(estimates, Facet == "Person")) # First six persons
subset(estimates, Facet == "Rater")       # All raters
subset(estimates, Facet == "Criterion")   # All criteria
```

`summary.mfrm_jeffreys_fit` *Inspect a Jeffreys-penalized fit*

Description

These methods inspect the separate result returned by `fit_mfrm()` with `method = "JML", jml_penalty = "jeffreys"`. The result does not inherit ordinary JML uncertainty or FACETS diagnostics.

Usage

```
## S3 method for class 'mfrm_jeffreys_fit'
summary(object, ...)

## S3 method for class 'mfrm_jeffreys_fit'
print(x, ...)

## S3 method for class 'mfrm_jeffreys_fit'
coef(object, ...)

## S3 method for class 'mfrm_jeffreys_fit'
fitted(object, ...)

## S3 method for class 'mfrm_jeffreys_fit'
vcov(object, ...)

## S3 method for class 'mfrm_jeffreys_fit'
confint(object, ...)

## S3 method for class 'mfrm_jeffreys_fit'
logLik(object, ...)
```

Arguments

object, x	A mfrm_jeffreys_fit result.
...	Unused; additional arguments are rejected.

Details

See fit_mfrm() for the penalty, assumptions and restrictions. checks$ordinary_nll is the ordinary negative log likelihood evaluated at the penalized point; checks$objective also includes the information penalty. Neither is an ordinary maximized likelihood for AIC or testing. category_support describes observed category usage, including missing categories; its ordinary-MLE warnings are not penalty-rank diagnostics.

Value

summary() returns Person, facet and step point-estimate tables, checks, category-support evidence and notes. coef() returns the three point-estimate tables. fitted() returns category probabilities in prepared response-row order, with columns from rating_min to rating_max. print() displays the estimator and numerical status. vcov(), confint() and logLik() on this fit stop explicitly: parameter uncertainty and likelihood-based model comparison are not implemented. To estimate common-plan expected-score intervals, use fair_average_table() followed by confint.mfrm_fair_average().

Examples

```
# Long-running workflow; run explicitly when needed.
# example("summary.mfrm_jeffreys_fit", run.dontrun = TRUE)
## Not run:
ratings <- load_mfrmr_data("example_core")
# Each row is one Person's rating by a Rater on a Criterion.
fit <- fit_mfrm(ratings, "Person", c("Rater", "Criterion"), "Score",
  method = "JML", jml_penalty = "jeffreys", rating_min = 1, rating_max = 4,
  category_policy = "preserve")
summary(fit)$persons # Finite penalized Person points, without SEs
summary(fit)$checks$max_gradient # Penalized score-equation diagnostic
plan <- unique(ratings[c("Rater", "Criterion")])
scores <- fair_average_table(fit, reference_plan = plan)
scores$reference_scores # Expected averages under the same plan for all Persons
intervals <- confint(scores, parm = c("P001", "P002"), method = "delta")
intervals[, c("Target", "Estimate", "SE", "Available", "CIEligible")]
# Numerical availability is not a qualification of small-sample coverage.

## End(Not run)
```

---

summary.mfrm_multivariate_d_study

Summarize assessment plans with their uncertainty

Description

Read G, Phi and measurement SEM together with their confidence intervals, methods and limitations. Ordinary `summary(d)` retains its coefficient-table return value. Request intervals explicitly, or reuse saved interval results.

Usage

```
## S3 method for class 'mfrm_multivariate_d_study'
summary(
  object,
  intervals = FALSE,
  parm = c("G", "Phi", "RelativeSEM", "AbsoluteSEM"),
  ...
)

## S3 method for class 'summary.mfrm_multivariate_d_study'
print(x, digits = NULL, row.names = FALSE, ...)
```

Arguments

`object`	A result from `mfrm_d_study()`.
`intervals`	`FALSE` (default) returns the original coefficient table. `TRUE` calculates the selected intervals using `confint.mfrm_multivariate_d_study()`. Alternatively supply a saved `confint(d)` result or a list of such results. Saved results may contain different metrics, methods, scores or reference plans, but must match this D-study's source, plans and score weights. GENOVA-adjusted results retain adjusted points alongside original-source population intervals, with separate `IntervalEstimate` and `IntervalBasis`. Saved intervals must match the component treatment too.
`parm`	Metrics to calculate when `intervals = TRUE`. The default includes `"G"`, `"Phi"`, `"RelativeSEM"` and `"AbsoluteSEM"`. Select fewer metrics when only those answer your question, particularly for explicitly requested expensive profile methods.
`...`	With `intervals = TRUE`, arguments to `confint(d)`, including the required `assumption = "normal"`, `score` or `composite`, `level`, `method`, `reference`, `draws` and `seed`. The default methods and score selection are those of `confint(d)`. Unsupported requests fail explicitly; no alternative method is selected after a failure. With saved results or `intervals = FALSE`, additional arguments are rejected. For printing, arguments are passed to the displayed tables.
`x`	An interval-aware summary returned by this method.
`digits`	Display precision passed to the printed tables; `NULL` uses R's current setting. Stored estimates are unchanged.
`row.names`	Logical; print table row names? Defaults to `FALSE`.

Details

The intervals describe uncertainty from the observed G-study in the population coefficient or SEM of a specified future complete plan. They do not represent data already collected under that future

plan. G and Phi are dimensionless; SEM is in the selected score's units. A larger G/Phi or smaller SEM does not alone determine a worthwhile plan: consider the intended decision, score meaning and task/rater burden.

`reference` requests the existing paired plan-difference procedure for each metric. It does not subtract marginal interval endpoints. Positive G/Phi differences and negative SEM differences favor the comparison plan, conditional on the stated model. Individual-plan and difference results remain explicitly labeled if both are supplied as saved results.

Printing and reopening a saved summary do not calculate intervals or refit models. Printing separates the main estimates, method qualifications and rows with unavailable points/intervals or incomplete width formulas; `table` retains all statuses and diagnostics columns. A completed approximate interval is qualified by its method, not automatically labeled a failure. Save the complete summary with `saveRDS()` to retain its context; CSV export of `table` alone loses weights, notes and diagnostics. Use `plot(s$intervals[[1]])` for its existing forest plot, or supply a suitable single-metric individual-plan result to `plot(d, intervals = ...)`. A summary does not create simultaneous guarantees across metrics, scores or selected plans. Read each method's qualifications. Availability is not proof of model validity or accurate finite-sample coverage.

Value

With `intervals = FALSE`, the unchanged `object$coefficients` data frame. Otherwise a `summary.mfrm_multivariate_d` list containing:

- `table`: one row per returned plan, metric and method, with `Interval` identifying the entry in `intervals`. It retains all original interval columns, including point/interval status, qualifications and any separate REML or residual-contrast `IntervalCenter`. Columns not applicable to a method are `NA`.
- `intervals`: the complete original interval objects, retaining notes, diagnostics, reference context and their existing `plot()` methods.
- `source_design`, `source_rows`, `design_grid`, `weights` and `component_diagnostics`: the original design, score and review context.
- `notes`: method qualifications and interpretation reminders. `print()` returns the same saved summary invisibly. For G/Phi with `method = "delta"`, the table retains and prints the coefficient sampling `SE`, distinct from measurement SEM. For GENOVA-adjusted points it is named `IntervalSE` and describes the original estimator, not the adjusted point. Normal-reference bounds outside `[0,1]` are not clipped.

Choose the operation

`print(d)` displays the D-study and returns it invisibly; `summary(d)` returns its coefficient table. This opt-in summary collects intervals; `confint(d)` calculates one metric at a time with the same inference rules. Neither changes `d`. `print(s)` displays the saved summary without any estimation, and `s$table` is the data frame for subsetting or CSV export. Use `plot(d)` for planning curves and `plot(s$intervals$Phi)` for a saved interval forest. A summary list itself is not a new plotting input.

This page explains the interface and reading order. Mathematical constructions, design-specific assumptions, numerical diagnostics and references are in `confint.mfrm_multivariate_d_study()`.

See Also

mfrm_d_study(), confint.mfrm_multivariate_d_study(), plot.mfrm_multivariate_d_study(), plot.mfrm_dstudy_intervals()

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_multivariate_d_study", run.dontrun = TRUE)
## Not run:
# Published synthetic data, not empirical speech-contest records.
ratings <- read.csv(system.file("extdata", "mgenova-table12.csv", package = "mfrmr"))
g <- mfrm_g_study(ratings, c("V", "W"), rater = NULL)
d <- mfrm_d_study(g, data.frame(Tasks = c(6, 12)),
  weights = c(V = .5, W = .5))
# Compare dependability and measurement error under the normal-effects model.
s <- summary(d, intervals = TRUE, assumption = "normal")
print(s, digits = 3)
# One row per plan for absolute dependability; the complete table is retained.
subset(s$table, Metric == "Phi",
  select = c(Tasks, Estimate, Lower, Upper, Status))
# Reuse an interval for a curve, with no interval calculation.
p <- plot(d, metric = "Phi", intervals = s$intervals$Phi, draw = FALSE)
plot_data(p, "notes")

## End(Not run)
```

---

summary.mfrm_population_prediction

Summarize a population-level design forecast

Description

Summarize a population-level design forecast

Usage

```
## S3 method for class 'mfrm_population_prediction'
summary(object, digits = 3, ...)
```

Arguments

object	Output from predict_mfrm_population().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_population_prediction` with:

- `design`: requested future design
- `overview`: run-level overview
- `forecast`: facet-level forecast table
- `facet_names`: public non-person facet names used in the forecast
- `design_variable_aliases`: public aliases for design variables
- `design_descriptor`: role-based description of design variables
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of mutable/locked design variables
- `planning_schema`: structured planning metadata
- `gpcm_boundary`: `GPCM` caveat row when present
- `structural_design_review`: deterministic structural review of the named-facet design grid; it is not a forecast-uncertainty result
- `ademp`: simulation-study metadata
- `notes`: interpretation notes

See Also

`predict_mfrm_population()`

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_population_prediction", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-population.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
pred <- saved$result
s <- summary(pred)
s$overview
s$forecast[, c("Facet", "MeanSeparation", "McseSeparation")]

## End(Not run)
```

summary.mfrm_signal_detection *Summarize a DIF/bias screening simulation*

Description

Summarize a DIF/bias screening simulation

Usage

```
## S3 method for class 'mfrm_signal_detection'
summary(object, digits = 3, ...)
```

Arguments

object	Output from evaluate_mfrm_signal_detection().
digits	Number of digits used when printing numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_signal_detection with:

- overview: run-level overview
- detection_summary: full-precision rates by design, with design-variable aliases when applicable. DIF*, DIFClass* and BiasScreen* columns retain Planned, Available, Unavailable, Positive, MCLower, MCUpper, AllTrialsLower and AllTrialsUpper for target outcomes. Monte Carlo bounds are exact 95% binomial intervals conditional on availability; all-trial bounds describe unresolved outcomes, not confidence intervals. DIFNonTarget*/BiasNonTarget* retain summed Planned and Available cell counts and AvailableReps, the number of trials contributing a non-target proportion. Those proportions receive equal weight across contributing trials, even with unequal cell counts; their MCSE is the sample SD of the proportions divided by the square root of AvailableReps. TargetContrastAvailable, TargetContrastAbsAvailable, TargetBiasAvailable and TargetBiasTAvailable are denominators for the corresponding finite point summaries. All-missing means return NA.
- ademp: simulation-study metadata carried forward from the original object
- facet_names: public facet labels carried from the simulation specification
- design_variable_aliases: accepted public aliases for design variables
- design_descriptor: role-based design-variable metadata
- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: structured planning metadata
- structural_design_review: deterministic structural review of the named-facet design grid; it reports design bookkeeping rather than signal-detection performance
- gpcm_boundary: GPCM caveat row when present
- notes: short interpretation notes, including the bias-side screening caveat

See Also

evaluate_mfrm_signal_detection(), plot.mfrm_signal_detection

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("summary.mfrm_signal_detection", run.dontrun = TRUE)
## Not run:
# One saved replication illustrates output structure, not precise rates.
# Regeneration recipe: examples/simulation-results.R in the package.
saved <- readRDS(system.file("examples", "simulation-signal-detection.rds",
  package = "mfrmr", mustWork = TRUE))
# Inspect saved$generation_warnings and the result's run/status tables.
sig_eval <- saved$result
summary(sig_eval)

## End(Not run)
```

---

unexpected_after_bias_table

Build an unexpected-after-adjustment screening report

Description

Build an unexpected-after-adjustment screening report

Usage

```
unexpected_after_bias_table(
  fit,
  bias_results,
  diagnostics = NULL,
  abs_z_min = 3,
  prob_max = 0.3,
  top_n = 100,
  rule = c("residual", "either", "both")
)
```

Arguments

fit	Output from fit_mfrm().
bias_results	Output from estimate_bias().
diagnostics	Optional output from diagnose_mfrm() for baseline comparison.
abs_z_min	Absolute standardized-residual cutoff.
prob_max	Maximum observed-category probability cutoff.

top_n	Maximum number of ranked rows to return in table. Summary counts, percentages, and before/after comparisons use all flagged observations, regardless of this display limit.
rule	"residual" (default), "either", or "both"; see unexpected_response_table() for the same baseline/after-bias rule.

Details

This helper recomputes expected values and residuals after interaction adjustments from estimate_bias() have been introduced. Screening coverage is retained separately before and after adjustment. Reduction counts, percentages and the comparison plot are unavailable if either screen leaves responses unclassified. Recreate older saved results with the original settings; the MFRM fit does not need re-estimation.

summary(t10) is supported through summary(). plot(t10) is dispatched through plot() for class mfrm_unexpected_after_bias (type = "scatter", "severity", "comparison").

Value

A named list with:

- table: unexpected responses after bias adjustment
- summary: one-row summary (includes baseline-vs-after counts)
- thresholds: applied thresholds
- facets: analyzed bias facet pair
- gpcm_boundary: GPCM interpretation guidance when applicable

Interpreting output

- summary: before/after unexpected counts and reduction metrics.
- table: residual unexpected responses after bias adjustment.
- thresholds: screening settings used in this comparison.

Lower after-adjustment counts describe an in-sample change in flags; they do not show that bias has been removed or establish fairness. For GPCM, both the bias estimate and the post-adjustment comparison use the fitted slope-aware probability kernel while holding the other fitted quantities fixed. Read the returned gpcm_boundary before reporting the comparison.

Typical workflow

1. Run unexpected_response_table() as baseline.
2. Estimate bias via estimate_bias().
3. Run unexpected_after_bias_table(...) and compare reductions.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and vignette("mfrmr-visual-diagnos
package = "mfrmr").

Output columns

The `table` data.frame has the same structure as `unexpected_response_table()` output, with an additional `BiasAdjustment` column showing the bias correction applied to each observation's expected value.

The `summary` data.frame contains:

TotalObservations Total observations analyzed.

BaselineUnexpectedN Unexpected count before bias adjustment.

AfterBiasUnexpectedN Unexpected count after adjustment.

ReducedBy, ReducedPercent Reduction in unexpected count.

Saved-result reuse

The returned `fit_identity` records the fitted analysis for reuse with reports and exports. Recreate older results from the saved fit and matching diagnostics using the original screening options; no model refit is needed. See mfrmr_compatibility_layer for migration instructions.

See Also

`estimate_bias()`, `unexpected_response_table()`, `bias_count_table()`, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("unexpected_after_bias_table", run.dontrun = TRUE)
## Not run:
# Saved synthetic JML/RSM analysis of example_bias.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "bias-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t10 <- unexpected_after_bias_table(fit, bias, diagnostics = diag, top_n = 20)
summary(t10)
p_t10 <- plot(t10, draw = FALSE)
p_t10$data$plot

## End(Not run)
```

---

unexpected_response_table

Build an unexpected-response screening report

Description

Build an unexpected-response screening report

Usage

```
unexpected_response_table(
  fit,
  diagnostics = NULL,
  abs_z_min = 3,
  prob_max = 0.3,
  top_n = 100,
  rule = c("residual", "either", "both")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
abs_z_min	Absolute standardized-residual cutoff.
prob_max	Maximum observed-category probability cutoff.
top_n	Maximum number of ranked rows to return in table. Summary counts and percentages always use all flagged observations.
rule	"residual" (default) uses only the absolute residual cutoff; "either" and "both" also use the category-probability cutoff.

Details

The default is the FACETS Table 4 absolute residual rule, with cutoff 3. A response is flagged as unexpected when:

- rule = "residual": |StdResidual| >= abs_z_min.
- rule = "either": |StdResidual| >= abs_z_min OR ObsProb <= prob_max
- rule = "both": both conditions must be met.

Missing inputs preserve an unavailable rule outcome unless the other condition determines the result (for example, a true condition suffices for either). Summaries retain evaluated and unavailable counts; the full-sample percentage is withheld if any outcome is unavailable.

The table includes row-level observed/expected values, residuals, observed-category probability, most-likely category, and a composite severity score for sorting.

Value

A named list with:

- table: flagged response rows
- summary: one-row overview
- thresholds: applied thresholds

Interpreting output

- `summary`: prevalence of unexpected responses under current thresholds, before limiting the displayed rows with `top_n`.
- `table`: ranked row-level diagnostics for case review.
- `thresholds`: active cutoffs and flagging rule.

Compare results across `rule = "either"` and `rule = "both"` to assess how conservative your screening should be.

Typical workflow

1. Start with the default residual screen. Category probability is an optional case-review aid, not a tail probability or calibrated false-alarm rate.
2. Re-run with `rule = "both"` for strict subset.
3. Inspect top rows and visualize with plot(x, type = "unexpected").

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `table` data.frame contains:

Row Original row index in the prepared data.

Person Person identifier (plus one column per facet).

Score Observed score category.

Observed, Expected Observed and model-expected score values.

Residual, StdResidual Raw and standardized residuals.

ObsProb Probability of the observed category under the model.

MostLikely, MostLikelyProb Most probable category and its probability. Tied modes give `MostLikely = NA`; `MostLikelyCount` records their number, and CategoryGap uses the nearest tied mode.

Severity Composite severity index (higher = more unexpected).

Direction "Higher than expected" or "Lower than expected".

FlagLowProbability, FlagLargeResidual Logical flags for each criterion.

The `summary` data.frame contains:

TotalObservations Total observations analyzed.

UnexpectedN, UnexpectedPercent Known flagged count and full-sample percentage. The percentage is unavailable when any rule outcome is unknown; the count is unavailable when no response can be evaluated.

EvaluatedObservations, UnavailableObservations Responses whose rule outcome is determined or unavailable.

AbsZThreshold, ProbThreshold Applied cutoff values.

Rule "either" or "both".

Saved-result reuse

The returned `fit_identity` records the fitted analysis for reuse with reports and exports. Recreate older results from the saved fit and matching diagnostics using the original screening options; no model refit is needed. See mfrmr_compatibility_layer for migration instructions.

See Also

`diagnose_mfrm()`, `displacement_table()`, `fair_average_table()`, mfrmr_visual_diagnostics

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("unexpected_response_table", run.dontrun = TRUE)
## Not run:
library(mfrmr)
# Saved synthetic MML/RSM analysis of example_operational.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "operational-mml.rds", package = "mfrmr"))
fit <- example_fit$fit
diagnostics <- example_fit$diagnostics
unexpected <- unexpected_response_table(
  fit, diagnostics = diagnostics, abs_z_min = 1.5, prob_max = 0.4, top_n = 5
)
unexpected$summary # Counts and percentages for all flagged observations
unexpected$table   # Only the five highest-ranked cases
plot(unexpected)
# The rule is exploratory: inspect the scoring context before changing a rating

## End(Not run)
```

---

vcov.mfrm_multivariate_gstudy

Sampling covariance and SEs of G-study component estimates

Description

Quantify how estimated sources of score variation would vary across repeated G-studies with the same assignment design. Supports balanced ANOVA and identifiable MINQUE(0), including incomplete and supported nested sources.

Usage

```
## S3 method for class 'mfrm_multivariate_gstudy'
vcov(object, score = NULL, weights = NULL, assumption, ...)
```

Arguments

`object`	A result from `mfrm_g_study()`.
`score`	One original score, or exactly two distinct original scores for the sampling covariance of their between-score component covariances. The default selects the first original score.
`weights`	Optional named weights for one prespecified composite; name every original score. Do not also supply `score`.
`assumption`	Required: `"normal"` asserts independent normal random effects, a common mean for each score and constant component covariances. Assignment and omissions must preserve that model when conditioning on the observed assignment. The assumption is not tested.
`...`	Reserved; extra arguments are rejected.

Details

Rows and columns refer to **estimated components**, such as Person, Task and Residual. This is neither the covariance matrix between individual ratings nor the component matrices between score columns in `object`. Use `attr(result, "summary")` to read the original estimates, sampling variance estimates, SEs and availability. A component SE has squared score units; the returned covariance matrix has fourth-power score units. For a between-score covariance, the SE has the product of the two score units. These SEs differ from the D-study's measurement SEM in original score units.

The component estimates are unchanged. This method uses Gaussian product moments to correct plug-in bias in their estimated sampling covariance. It does not fit a likelihood, resample data or construct confidence intervals. In particular, `Estimate +/- 1.96 * SE` is not a generally justified interval. Use `confint.mfrm_multivariate_gstudy()` for its separately qualified interval methods. A REML profile's center belongs to a different estimator; the SE returned here describes the original ANOVA/MINQUE estimator.

Value

A named matrix estimating the sampling covariance of component estimates. Attribute `summary` is a data frame with `Source`, `Estimate`, `SamplingVariance`, `SE`, and `SEStatus`. `diagnostics` retains the moment rank, singular values, entrywise estimability and matrix PSD check. `score`, `weights`, `assumption`, `source_design`, `source_rows` and `notes` retain interpretation. Save the full result using `saveRDS()`. MINQUE requires its retained grouping assignments, moment equations and quadratic products; balanced ANOVA uses saved summaries without raw data.

Calculation

With mean-removing MINQUE quadratic forms $q_s = y^T A_s y$, $E(q_s) = \theta_s$ and $C_{st} = 2\mathrm{tr}(A_s V A_t V)$, where $V = \sum_u \theta_u K_u$. Let the linear operator P map $\mathrm{vec}(\theta\theta^T)$ to $\mathrm{vec}(C)$. The normal product-moment identity gives $E\{\mathrm{vec}(qq^T)\} = (I + P)\mathrm{vec}(\theta\theta^T)$. Inverting this moment system gives an unbiased estimator of C for its identifiable entries. This is a consequence of the cited moment identity, not a claim that the reference proposes this package's G-theory procedure. The squared SE, when available, is the corresponding diagonal estimate; the square root is not itself an unbiased SE estimator.

For two scores, write x and y for their variance-component vectors and c for their covariance-component vector. On symmetric matrix coordinates, $E\{\mathrm{sym}(\hat{x}\hat{y}^T)\} = \mathrm{sym}(xy^T) + P(cc^T)$ and $E(\hat{c}\hat{c}^T) = (I + P/2)(cc^T) + (P/2)\mathrm{sym}(xy^T)$. Solve these coupled equations for the sampling covariance $(P/2)\{\mathrm{sym}(xy^T) + cc^T\}$. Applying only the scalar correction to a covariance component would be incorrect.

Balanced ANOVA reduces to the existing independent-mean-square result `A %*% diag(2 * MS^2 / (df + 2)) %*% t(A)` for a variance/fixed composite, and the Wishart product-moment correction for a between-score covariance. MINQUE uses the actual observed grouping kernels. Column-block calculations avoid allocating observation-square matrices, but time still increases quadratically with the number of observed ratings. No sampling simulation is performed. Balanced MINQUE uses the same ANOVA limit.

Unavailable or negative results

A moment system can fail to identify some sampling-covariance entries even when point components are identifiable. A contributing balanced stratum with df one is an example for between-score covariance SEs. An SVD checks each target entry's estimability; only unidentified entries become `NA`. Its rank tolerance is `sqrt(.Machine$double.eps)` relative to the largest singular value, with relative target reconstruction error below `1e-8`.

An unbiased sampling-variance estimate may be negative. It is retained in the returned matrix and `SamplingVariance`, while its `SE` is `NA` and `SEStatus` explains why. The matrix can also be indefinite with positive diagonal entries. Nothing is clipped or repaired to make it positive semidefinite. A mathematically unbiased variance estimate does not ensure a useful SE in a small sample or correct normal-reference coverage.

References

Magnus, J. R. (1978). The moments of products of quadratic forms in normal variables. *Statistica Neerlandica, 32*, 201–210. Lemma 6.2, p.209. doi:10.1111/j.14679574.1978.tb01399.x.

Rao, C. R. (1971). Estimation of variance and covariance components–MINQUE theory. *Journal of Multivariate Analysis, 1*, 257–275. doi:10.1016/0047259X(71)900017.

Examples

```
# Supplementary workflow: run explicitly to keep routine checks brief.
# example("vcov.mfrm_multivariate_gstudy", run.dontrun = TRUE)
## Not run:
data("mfrmr_example_gtheory_crossed", package = "mfrmr")
ratings <- mfrmr_example_gtheory_crossed
# Complete balanced ratings: compute both targets directly.
g <- mfrm_g_study(ratings, c("Content", "Organization"),
                           method = "anova")
sampling <- vcov(g, score = "Content", assumption = "normal")
attr(sampling, "summary") # Component estimates and their sampling SEs.
attr(sampling, "notes")
# Uncertainty in the covariance between two criteria, by source.
cross_score <- vcov(g, score = c("Content", "Organization"),
                    assumption = "normal")
attr(cross_score, "summary")
```

```
# Incomplete assignments also work, but their moment calculation is longer.
# These saved results use the same ratings with rows 1, 8 and 23 omitted.
# Omitted ratings are absent observations, never zero scores.
incomplete <- readRDS(system.file("examples", "gstudy-sampling.rds",
                                  package = "mfrmr", mustWork = TRUE))
incomplete$study$design
attr(incomplete$variance, "summary")
attr(incomplete$covariance, "summary")
attr(incomplete$covariance, "notes")
# To repeat the full incomplete-data calculation:
# source(system.file("examples", "gstudy-sampling.R", package = "mfrmr"))
# incomplete <- gstudy_sampling_example()

## End(Not run)
```

`visual_reporting_template` *Figure-reporting template for visual diagnostics*

Description

Return a compact, beginner-oriented template that explains where each visual family normally belongs in a report, which helper to call, what to say, and what not to claim. Use this static table together with the dynamic `reporting_checklist(fit, diagnostics)$visual_scope` table: the template answers "how should I use this figure?", while the checklist answers "is this figure ready for the current run?".

Usage

```
visual_reporting_template(
  scope = c("all", "manuscript", "appendix", "diagnostic", "surface")
)
```

Arguments

`scope`	Which part of the template to return: `"all"` (default), `"manuscript"`, `"appendix"`, `"diagnostic"`, or `"surface"`.

Details

This helper is intentionally conservative. It does not inspect a fitted object and does not certify that a plot is available. Run `reporting_checklist()` for run-specific readiness, then use this table to decide how to describe the resulting figure.

Value

A data.frame with columns:

- `FigureFamily`: short visual family label.
- `Scope`: broad reporting role used for filtering.
- `PrimaryHelper`: public helper or plot route.
- `DefaultPlacement`: recommended location in a report.
- `WhatToReport`: wording focus for results sections or captions.
- `CaptionSkeleton`: caption starter that must be tailored to the study.
- `ResultsWording`: results-sentence starter that must be checked against the fitted object and diagnostics.
- `WhatNotToClaim`: common overclaim to avoid.
- `BeginnerCheck`: first thing a new user should inspect.
- `ThreeDPolicy`: whether 3D is recommended, discouraged, or data-only.

Examples

```
visual_reporting_template()
visual_reporting_template("manuscript")
visual_reporting_template("surface")
mfrmr_interval_guide("visual")[, c("Route", "PrimaryHelper", "DefaultLevel")]
```

---

write_mfrm_residual_file

Write a standalone residual file

Description

Write a standalone residual file

Usage

```
write_mfrm_residual_file(
  fit,
  diagnostics = NULL,
  path,
  format = c("csv", "tsv"),
  digits = 4,
  overwrite = FALSE,
  include_probabilities = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). Supplying it avoids recomputing observation diagnostics. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
path	Output file path.
format	File format: "csv" or "tsv". If omitted, inferred from path when the extension is .csv or .tsv, otherwise "csv".
digits	Rounding digits for numeric columns. ConditionalVariance retains full precision so small positive variances are not exported as zero.
overwrite	If FALSE, existing files are not overwritten.
include_probabilities	If TRUE, append model probabilities for all response categories as PrCategory_* columns.

Details

The exported table is observation-level and model-native. It includes the observed score, expected score, residual, standardized residual, variance, score information, observed-category probability, and modeled person measure when those quantities are available. Current diagnostics also retain the primary Person estimate and diagnostic basis, the unthresholded ConditionalVariance, PearsonAvailable, ResponseStatus and ResponseReason; see diagnose_mfrm() for their interpretation and the legacy small-variance cutoff.

This writer is separate from facets_output_file_bundle() because it is a direct analysis handoff rather than a legacy graph/score bundle.

Value

A bundle with table, summary, written_files, and settings.

See Also

diagnose_mfrm(), facets_output_file_bundle(), write_mfrm_subset_file()

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
path <- tempfile(fileext = ".csv")
out <- write_mfrm_residual_file(fit, diag, path, overwrite = TRUE)
data.frame(
  Component = out$written_files$Component,
  File = basename(out$written_files$Path)
)
# Full paths remain in out$written_files$Path.
```

write_mfrm_subset_file *Write standalone subset-connectivity files*

Description

Write standalone subset-connectivity files

Usage

```
write_mfrm_subset_file(
  fit,
  diagnostics = NULL,
  path,
  node_path = NULL,
  format = c("csv", "tsv"),
  digits = 4,
  overwrite = FALSE,
  include_nodes = TRUE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). Supplying it avoids recomputing subset connectivity. Must match this fitted analysis; see mfrmr_compatibility_layer for migration of saved diagnostics.
path	Output file path for the subset summary table.
node_path	Optional output file path for the node-level subset table. When NULL and include_nodes = TRUE, a sibling file ending in _nodes.csv or _nodes.tsv is created.
format	File format: "csv" or "tsv". If omitted, inferred from path when the extension is .csv or .tsv, otherwise "csv".
digits	Rounding digits for numeric columns.
overwrite	If FALSE, existing files are not overwritten.
include_nodes	If TRUE, also write the node-level facet/level to subset membership table.

Details

Subsets are connected components in the observation design graph. The graph links Person and all modeled facet levels that co-occur in an observation. A single co-observation graph component does not prove common-scale identification; shared other facets can conceal confounded panels. Consult the fitted estimability audit. Multiple graph subsets require external anchoring or a deliberate separate-calibration design justifies it.

Value

A bundle with `table`, `nodes`, `summary`, `written_files`, and `settings`.

See Also

`diagnose_mfrm()`, `subset_connectivity_report()`, `write_mfrm_residual_file()`

Examples

```
# Saved synthetic JML/RSM analysis of example_core.
# For your own data, fit a model first; see ?fit_mfrm.
example_fit <- readRDS(system.file("examples", "core-jml.rds", package = "mfrmr"))
fit <- example_fit$fit
diag <- example_fit$diagnostics
path <- tempfile(fileext = ".csv")
out <- write_mfrm_subset_file(fit, diag, path, overwrite = TRUE)
data.frame(
  Component = out$written_files$Component,
  File = basename(out$written_files$Path)
)
# Full paths remain in out$written_files$Path.
```

Index

∗ **GPCM boundaries**
gpcm_capability_matrix, 410
mfrmr_output_guide, 486
mfrmr_visual_diagnostics, 503
∗ **ICC**
compute_facet_icc, 160
∗ **calculation methods**
mfrmr_methods_reference, 474
∗ **confidence intervals**
analyze_facet_equivalence, 28
analyze_hierarchical_structure, 31
compute_facet_icc, 160
fit_measures_table, 347
mfrmr_interval_guide, 461
mfrmr_visual_diagnostics, 503
plot.mfrm_fit, 700
∗ **facet equivalence**
analyze_facet_equivalence, 28
∗ **figure captions**
visual_reporting_template, 880
∗ **fit statistics**
fit_measures_table, 347
∗ **hierarchical structure**
analyze_hierarchical_structure, 31
compute_facet_icc, 160
∗ **methodological references**
mfrmr_methods_reference, 474
∗ **quality control**
response_time_review, 795
∗ **reporting workflow**
analyze_facet_equivalence, 28
analyze_hierarchical_structure, 31
mfrmr_interval_guide, 461
mfrmr_output_guide, 486
mfrmr_visual_diagnostics, 503
visual_reporting_template, 880
∗ **response time**
response_time_review, 795
∗ **route selection**
gpcm_capability_matrix, 410
mfrmr_output_guide, 486
∗ **shrinkage**
plot.mfrm_fit, 700
∗ **uncertainty displays**
mfrmr_interval_guide, 461
∗ **visual diagnostics**
mfrmr_interval_guide, 461
mfrmr_visual_diagnostics, 503
plot.mfrm_fit, 700
response_time_review, 795
visual_reporting_template, 880

AIC.mfrm_cml_fit *(*mfrm_fit_extractors*)*, 588
AIC.mfrm_fit *(*mfrm_fit_extractors*)*, 588
analyze_dff, 21
analyze_dff(), *11*, *13*, *22, 23*, *157*, *233*, *253, 254*, *256–258*, *294, 295*, *297*, *442*, *454*, *458*, *466*, *468–470*, *472, 473*, *477*, *507*, *511, 512*, *515*, *523*, *561, 562*, *677*, *692, 693*
analyze_dif *(*analyze_dff*)*, 21
analyze_dif(), *11*, *256–258*, *297*, *458*, *511*, *523*, *561, 562*, *692*
analyze_facet_equivalence, 28
analyze_facet_equivalence(), *11*, *29*, *523*, *585*, *707*
analyze_hierarchical_structure, 31
analyze_hierarchical_structure(), *159*, *163*, *243*, *339*, *374*, *455*, *524*, *700*, *791*
analyze_residual_pca, 35
analyze_residual_pca(), *11*, *13*, *36*, *39, 40*, *251, 252*, *477*, *515*, *523*, *527*, *753*
anchor_review *(*review_accessors*)*, 797
anchor_to_baseline, 41
anchor_to_baseline(), *92*, *95*, *239, 240*, *465*, *469, 470*, *472, 473*
apa_table, 45

apa_table(), *11*, *46–49*, *59*, *60*, *65*, *66*, *127*, *128*, *131*, *187*, *314*, *386*, *416*, *492–494*, *496*, *497*, *499*, *501*, *502*, *528*, *529*, *536*, *543*, *544*, *577*, *584*, *622*, *647*, *652*, *667*, *722*, *749*, *779*, *828*
apply_empirical_bayes_shrinkage, 50
apply_empirical_bayes_shrinkage(), *108*, *358*, *453*, *483*, *508*, *511*, *524*, *833*
as.data.frame(), *774*
as.data.frame.mfrm_facets_results *(*read_facets_results*)*, 772
as.data.frame.mfrm_fit, 52, *311*
as_flextable, 58
as_flextable(), *8*, *65*
as_flextable.apa_table, 59
as_flextable.apa_table(), *59*, *66*
as_ggplot, 60
as_ggplot(), *8*, *37*, *72*, *95*, *139*, *143*, *152*, *267*, *269*, *273*, *338*, *402*, *416*, *429*, *454*, *485*, *489*, *492*, *494*, *504*, *513*, *521–524*, *536*, *550*, *551*, *577*, *584*, *585*, *610*, *616*, *621*, *622*, *637*, *642*, *643*, *652*, *659*, *662*, *683*, *684*, *687*, *688*, *695*, *696*, *698*, *700*, *707*, *708*, *710*, *713*, *714*, *718*, *722*, *728*, *729*, *735*, *736*, *753*, *760*, *767*, *780*, *786*, *832*
as_kable, 65
as_kable(), *8*, *59*
as_kable.apa_table, 66
as_kable.apa_table(), *60*, *65*
assess_mfrm_recovery, 54
assess_mfrm_recovery(), *11*, *129*, *313*, *379*, *500*, *502*, *528–530*, *839*, *842*, *844*, *845*
automated_scoring_table, 67
automated_scoring_table(), *826*

base::readRDS(), *8*, *429*, *489*, *632*, *756*, *825*
base::saveRDS(), *8*, *37*, *72*, *146*, *406*, *419*, *429*, *433*, *442*, *479*, *489*, *494*, *632*, *677*, *679*, *681*, *753*, *756*, *759*, *825*
bias_count_table, 68
bias_count_table(), *874*
bias_interaction_report, 70
bias_interaction_report(), *11*, *70*, *73*, *74*, *78*, *98*, *273*, *276*, *278*, *447*, *477*
bias_iteration_report, 73
bias_iteration_report(), *70*
bias_pairwise_report, 74
bias_pairwise_report(), *70*, *77*, *97*
BIC.mfrm_cml_fit *(*mfrm_fit_extractors*)*, 588
BIC.mfrm_fit *(*mfrm_fit_extractors*)*, 588
bootstrap_mfrm_gpcm, 78
bootstrap_mfrm_gpcm(), *183*, *379*, *410*, *415*, *529*, *721*
build_apa_outputs, 81
build_apa_outputs(), *10*, *11*, *13*, *49*, *82*, *84*, *85*, *98*, *129*, *131*, *134*, *258*, *278*, *299*, *301*, *338*, *391*, *446*, *447*, *456*, *492–494*, *496*, *497*, *499–502*, *525*, *529*, *605*, *632*, *634*, *793*, *794*, *813*
build_conquest_overlap_bundle, 86
build_conquest_overlap_bundle(), *11*, *668*, *671*, *674*, *799–801*
build_equating_chain, 90
build_equating_chain(), *11*, *44*, *45*, *99*, *100*, *237*, *239*, *240*, *454*, *470–472*, *512*, *526*, *528*
build_fixed_reports, 96
build_fixed_reports(), *70*, *72*, *74*, *77*, *78*, *97*, *276*, *278*, *327*, *446*, *447*, *456*, *528*, *790*
build_linking_review, 99
build_linking_review(), *11*, *99*, *100*, *129*, *469*, *470*, *472*, *473*, *515*, *526*, *528*, *627*, *794*
build_mfrm_manifest, 101
build_mfrm_manifest(), *10*, *11*, *33*, *34*, *109*, *110*, *301*, *302*, *353*, *492*, *529*
build_mfrm_network_review, 104
build_mfrm_network_review(), *105*, *107*, *125*, *507*, *512*, *515*, *525*, *809*
build_mfrm_replay_script, 107
build_mfrm_replay_script(), *10*, *11*, *51*, *89*, *102*, *104*, *301*, *302*, *468*, *492*, *529*
build_mfrm_resampling_spec, 110
build_mfrm_resampling_spec(), *261*
build_mfrm_sim_spec, 113
build_mfrm_sim_spec(), *10*, *11*, *13*, *112*, *113*, *124*, *125*, *282*, *289*, *293*, *317*, *379*, *470*, *522*, *528–530*, *737*, *740*, *835–838*, *841*, *844*
build_misfit_casebook, 119
build_misfit_casebook(), *11*, *120*, *121*,

129, *454*, *526*, *528*, *605, 606*, *793, 794*
build_model_choice_review, 121
build_model_choice_review(), *11*, *122, 123*, *129*, *131*, *493–495*, *497*, *515*, *523*, *627*
build_peer_review_design_review *(*review_peer_design*)*, 808
build_peer_review_sim_spec, 123
build_peer_review_sim_spec(), *116*, *809*
build_summary_table_bundle, 125
build_summary_table_bundle(), *46*, *107*, *126, 127*, *131*, *299*, *301*, *312–314*, *493–497*, *499–502*, *515*, *525, 526*, *528–530*, *536*, *543*, *637*, *652*, *655*, *749*, *779*, *809*, *828*
build_visual_summaries, 132
build_visual_summaries(), *10, 11*, *13*, *85*, *252*, *301*, *492*, *496, 497*, *509*, *528, 529*, *663, 664*, *794*, *818*
build_weighting_review, 135
build_weighting_review(), *11*, *122, 123*, *129*, *131*, *136, 137*, *187*, *370*, *379*, *495*, *523*, *528, 529*

category_curves_report, 138
category_curves_report(), *13*, *143*, *329, 330*, *379*, *447*, *476*, *500–502*, *528, 529*, *560*
category_structure_report, 142
category_structure_report(), *13*, *141*, *329*, *365*, *379*, *447*, *476*, *500–502*, *528*
classification_table, 144
classification_table(), *7*, *9*, *477*, *488*, *749*
cluster::daisy(), *546*
cluster::pam(), *546*
cluster::silhouette(), *546*, *551*
coef(), *180*
coef.mfrm_cml_fit *(*mfrm_fit_extractors*)*, 588
coef.mfrm_comparison_fit *(*mfrm_fit_extractors*)*, 588
coef.mfrm_fit *(*mfrm_fit_extractors*)*, 588
coef.mfrm_jeffreys_fit *(*summary.mfrm_jeffreys_fit*)*, 865
coef.mfrm_network_ergm *(*fit_rater_network*)*, 407
coefficient, covariance, likelihood and count methods, *450*
compare_mfrm, 147
compare_mfrm(), *7*, *11*, *27*, *79*, *81*, *122, 123*, *131*, *136, 137*, *149*, *183*, *268, 269*, *355*, *370*, *398*, *403*, *416*, *424*, *427*, *450*, *493–495*, *497*, *515*, *523*, *529*, *589*, *591*, *642*, *649*, *745*, *827*
compatibility_alias_table, 156
compatibility_alias_table(), *449*, *518*
compute_facet_design_effect, 158
compute_facet_design_effect(), *33, 34*, *162, 163*, *243*, *339*, *374*, *524*
compute_facet_icc, 160
compute_facet_icc(), *18*, *32–34*, *52*, *158, 159*, *163*, *243*, *339*, *374*, *524*, *597*, *679*
compute_information, 164
compute_information(), *11*, *13*, *123*, *136, 137*, *164, 165*, *168*, *379*, *417*, *503*, *511*, *528–530*, *559, 560*, *864*
compute_person_fit_indices, 169
compute_person_fit_indices(), *128*, *169, 170*, *523*
confint(), *590*
confint.mfrm_cml_fit *(*confint.mfrm_fit*)*, 180
confint.mfrm_comparison_fit *(*confint.mfrm_fit*)*, 180
confint.mfrm_facet_intervals *(*mfrm_facet_intervals*)*, 572
confint.mfrm_fair_average, 172
confint.mfrm_fair_average(), *174*, *176*, *342*, *375*, *866*
confint.mfrm_fit, 180
confint.mfrm_fit(), *7*, *79*, *81*, *150*, *248*, *360*, *370*, *415*, *417*, *450*, *462*, *468*, *529*, *559, 560*, *583*, *590*, *633*, *721*
confint.mfrm_gpcm_bootstrap *(*bootstrap_mfrm_gpcm*)*, 78
confint.mfrm_jeffreys_fit *(*summary.mfrm_jeffreys_fit*)*, 865
confint.mfrm_multivariate_d_study, 188
confint.mfrm_multivariate_d_study(), *215, 216*, *481*, *489*, *564*, *567*, *607*,

608, *694*, *696*, *710*, *712, 713*, *867–869*
confint.mfrm_multivariate_gstudy, 212
confint.mfrm_multivariate_gstudy(), *192*, *200*, *210*, *320*, *480, 481*, *489*, *601*, *716, 717*, *878*
confint.mfrm_network_ergm *(*fit_rater_network*)*, 407
confint.mfrm_posterior_summary *(*read_mfrm_posterior*)*, 777
confint.mfrm_random_rater, 221
confint.mfrm_random_rater(), *224*, *397, 398*, *462*, *464*
confint.mfrm_random_rater_intervals *(*mfrm_random_rater_intervals*)*, 623
confint.mfrm_testlet, 223
confint.mfrm_testlet(), *222*, *402*, *462*

data_quality_report, 224
data_quality_report(), *12*, *280*, *333, 334*, *500*, *502*, *794*, *846*, *848*
describe_mfrm_data, 226
describe_mfrm_data(), *8, 9*, *11*, *23*, *129*, *226*, *228–231*, *235*, *285*, *351*, *360*, *362*, *367*, *426, 427*, *444*, *451*, *459*, *466*, *469, 470*, *475, 476*, *483*, *487*, *500*, *515, 516*, *519*, *527*, *574*, *600*, *781, 782*, *804*, *807*, *860*, *864*
detect_anchor_drift, 236
detect_anchor_drift(), *11*, *44, 45*, *92*, *95*, *99, 100*, *237*, *239, 240*, *454*, *465*, *469–473*, *526*, *528*
detect_facet_nesting, 241
detect_facet_nesting(), *33, 34*, *163*, *339*, *374*, *455*, *700*
diagnose_mfrm, 244, *310, 311*
diagnose_mfrm(), *11*, *13*, *22*, *27, 28*, *35, 36*, *39, 40*, *43–46*, *49*, *71*, *73*, *75*, *82*, *85*, *101*, *103*, *105*, *108, 109*, *119–121*, *128*, *133*, *142, 143*, *155*, *169*, *171*, *184*, *238*, *245*, *251–253*, *259, 260*, *263*, *265*, *268*, *270*, *282, 283*, *289, 290*, *293*, *299*, *317, 318*, *322, 323*, *325*, *327, 328*, *330*, *333*, *335*, *337*, *340, 341*, *344*, *346–348*, *350*, *358*, *365*, *367*, *379*, *391*, *417*, *428*, *431*, *443, 444*, *446, 447*, *450*, *453*, *456*, *469*, *473*, *476, 477*, *482, 483*, *493*, *497*, *502, 503*, *511, 512*, *515*, *521*, *523*, *527*, *529*, *531*, *592, 593*, *597*, *611*, *613*, *633*, *636*, *641–643*, *648, 649*, *655*, *665*, *667*, *702*, *730, 731*, *738*, *752*, *754–757*, *762*, *765*, *767*, *769*, *771*, *789–791*, *793, 794*, *812, 813*, *815*, *818*, *824*, *838*, *841*, *847, 848*, *854*, *856*, *860*, *864*, *872*, *875*, *877*, *882–884*
dif_interaction_table, 253
dif_interaction_table(), *11*, *27*, *254*, *256–258*, *458*, *523*, *693*
dif_report, 256
dif_report(), *11*, *26, 27*, *256*, *473*
displacement_table, 258
displacement_table(), *119–121*, *346*, *379*, *454*, *527*, *877*
draw_mfrm_resamples, 261
draw_mfrm_resamples(), *112, 113*

ej2021_combined *(*ej2021_data*)*, 262
ej2021_combined_itercal *(*ej2021_data*)*, 262
ej2021_data, 262, *436*, *438*
ej2021_study1 *(*ej2021_data*)*, 262
ej2021_study1_itercal *(*ej2021_data*)*, 262
ej2021_study2 *(*ej2021_data*)*, 262
ej2021_study2_itercal *(*ej2021_data*)*, 262
eRm::PCM(), *420*
eRm::RM(), *420*
eRm::RSM(), *420*
estimate_all_bias, 264
estimate_all_bias(), *11*, *265*, *269*, *275*, *378*, *426*, *454*, *477*, *488*, *627*
estimate_bias, 270
estimate_bias(), *10, 11*, *13*, *27*, *47*, *68–75*, *78*, *82*, *85*, *96–98*, *101*, *108*, *129*, *256*, *265–269*, *271*, *273*, *276*, *278*, *294, 295*, *297*, *299*, *325*, *333–335*, *378, 379*, *391*, *425–427*, *453, 454*, *458*, *477*, *488*, *506*, *511, 512*, *515*, *528, 529*, *531*, *577*, *627*, *789*, *791*, *815*, *818*, *872–874*
estimation_iteration_report, 279
estimation_iteration_report(), *500*, *812, 813*, *846*
evaluate_mfrm_design, 280
evaluate_mfrm_design(), *11*, *13*, *129*, *290*, *296, 297*, *522*, *528, 529*, *571*, *689,*

690, *738–740*, *783*, *786*, *838*, *842*, *845*, *851–853*
evaluate_mfrm_diagnostic_screening, 287
evaluate_mfrm_diagnostic_screening(), *690*, *691*, *856*, *857*
evaluate_mfrm_recovery *(*simulate_mfrm_recovery*)*, 839
evaluate_mfrm_signal_detection, 291
evaluate_mfrm_signal_detection(), *11*, *129*, *294*, *297*, *528*, *659*, *871*, *872*
export_mfrm *(*export_mfrm_tables*)*, 309
export_mfrm_bundle, 298
export_mfrm_bundle(), *10*, *11*, *89*, *102–104*, *108–110*, *308–311*, *313*, *314*, *329*, *330*, *447*, *448*, *492*, *495*, *497*, *499*, *500*, *502*, *529*, *651*
export_mfrm_model, 303
export_mfrm_model(), *372*, *463*, *488*, *500*, *780*
export_mfrm_results, 306
export_mfrm_results(), *108*, *300*, *302*, *306*, *310*, *311*, *386*, *412*, *434*, *448*, *478*, *480*, *493*, *494*, *499*, *517*, *521*, *525*, *528*, *584*, *585*, *622*, *634*, *651*, *652*, *654*, *655*, *722*
export_mfrm_tables, *53*, 309
export_mfrm_tables(), *102*, *300–302*, *308*, *448*, *499*
export_summary_appendix, 312
export_summary_appendix(), *11*, *268*, *302*, *309*, *345*, *409*, *493–497*, *499–502*, *526*, *528–530*, *536*, *543*, *544*, *631*, *749*, *779*, *786*, *828*
extract_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
extract_mfrm_calibration(), *43*, *369*, *370*, *382*, *413*, *439*, *465*, *537*, *653*, *745*, *748*
extract_mfrm_sim_spec, 315
extract_mfrm_sim_spec(), *10*, *11*, *13*, *118*, *282*, *289*, *293*, *379*, *528–530*, *737*, *738*, *740*, *836–838*, *841*

facet_quality_dashboard, 332
facet_quality_dashboard(), *11*, *268*, *269*, *299*, *301*, *334*, *335*, *379*, *527*, *605*, *606*, *659*
facet_small_sample_review, 336
facet_small_sample_review(), *34*, *52*, *163*, *243*, *337*, *339*, *374*, *456*
facet_statistics_report, 339
facet_statistics_report(), *493*, *496*, *497*, *500*, *502*, *731*
facets_chisq_table, 317
facets_chisq_table(), *31*, *431*, *500*
facets_feature_coverage, 319
facets_feature_coverage(), *331*, *332*, *456*
facets_fit_df_guide, 322
facets_fit_df_guide(), *17*, *321*
facets_fit_review, 323
facets_fit_review(), *128*, *247*, *321*, *322*, *331*, *350*, *771*, *793*
facets_output_contract_review, 325
facets_output_contract_review(), *325*, *447*, *456*, *790*
facets_output_file_bundle, 327
facets_output_file_bundle(), *13*, *379*, *446*, *447*, *456*, *882*
facets_positioning_guide, 330
facets_positioning_guide(), *321*, *332*, *446*, *449*, *456*
facets_term_crosswalk, 331
facets_term_crosswalk(), *332*
facets_visual_contract, 332
facets_visual_contract(), *332*
fair_average_table, 341
fair_average_table(), *10*, *13*, *31*, *157*, *173*, *260*, *374*, *379*, *528*, *529*, *704*, *812*, *813*, *851*, *866*, *877*
fit_measures_table, 347
fit_measures_table(), *128*, *322*, *464*, *476*, *519*, *605*, *649*, *662*, *683*, *706*, *793*
fit_mfrm, *53*, *310*, *311*, 350
fit_mfrm(), *7–9*, *11–14*, *16*, *20*, *22*, *27*, *28*, *31*, *34–36*, *43–45*, *49*, *51*, *52*, *68*, *71*, *73*, *75*, *82*, *85*, *86*, *101*, *103*, *105*, *108*, *109*, *119*, *120*, *122*, *128*, *133*, *136*, *139*, *141–143*, *147*, *155*, *164*, *166–169*, *181*, *186*, *224*, *226–228*, *233–235*, *243*, *244*, *252*, *253*, *255*, *259*, *263*, *265*, *268*, *270*, *271*, *279*, *280*, *282–284*, *289*, *293*, *299*, *302–304*, *306*, *315*, *317*, *323*, *325*, *327*, *328*, *333*, *337*, *340*, *341*, *347*, *373*, *392*, *405*, *407*, *411*, *412*, *417*,

425–428, *437–439*, *443*, *446*, *447*, *451*, *456*, *465*, *467*, *469*, *473*, *475*, *476*, *482*, *483*, *487*, *493*, *495*, *497*, *502*, *512*, *515–518*, *520*, *521*, *526*, *527*, *529*, *531*, *532*, *538*, *541*, *573*, *580*, *582*, *589*, *591*, *592*, *594*, *611*, *632*, *636*, *639*, *647*, *651*, *652*, *655*, *656*, *665*, *667*, *701*, *707*, *708*, *730*, *737*, *738*, *741*, *749*, *752*, *755*, *757*, *762*, *765*, *767*, *781*, *782*, *787*, *789*, *791*, *794*, *795*, *797*, *803*, *805–807*, *811–813*, *815*, *819*, *824*, *833*, *837*, *838*, *841*, *845–847*, *860*, *863–866*, *872*, *875*, *882*, *883*
fit_mfrm_imputed, 391
fit_mfrm_imputed(), *447*, *517*, *522*, *644*, *646*, *726*, *729*
fit_mfrm_random_rater, 393
fit_mfrm_random_rater(), *8*, *148*, *221*, *394*, *399*, *403*, *414*, *516*, *520*, *624*, *626*, *639*, *647*, *654*, *732*, *830*
fit_mfrm_testlet, 399
fit_mfrm_testlet(), *148*, *223*, *393*, *397*, *400*, *414*, *464*, *480*, *516*, *520*, *639*, *647*, *654*, *734*, *736*
fit_rater_accuracy, 405
fit_rater_accuracy(), *447*, *477*, *488*
fit_rater_network, 407
fit_rater_network(), *408*, *627*, *760*, *764*
fitted rating and residual methods, *450*
fitted(), *245*
fitted.mfrm_cml_fit *(*mfrm_fit_responses*)*, 591
fitted.mfrm_comparison_fit *(*mfrm_fit_responses*)*, 591
fitted.mfrm_fit *(*mfrm_fit_responses*)*, 591
fitted.mfrm_jeffreys_fit *(*summary.mfrm_jeffreys_fit*)*, 865
freeze_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
freeze_mfrm_calibration(), *537*

gpcm_capability_matrix, *9*, *391*, 410, *498*, *513*, *531*
gpcm_capability_matrix(), *7*, *10*, *13*, *122*, *123*, *137*, *187*, *321*, *326*, *329*, *370*, *379*, *386*, *398*, *403*, *463*, *464*, *476*, *482*, *484*, *488*, *492*, *500*, *504*, *520*, *521*, *528*
gpcm_mml_quadrature_sensitivity *(*mml_quadrature_sensitivity*)*, 664
gpcm_mml_quadrature_sensitivity(), *665*
gpcm_runtime_guard_coverage *(*gpcm_capability_matrix*)*, 410
gpcm_score_side_contract *(*gpcm_capability_matrix*)*, 410
graphics::plot(), *408*
grDevices::cairo_pdf(), *776*
grDevices::png(), *776*
gwet_ac2_table, 418
gwet_ac2_table(), *431*, *445*, *477*, *479*, *487*, *681*

import_erm_fit, 420
import_erm_fit(), *376*, *423*, *425*, *525*
import_facets_fit_table *(*read_facets_fit_table*)*, 770
import_mirt_fit, 421
import_mirt_fit(), *420*, *421*, *425*, *525*
import_tam_fit, 423
import_tam_fit(), *421*, *423*, *525*
interaction_effect_table, 425
interaction_effect_table(), *233*, *278*, *466*, *860*
interactive(), *282*
interrater_agreement_table, 427
interrater_agreement_table(), *7*, *83*, *318*, *379*, *433*, *447*, *479*, *480*, *512*, *525*, *527*, *632*, *679*, *756*, *761–765*, *826*

krippendorff_alpha_table, 431
krippendorff_alpha_table(), *419*, *431*, *445*, *477*, *479*, *487*, *515*, *679*, *681*, *763*

launch_mfrmr_viewer, 433
launch_mfrmr_viewer(), *309*, *487*, *651*, *652*, *655*
lifecycle::deprecate_warn(), *32*
list_mfrmr_data, 435
list_mfrmr_data(), *263*, *437*, *438*
lme4::bootMer(), *160*, *161*
lme4::lmer(), *594*

load_mfrm_calibration (mfrm_calibration_workflow), 537
load_mfrm_calibration(), 468, 525
load_mfrmr_data, 436
load_mfrmr_data(), 8, 11, 12, 262, 263, 435, 436, 445, 458, 461
logLik.mfrm_cml_fit (mfrm_fit_extractors), 588
logLik.mfrm_comparison_fit (mfrm_fit_extractors), 588
logLik.mfrm_fit (mfrm_fit_extractors), 588
logLik.mfrm_jeffreys_fit (summary.mfrm_jeffreys_fit), 865

main-effects method, 562
make_anchor_table, 438
make_anchor_table(), 11, 43, 45, 95, 104, 238, 240, 301, 382, 469–471, 473, 806, 807
mantel_haenszel_table, 440
mantel_haenszel_table(), 11, 26, 449, 472, 477, 488, 677
measurable_summary_table, 443
measurable_summary_table(), 44, 379, 500, 527, 769, 848
mfrm_calibration_capabilities, 532, 536
mfrm_calibration_capabilities(), 9, 331, 412, 540
mfrm_calibration_methods, 533
mfrm_calibration_score_methods, 9, 504, 533, 534, 544
mfrm_calibration_workflow, 9, 478, 525, 533, 536, 537
mfrm_cluster, 544
mfrm_cluster_compare, 546
mfrm_cluster_compare(), 524, 546, 547, 550, 552, 555, 557, 587, 617, 688
mfrm_cluster_hierarchical, 549
mfrm_cluster_hierarchical(), 517, 522, 546–548, 553, 555, 615, 638, 687
mfrm_cluster_imputed, 552
mfrm_cluster_imputed(), 546–548, 552, 554, 557, 587, 646, 686, 688
mfrm_cluster_kmeans, 555
mfrm_cluster_kmeans(), 517, 522, 547, 553, 556, 587, 614, 615, 617, 638, 686
mfrm_cluster_pam (mfrm_cluster), 544
mfrm_cluster_pam(), 517, 522, 545, 547, 548, 550, 552, 553, 555, 557, 587, 615, 686, 688
mfrm_curve_intervals, 558
mfrm_curve_intervals(), 369, 412, 416, 468, 529, 653
mfrm_d_study, 562
mfrm_d_study(), 7, 13, 128, 189, 198, 210, 286, 320, 449, 461, 476, 478, 482, 515, 517, 521, 524, 568, 570, 571, 595–597, 599, 600, 603, 607, 610, 631, 696–699, 709, 711, 713–715, 718, 867, 869
mfrm_d_study.mfrm_fit (mfrm_d_study_main_effects), 569
mfrm_d_study.mfrm_generalizability (mfrm_d_study_main_effects), 569
mfrm_d_study_main_effects, 565, 569
mfrm_dff_methods, 561
mfrm_facet_intervals, 572
mfrm_facet_intervals(), 7, 15, 181, 182, 184, 248, 271, 359, 367, 370, 376, 377, 381, 383, 384, 386, 388, 412, 413, 426, 427, 453, 464, 467, 471, 476, 489, 492, 493, 499, 516, 517, 519, 527, 542, 574, 590, 591, 618, 619, 622, 632, 633, 648, 652, 653
mfrm_features, 586
mfrm_features(), 515, 522, 544, 546, 548, 549, 553, 555, 556, 614
mfrm_fit_extractors, 182, 391, 588, 593
mfrm_fit_responses, 391, 591, 591
mfrm_g_study, 597
mfrm_g_study(), 7, 128, 205, 213, 220, 286, 320, 449, 461, 478, 515, 517, 521, 522, 524, 562, 566, 568, 570, 594, 597, 609, 631, 698, 714, 716, 719, 878
mfrm_generalizability, 594
mfrm_generalizability(), 13, 286, 320, 482, 524, 562, 563, 569–571, 603
mfrm_misfit_thresholds, 605
mfrm_misfit_thresholds(), 333, 334, 347, 350, 519
mfrm_multivariate_d_compare, 606

mfrm_multivariate_d_compare(), *62*, *128*, *191*, *199*, *205*, *206*, *210*, *214*, *220*, *321*, *461*, *481*, *515*, *521*, *524*, *565*, *568*, *601*, *602*, *607*, *610*, *631*, *712*
mfrm_multivariate_d_study *(*mfrm_d_study*)*, 562
mfrm_multivariate_gstudy *(*mfrm_g_study*)*, 597
mfrm_network_analysis, 611
mfrm_network_analysis(), *105–107*, *472*, *507*, *508*, *512*, *763*, *765*, *848*
mfrm_pca, 614
mfrm_pca(), *517*, *522*, *556*, *557*, *587*, *614*, *617*
mfrm_population_intervals, 617
mfrm_population_intervals(), *182*, *376*, *380*, *381*, *383*, *463*, *464*, *471*, *476*, *583*, *585*, *619*, *621*, *622*, *648*, *745*
mfrm_random_rater_intervals, 623
mfrm_random_rater_intervals(), *398*, *464*, *624*, *648*
mfrm_report, 627
mfrm_report(), *7*, *8*, *128*, *269*, *289*, *300*, *302*, *305*, *307*, *386*, *409*, *412*, *478*, *488*, *493*, *497*, *499*, *500*, *515*, *517*, *521*, *525*, *528*, *571*, *584*, *585*, *637*, *652*, *654*, *756*, *760*, *779*, *780*, *784*, *786*, *826*
mfrm_residual_network, 635
mfrm_residual_network(), *128*, *515*, *627*, *631*, *632*, *636*, *761*, *764*
mfrm_response_diagnostics, 639
mfrm_response_diagnostics(), *149*, *245*, *249*, *347*, *368*, *370*, *398*, *404*, *413*, *452*, *476*, *485*, *504*, *521*, *592*, *593*, *629*, *632*, *633*, *640*, *648*, *649*, *652*, *827*, *829*
mfrm_response_imputations, 644
mfrm_response_imputations(), *645*
mfrm_results, 647
mfrm_results(), *8*, *10*, *276*, *289*, *300*, *302*, *306*, *307*, *309*, *370*, *386*, *396*, *398*, *402*, *404*, *412*, *416*, *421*, *423*, *425*, *433*, *434*, *478*, *484*, *493*, *497*, *499*, *515*, *521–523*, *525*, *542*, *577*, *584*, *585*, *617*, *622*, *627*, *629*, *632–634*, *643*, *656*, *732*, *733*, *735*, *736*, *827*, *832*
mfrm_results_interactive, 655
mfrm_results_interactive(), *10*, *434*, *651*
mfrm_screening_performance, 657
mfrm_screening_performance(), *295*, *517*, *528*, *658*, *662*
mfrm_screening_sensitivity, 659
mfrm_screening_sensitivity(), *350*, *517*, *605*, *659*, *661*
mfrm_threshold_profiles, 663
mfrm_threshold_profiles(), *134*, *663*, *664*, *818*
mfrmr *(*mfrmr-package*)*, 7
mfrmr-package, 7, *372*, *417*
mfrmr_agreement_data, *435*, *437*, 445
mfrmr_compatibility_layer, *22*, *34*, *85*, *98*, *103*, *110*, *119*, *120*, *157*, *226*, *259*, *260*, *265*, *268*, *270*, *277*, *280*, *299*, *302*, *311*, *317*, *323*, *325*, *327*, *328*, *330*, *335*, *443*, 446, *502*, *531*, *591*, *593*, *599*, *702*, *740*, *749*, *794*, *813*, *817*, *822*, *845*, *846*, *874*, *877*, *882*, *883*
mfrmr_example_agreement *(*mfrmr_agreement_data*)*, 445
mfrmr_example_bias *(*mfrmr_example_data*)*, 457
mfrmr_example_core *(*mfrmr_example_data*)*, 457
mfrmr_example_data, 457
mfrmr_example_gtheory_crossed *(*mfrmr_gtheory_data*)*, 460
mfrmr_example_gtheory_nested *(*mfrmr_gtheory_data*)*, 460
mfrmr_example_operational *(*mfrmr_example_data*)*, 457
mfrmr_example_operational_design, 459
mfrmr_gtheory_data, *435*, *437*, 460
mfrmr_interval_guide, 461
mfrmr_interval_guide(), *224*, *404*, *476*, *483*, *484*, *489*, *513*, *515*, *516*, *521*, *527*, *622*, *736*
mfrmr_linking_and_dff, *9*, *27*, *45*, *100*, *233*, *238*, *240*, *383*, *427*, *442*, *456*, 464, *476*, *502*, *513*, *531*, *613*, *677*, *693*, *807*, *848*
mfrmr_methods_reference, *9*, *191*, *214*, *220*, *367*, *429*, 474, *488*, *506*, *515*, *516*, *601*, *756*, *759*
mfrmr_model_maps, 484, *649*, *655*

mfrmr_output_guide, 486
mfrmr_output_guide(), 9, 64, 321, 331, 404, 484, 492, 634, 655, 797
mfrmr_plot_migration, 31, 40, 168, 240, 252, 256, 278, 335, 477, 490, 513, 518, 693, 701, 703, 797, 818
mfrmr_reporting_and_apa, 9, 49, 85, 252, 391, 456, 492, 502, 513, 531
mfrmr_reports_and_tables, 9, 98, 141, 143, 226, 280, 330, 340, 386, 446, 448, 456, 473, 497, 498, 513, 525, 531, 846
mfrmr_visual_diagnostics, 9, 40, 69, 70, 141, 143, 252, 255, 278, 431, 444, 446, 456, 464, 473, 475, 492, 498, 502, 503, 515, 531, 613, 636, 638, 705, 708, 767, 769, 797, 813, 848, 873, 874, 876, 877
mfrmr_workflow_methods, 8, 9, 235, 286, 351, 391, 404, 417, 456, 473, 474, 484, 488, 494, 497, 502, 513, 514, 813, 859
mfrmRFacets (run_mfrm_facets), 809
mfrmRFacets(), 11, 446, 527, 531
mice::complete(), 555
mice::mice(), 554, 555, 645
mice::pool.scalar(), 729
mirt::itemfit(), 422
mirt::mirt(), 421, 422
mirt::personfit(), 422
mml_quadrature_sensitivity, 664
mml_quadrature_sensitivity(), 81, 150, 183, 185, 186, 355, 364, 369, 376, 385, 413, 416, 529, 538, 540, 544, 642

nobs.mfrm_cml_fit (mfrm_fit_extractors), 588
nobs.mfrm_comparison_fit (mfrm_fit_extractors), 588
nobs.mfrm_fit (mfrm_fit_extractors), 588
normalize_conquest_overlap_exports, 667
normalize_conquest_overlap_files, 669
normalize_conquest_overlap_files(), 11, 88, 89, 669, 799–801
normalize_conquest_overlap_tables, 672
normalize_conquest_overlap_tables(), 11, 88, 89, 669, 671, 799–801

ordinal::clm(), 676
ordinal_logistic_table, 675
ordinal_logistic_table(), 8, 11, 26, 442, 472, 477, 488

pairwise_icc_table, 678
pairwise_icc_table(), 162, 163, 419, 428, 431, 433, 445, 477, 479, 487, 681
pairwise_kappa_table, 680
pairwise_kappa_table(), 419, 431, 445, 477, 479, 487, 515
plot(), 49, 131, 502, 584, 637
plot() method, 463
plot(ci), 621
plot(fit, type = ccc), 63, 139
plot(fit, type = shrinkage), 63
plot(reference, comparison = comparison), 64
plot(x), 11, 95, 100, 168, 240, 469, 473, 500, 503, 507, 509, 511–513, 526, 528–530, 797, 817, 818, 864
plot(x, type = agreement), 429, 431, 503, 504, 507–509, 511–513, 682, 765
plot(x, type = agreement, view = heatmap), 507, 512, 513
plot(x, type = apa_summary), 464
plot(x, type = bias), 11, 72, 276, 464, 496, 504, 507, 509, 511–513, 530, 682
plot(x, type = bubble), 61, 350, 504, 708
plot(x, type = displacement), 11, 121, 260, 464, 503, 507–509, 513, 526, 530, 682, 685
plot(x, type = facet_dashboard), 11, 335, 504
plot(x, type = facet_equivalence), 11, 31
plot(x, type = facets_chisq), 318, 340, 503, 504, 507, 513, 682
plot(x, type = fair_average), 157, 344, 345, 464, 507, 513, 682, 685
plot(x, type = guttman), 507, 511, 513
plot(x, type = heatmap), 11, 27, 255, 256, 258, 458, 469, 473, 507, 509, 511–513
plot(x, type = local_dependence), 508, 752, 754
plot(x, type = marginal_fit), 11, 121, 134, 245, 494, 496, 504, 507–509,

512, *513*, *526*
plot(x, type = marginal_pairwise), *11*, *121*, *134*, *245*, *494*, *504*, *507–509*, *512*, *513*, *526*
plot(x, type = qc), *13*, *335*, *496*, *503*, *504*, *508*, *509*, *512*, *513*, *528–530*, *811–813*, *818*
plot(x, type = rater_severity), *464*
plot(x, type = reliability), *477*, *508*
plot(x, type = residual_matrix), *508*
plot(x, type = residual_pca), *11*, *38*, *40*, *496*, *503*, *507*, *509*, *513*, *683*
plot(x, type = residual_qq), *507*, *513*
plot(x, type = shrinkage, view = funnel), *508*
plot(x, type = summary), *469*, *473*, *507*, *509*, *511–513*
plot(x, type = unexpected), *11*, *121*, *503*, *504*, *507–509*, *513*, *526*, *530*, *682*, *685*, *876*
plot(x, type = wright, view = unified), *332*, *464*, *507*, *511*, *529*, *708*
plot.apa_table *(*apa_table*)*, 45
plot.apa_table(), *49*, *128*, *131*
plot.mfrm_anchor_drift *(*detect_anchor_drift*)*, 236
plot.mfrm_anchor_drift(), *490*
plot.mfrm_anchor_review *(*review_mfrm_anchors*)*, 802
plot.mfrm_bias *(*estimate_bias*)*, 270
plot.mfrm_bias(), *265*, *266*, *269*, *490*
plot.mfrm_bias_collection *(*estimate_all_bias*)*, 264
plot.mfrm_bias_collection(), *268*, *269*
plot.mfrm_bundle, 682
plot.mfrm_bundle(), *761*, *764*, *765*
plot.mfrm_calibration_score *(*mfrm_calibration_score_methods*)*, 534
plot.mfrm_clusters, 685
plot.mfrm_clusters(), *546*, *550*, *555*, *557*
plot.mfrm_curve_intervals *(*mfrm_curve_intervals*)*, 558
plot.mfrm_d_study, 696
plot.mfrm_d_study(), *571*
plot.mfrm_data_description *(*describe_mfrm_data*)*, 226
plot.mfrm_design_evaluation, *287*, 689, *786*, *853*
plot.mfrm_design_evaluation(), *786*
plot.mfrm_design_recommendation *(*recommend_mfrm_design*)*, 782
plot.mfrm_design_recommendation(), *784*, *786*
plot.mfrm_diagnostic_screening, 690, *857*
plot.mfrm_diagnostics *(*diagnose_mfrm*)*, 244
plot.mfrm_diagnostics(), *490*
plot.mfrm_dif, *472*, 692
plot.mfrm_dif(), *256*, *490*
plot.mfrm_dif_interaction *(*dif_interaction_table*)*, 253
plot.mfrm_dif_interaction(), *490*
plot.mfrm_dstudy_intervals, 694
plot.mfrm_dstudy_intervals(), *869*
plot.mfrm_equating_chain *(*build_equating_chain*)*, 90
plot.mfrm_extended_comparison *(*compare_mfrm*)*, 147
plot.mfrm_extended_comparison(), *62*, *153*
plot.mfrm_facet_dashboard *(*facet_quality_dashboard*)*, 332
plot.mfrm_facet_equivalence *(*analyze_facet_equivalence*)*, 28
plot.mfrm_facet_intervals *(*mfrm_facet_intervals*)*, 572
plot.mfrm_facet_intervals(), *585*, *622*
plot.mfrm_facet_nesting, 699
plot.mfrm_facet_sample_review *(*facet_small_sample_review*)*, 336
plot.mfrm_facets_results *(*read_facets_results*)*, 772
plot.mfrm_facets_results(), *772*
plot.mfrm_facets_run *(*run_mfrm_facets*)*, 809
plot.mfrm_fair_intervals *(*confint.mfrm_fair_average*)*, 172
plot.mfrm_fair_intervals(), *177*
plot.mfrm_fit, *506*, 700
plot.mfrm_fit(), *11*, *13*, *141*, *143*, *478*, *492*, *507*, *513*, *528*, *529*, *531*, *769*,

811–813
plot.mfrm_gpcm_bootstrap *(*plot.mfrm_slope_intervals*)*, 721
plot.mfrm_hierarchical_clusters *(*mfrm_cluster_hierarchical*)*, 549
plot.mfrm_imputed_clusters *(*plot.mfrm_clusters*)*, 685
plot.mfrm_information *(*compute_information*)*, 164
plot.mfrm_information(), *490*
plot.mfrm_multivariate_d_comparison *(*mfrm_multivariate_d_compare*)*, 606
plot.mfrm_multivariate_d_comparison(), *610*
plot.mfrm_multivariate_d_study, 709
plot.mfrm_multivariate_d_study(), *204*, *564*, *568*, *608*, *699*, *719*, *869*
plot.mfrm_multivariate_gstudy, 715
plot.mfrm_multivariate_gstudy(), *220*, *478*
plot.mfrm_network_ergm *(*fit_rater_network*)*, 407
plot.mfrm_network_ergm(), *409*, *684*
plot.mfrm_pca *(*mfrm_pca*)*, 614
plot.mfrm_pca(), *617*
plot.mfrm_person_scores *(*score_mfrm_persons*)*, 826
plot.mfrm_pooled *(*pool_mfrm_imputed*)*, 725
plot.mfrm_population_intervals *(*mfrm_population_intervals*)*, 617
plot.mfrm_population_intervals(), *490*, *622*
plot.mfrm_posterior_summary *(*read_mfrm_posterior*)*, 777
plot.mfrm_posterior_summary(), *780*
plot.mfrm_qc_pipeline *(*run_qc_pipeline*)*, 814
plot.mfrm_qc_pipeline(), *490*
plot.mfrm_random_rater *(*fit_mfrm_random_rater*)*, 393
plot.mfrm_random_rater(), *398*
plot.mfrm_random_rater_intervals *(*mfrm_random_rater_intervals*)*, 623
plot.mfrm_random_rater_intervals(), *626*
plot.mfrm_random_rater_scores *(*score_mfrm_random_rater*)*, 829
plot.mfrm_random_rater_scores(), *832*
plot.mfrm_recovery_assessment *(*assess_mfrm_recovery*)*, 54
plot.mfrm_recovery_simulation, 719
plot.mfrm_recovery_simulation(), *57*
plot.mfrm_residual_network *(*mfrm_residual_network*)*, 635
plot.mfrm_residual_pca *(*analyze_residual_pca*)*, 35
plot.mfrm_residual_pca(), *245*, *252*, *490*
plot.mfrm_response_diagnostics *(*mfrm_response_diagnostics*)*, 639
plot.mfrm_response_diagnostics(), *62*, *643*
plot.mfrm_response_time_review *(*response_time_review*)*, 795
plot.mfrm_screening_performance *(*mfrm_screening_performance*)*, 657
plot.mfrm_screening_sensitivity *(*mfrm_screening_sensitivity*)*, 659
plot.mfrm_screening_sensitivity(), *62*
plot.mfrm_signal_detection, *872*
plot.mfrm_signal_detection *(*evaluate_mfrm_signal_detection*)*, 291
plot.mfrm_slope_intervals, 721
plot.mfrm_summary_table_bundle *(*build_summary_table_bundle*)*, 125
plot.mfrm_testlet *(*fit_mfrm_testlet*)*, 399
plot.mfrm_testlet(), *404*, *736*
plot.mfrm_testlet_scores *(*predict.mfrm_testlet*)*, 733
plot.mfrm_testlet_scores(), *62*, *398*, *404*, *626*, *736*, *827*, *829*, *832*
plot_data, 723
plot_data(), *62–64*, *72*, *140*, *143*, *249*, *267*, *308*, *332*, *397*, *402, 403*, *409*, *416*, *478*, *489, 490*, *492*, *509*, *514*,

521–524, *529*, *535*, *536*, *550*, *571*, *577*, *584*, *609*, *616*, *621*, *626*, *636*, *640*, *641*, *659*, *687*, *695*, *698*, *707*, *710*, *712*, *713*, *722*, *725*, *728*, *735*, *736*, *760*, *779*, *780*, *786*, *832*, *857*
plot_data_components, 724
plot_data_components(), *63*, *402*, *490*, *509*, *536*, *723*, *736*
plot_facet_equivalence(), *29*
plot_facet_quality_dashboard(), *334*
plot_fair_average, *704*
plot_rater_severity_profile(), *477*
plot_response_time_review(), *796*
pool_mfrm_imputed, 725
pool_mfrm_imputed(), *392*, *447*, *464*, *517*, *522*, *585*, *644*, *646*, *726*, *729*
precision_review *(*review_accessors*)*, 797
precision_review_report, 729
precision_review_report(), *128*, *167*, *492–494*, *496*, *497*, *500–502*, *649*, *793*
predict.mfrm_cml_fit *(*mfrm_fit_responses*)*, 591
predict.mfrm_comparison_fit *(*mfrm_fit_responses*)*, 591
predict.mfrm_fit *(*mfrm_fit_responses*)*, 591
predict.mfrm_random_rater, 731
predict.mfrm_random_rater(), *398*, *831*, *832*
predict.mfrm_testlet, 733
predict.mfrm_testlet(), *223*, *402*, *404*, *452*, *464*, *643*, *734*, *827*
predict_mfrm_population, 737
predict_mfrm_population(), *11*, *13*, *101*, *108*, *109*, *129*, *299*, *301*, *355*, *455*, *526*, *528–530*, *743*, *749*, *869*, *870*
predict_mfrm_units, 741
predict_mfrm_units(), *11*, *13*, *14*, *20*, *47*, *49*, *102*, *108*, *129*, *145–147*, *299*, *301*, *369*, *370*, *378*, *380*, *411*, *413*, *417*, *450*, *452*, *455*, *468*, *477*, *495*, *524*, *526*, *528–530*, *532*, *536*, *537*, *539*, *540*, *542–544*, *592*, *593*, *622*, *649*, *653*, *667*, *738*, *743*, *749*, *820–822*, *828*, *829*
print.mfrm_anchor_drift *(*detect_anchor_drift*)*, 236
print.mfrm_anchored_fit *(*anchor_to_baseline*)*, 41
print.mfrm_apa_text, 750
print.mfrm_bias_collection *(*estimate_all_bias*)*, 264
print.mfrm_calibration *(*mfrm_calibration_methods*)*, 533
print.mfrm_calibration_score *(*mfrm_calibration_score_methods*)*, 534
print.mfrm_cluster_comparison *(*mfrm_cluster_compare*)*, 546
print.mfrm_clusters *(*mfrm_cluster*)*, 544
print.mfrm_cml_fit *(*fit_mfrm*)*, 350
print.mfrm_comparison_fit *(*fit_mfrm*)*, 350
print.mfrm_curve_intervals *(*mfrm_curve_intervals*)*, 558
print.mfrm_design_recommendation *(*recommend_mfrm_design*)*, 782
print.mfrm_dff *(*mfrm_dff_methods*)*, 561
print.mfrm_dif *(*mfrm_dff_methods*)*, 561
print.mfrm_equating_chain *(*build_equating_chain*)*, 90
print.mfrm_extended_comparison *(*compare_mfrm*)*, 147
print.mfrm_facet_intervals *(*mfrm_facet_intervals*)*, 572
print.mfrm_facets_results *(*read_facets_results*)*, 772
print.mfrm_fair_intervals *(*confint.mfrm_fair_average*)*, 172
print.mfrm_features *(*mfrm_features*)*, 586
print.mfrm_gpcm_bootstrap *(*bootstrap_mfrm_gpcm*)*, 78
print.mfrm_imputed_clusters *(*mfrm_cluster_imputed*)*, 552
print.mfrm_imputed_fits *(*fit_mfrm_imputed*)*, 391
print.mfrm_jeffreys_fit *(*summary.mfrm_jeffreys_fit*)*, 865
print.mfrm_multivariate_d_comparison *(*mfrm_multivariate_d_compare*)*, 606
print.mfrm_multivariate_d_study *(*mfrm_d_study*)*, 562

print.mfrm_multivariate_gstudy (mfrm_g_study), 597
print.mfrm_pca (mfrm_pca), 614
print.mfrm_person_scores (score_mfrm_persons), 826
print.mfrm_pooled (pool_mfrm_imputed), 725
print.mfrm_population_intervals (mfrm_population_intervals), 617
print.mfrm_posterior_summary (read_mfrm_posterior), 777
print.mfrm_random_rater (fit_mfrm_random_rater), 393
print.mfrm_random_rater_intervals (mfrm_random_rater_intervals), 623
print.mfrm_random_rater_scores (score_mfrm_random_rater), 829
print.mfrm_residual_network (mfrm_residual_network), 635
print.mfrm_response_diagnostics (mfrm_response_diagnostics), 639
print.mfrm_response_imputations (mfrm_response_imputations), 644
print.mfrm_screening_performance (mfrm_screening_performance), 657
print.mfrm_screening_sensitivity (mfrm_screening_sensitivity), 659
print.mfrm_sim_spec (build_mfrm_sim_spec), 113
print.mfrm_testlet (fit_mfrm_testlet), 399
print.mfrm_testlet_scores (predict.mfrm_testlet), 733
print.summary.mfrm_anchor_drift (detect_anchor_drift), 236
print.summary.mfrm_anchored_fit (anchor_to_baseline), 41
print.summary.mfrm_bias_collection (estimate_all_bias), 264
print.summary.mfrm_calibration (mfrm_calibration_methods), 533
print.summary.mfrm_calibration_score (mfrm_calibration_score_methods), 534
print.summary.mfrm_design_recommendation (recommend_mfrm_design), 782
print.summary.mfrm_dff (mfrm_dff_methods), 561
print.summary.mfrm_dif (mfrm_dff_methods), 561
print.summary.mfrm_equating_chain (build_equating_chain), 90
print.summary.mfrm_imported_fit (import_mirt_fit), 421
print.summary.mfrm_multivariate_d_study (summary.mfrm_multivariate_d_study), 866
psych::describe(), *230*

q3_statistic, 751
q3_statistic(), *39*, *477*, *515*, *523*, *635*

rater_centrality_table, 754
rater_centrality_table(), *333, 334*, *407*, *477*, *480*, *487*, *759*
rater_halo_network_analysis, 757
rater_halo_network_analysis(), *107*, *407*, *409*, *448*, *477*, *480*, *507, 508*, *525*, *612*, *756*, *760*, *764*
rater_network_analysis, 761
rater_network_analysis(), *107*, *407*, *409*, *507, 508*, *525*, *612*, *684*, *761*
rating_scale_table, 765
rating_scale_table(), *141*, *143*, *353*, *379*, *444*, *475, 476*, *478*, *500*, *502*, *506*, *527*, *529*, *812, 813*
read_facets_fit_table, 770
read_facets_fit_table(), *321*, *329*, *331*, *775*
read_facets_results, 772
read_facets_results(), *770*, *774*
read_mfrm_posterior, 777
read_mfrm_posterior(), *305*, *488*, *500*, *631*, *778*, *780*
recode_missing_codes, 781
recode_missing_codes(), *9*, *228*, *234, 235*, *353*, *520*
recommend_mfrm_design, 782
recommend_mfrm_design(), *571*, *631*, *783*, *786*
reference_case_benchmark, 787

reference_case_benchmark(), *89*, *355*, *447*, *528*, *801*
reference_case_review, 789
reference_case_review(), *447*, *528*
reporting_checklist, 791
reporting_checklist(), *11*, *31*, *33, 34*, *49*, *85*, *104*, *129*, *131*, *157*, *243*, *268, 269*, *299*, *301, 302*, *339*, *379*, *417*, *446, 447*, *455, 456*, *492–494*, *496, 497*, *500–503*, *508, 509*, *512*, *527–530*, *634*, *649*, *655*, *731*, *791*, *794*, *813*, *880*
residuals(fit, pearson), *246*
residuals.mfrm_cml_fit *(*mfrm_fit_responses*)*, 591
residuals.mfrm_comparison_fit *(*mfrm_fit_responses*)*, 591
residuals.mfrm_fit *(*mfrm_fit_responses*)*, 591
response_time_review, 795
response_time_review(), *507*, *513*, *525*, *796, 797*
retire_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
review_accessors, 797
review_conquest_overlap, 798
review_conquest_overlap(), *11*, *88, 89*, *669*, *671*, *673, 674*
review_mfrm_anchors, 802
review_mfrm_anchors(), *11*, *99, 100*, *129*, *228*, *235*, *360*, *383*, *439*, *469, 470*, *475*, *515*, *526*, *528*, *627*, *804*, *806, 807*
review_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
review_mfrm_imputations *(*mfrm_response_imputations*)*, 644
review_mfrm_imputations(), *392*, *447*, *517*, *519*, *522*, *645*, *729*
review_peer_design, 808
review_peer_design(), *808, 809*
rmarkdown::render(), *493*
run_mfrm_facets, 809
run_mfrm_facets(), *11*, *86*, *101*, *103*, *108–110*, *129*, *157*, *299*, *446, 447*, *456*, *527*, *531*, *647*, *655, 656*, *811*, *813*, *858, 859*
run_qc_pipeline, 814
run_qc_pipeline(), *10*, *421*, *423*, *425*, *492*, *497*, *523*, *815, 816*, *818*

sample_mfrm_plausible_values, 819
sample_mfrm_plausible_values(), *11*, *13*, *20*, *102*, *108*, *129*, *299*, *301*, *417*, *452*, *455*, *495*, *526*, *528–530*, *543*, *748*, *820*, *822*, *828*
save_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
saveRDS(), *223*, *463*, *543*, *749*, *828*
score_comparison_table, 823
score_comparison_table(), *67, 68*, *147*, *407*, *433*, *477*, *479*, *487*, *679*, *681*, *756*
score_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
score_mfrm_calibration(), *43*, *47*, *129*, *377*, *382*, *450*, *452*, *465*, *509*, *534*, *537*, *543*, *592*, *649*, *749*, *828*
score_mfrm_persons, 826
score_mfrm_persons(), *47*, *149*, *452*, *462*, *477*, *485*, *517*, *520*, *542–544*, *592*, *649*, *749*, *828*
score_mfrm_random_rater, 829
score_mfrm_random_rater(), *398*, *452*, *643*, *649*, *732*, *827*, *830*
shrinkage_report, 833
simulate_mfrm_data, 834
simulate_mfrm_data(), *10, 11*, *13*, *116–118*, *125*, *283, 284*, *287*, *289, 290*, *294*, *297*, *317*, *379*, *470*, *528*, *530*, *809*, *845*
simulate_mfrm_recovery, 839
simulate_mfrm_recovery(), *11*, *54–57*, *112*, *129*, *284*, *313*, *379*, *500*, *502*, *528–530*, *720*
specifications_report, 845
specifications_report(), *226*, *280*, *468*, *500*, *502*, *794*
stats::confint(), *412*
stats::cutree(), *550*, *637*
stats::hclust(), *550*, *637*
stats::kmeans(), *556, 557*

stats::mantelhaen.test(), *441*, *442*
stats::optim(), *356*, *366*
stats::p.adjust(), *25*, *441*, *675*, *758*
stats::prcomp(), *615*, *617*
subset_connectivity_report, 847
subset_connectivity_report(), *27*, *107*, *334*, *469*, *471*, *473*, *476*, *500*, *512*, *515*, *529*, *613*, *761*, *764*, *884*
summary(), *49*, *85*, *131*, *720*
summary.apa_table *(*apa_table*)*, 45
summary.mfrm_anchor_drift *(*detect_anchor_drift*)*, 236
summary.mfrm_anchor_review *(*review_mfrm_anchors*)*, 802
summary.mfrm_anchored_fit *(*anchor_to_baseline*)*, 41
summary.mfrm_apa_outputs, *794*
summary.mfrm_apa_outputs *(*build_apa_outputs*)*, 81
summary.mfrm_bias *(*estimate_bias*)*, 270
summary.mfrm_bias_collection *(*estimate_all_bias*)*, 264
summary.mfrm_bias_collection(), *269*
summary.mfrm_bundle, 849
summary.mfrm_calibration *(*mfrm_calibration_methods*)*, 533
summary.mfrm_calibration_score *(*mfrm_calibration_score_methods*)*, 534
summary.mfrm_cluster_comparison *(*mfrm_cluster_compare*)*, 546
summary.mfrm_clusters *(*mfrm_cluster*)*, 544
summary.mfrm_cml_fit *(*fit_mfrm*)*, 350
summary.mfrm_comparison_fit *(*fit_mfrm*)*, 350
summary.mfrm_curve_intervals *(*mfrm_curve_intervals*)*, 558
summary.mfrm_d_study *(*mfrm_d_study_main_effects*)*, 569
summary.mfrm_data_description *(*describe_mfrm_data*)*, 226
summary.mfrm_data_description(), *235*, *520*
summary.mfrm_design_evaluation, *287*, *690*, *786*, 851
summary.mfrm_design_evaluation(), *783*, *784*, *786*
summary.mfrm_design_recommendation *(*recommend_mfrm_design*)*, 782
summary.mfrm_design_recommendation(), *784*, *786*
summary.mfrm_dff *(*mfrm_dff_methods*)*, 561
summary.mfrm_diagnostic_screening, 856
summary.mfrm_diagnostics, 853
summary.mfrm_diagnostics(), *605*, *606*
summary.mfrm_dif *(*mfrm_dff_methods*)*, 561
summary.mfrm_equating_chain *(*build_equating_chain*)*, 90
summary.mfrm_extended_comparison *(*compare_mfrm*)*, 147
summary.mfrm_facet_dashboard *(*facet_quality_dashboard*)*, 332
summary.mfrm_facet_intervals *(*mfrm_facet_intervals*)*, 572
summary.mfrm_facets_results *(*read_facets_results*)*, 772
summary.mfrm_facets_run, 858
summary.mfrm_fair_intervals *(*confint.mfrm_fair_average*)*, 172
summary.mfrm_features *(*mfrm_features*)*, 586
summary.mfrm_fit, 859
summary.mfrm_fit(), *11*, *13*, *235*, *340*, *427*, *531*, *591*, *665*, *856*, *859*
summary.mfrm_gpcm_bootstrap *(*bootstrap_mfrm_gpcm*)*, 78
summary.mfrm_imported_fit *(*import_mirt_fit*)*, 421
summary.mfrm_imputed_clusters *(*mfrm_cluster_imputed*)*, 552
summary.mfrm_imputed_fits *(*fit_mfrm_imputed*)*, 391
summary.mfrm_jeffreys_fit, 865
summary.mfrm_jeffreys_fit(), *364*, *374*
summary.mfrm_linking_review *(*build_linking_review*)*, 99
summary.mfrm_misfit_casebook *(*build_misfit_casebook*)*, 119
summary.mfrm_model_choice_review *(*build_model_choice_review*)*, 121
summary.mfrm_multivariate_d_comparison *(*mfrm_multivariate_d_compare*)*,

606
summary.mfrm_multivariate_d_study, 866
summary.mfrm_multivariate_d_study(), *478*, *481*, *499*, *565*
summary.mfrm_multivariate_gstudy *(*mfrm_g_study*)*, 597
summary.mfrm_network_ergm *(*fit_rater_network*)*, 407
summary.mfrm_network_review *(*build_mfrm_network_review*)*, 104
summary.mfrm_pca *(*mfrm_pca*)*, 614
summary.mfrm_peer_review_design_review *(*review_peer_design*)*, 808
summary.mfrm_person_fit_indices *(*compute_person_fit_indices*)*, 169
summary.mfrm_person_scores *(*score_mfrm_persons*)*, 826
summary.mfrm_plausible_values, *822*
summary.mfrm_plausible_values *(*sample_mfrm_plausible_values*)*, 819
summary.mfrm_pooled *(*pool_mfrm_imputed*)*, 725
summary.mfrm_population_intervals *(*mfrm_population_intervals*)*, 617
summary.mfrm_population_prediction, *740*, 869
summary.mfrm_posterior_summary *(*read_mfrm_posterior*)*, 777
summary.mfrm_random_rater *(*fit_mfrm_random_rater*)*, 393
summary.mfrm_random_rater_intervals *(*mfrm_random_rater_intervals*)*, 623
summary.mfrm_random_rater_scores *(*score_mfrm_random_rater*)*, 829
summary.mfrm_reporting_checklist *(*reporting_checklist*)*, 791
summary.mfrm_residual_network *(*mfrm_residual_network*)*, 635
summary.mfrm_response_diagnostics *(*mfrm_response_diagnostics*)*, 639
summary.mfrm_response_imputations *(*mfrm_response_imputations*)*, 644
summary.mfrm_response_time_review *(*response_time_review*)*, 795
summary.mfrm_screening_performance *(*mfrm_screening_performance*)*, 657
summary.mfrm_screening_sensitivity *(*mfrm_screening_sensitivity*)*, 659
summary.mfrm_signal_detection, *297*, 871
summary.mfrm_summary_table_bundle *(*build_summary_table_bundle*)*, 125
summary.mfrm_testlet *(*fit_mfrm_testlet*)*, 399
summary.mfrm_testlet_scores *(*predict.mfrm_testlet*)*, 733
summary.mfrm_threshold_profiles *(*mfrm_threshold_profiles*)*, 663
summary.mfrm_unit_prediction, *749*
summary.mfrm_unit_prediction *(*predict_mfrm_units*)*, 741
summary.mfrm_weighting_review *(*build_weighting_review*)*, 135
supersede_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537

TAM::msq.itemfit(), *424*
TAM::tam.mml(), *423*
TAM::tam.mml.mfr(), *423*
TAM::tam.personfit(), *424*

unexpected_after_bias_table, 872
unexpected_after_bias_table(), *70*, *454*
unexpected_response_table, 874
unexpected_response_table(), *119–121*, *260*, *330*, *346*, *379*, *454*, *527*, *529*, *705*, *851*, *873*, *874*
utils::write.csv(), *37*, *72*, *146*, *345*, *406*, *429*, *433*, *442*, *479*, *677*, *679*, *681*, *753*, *756*, *760*, *825*
utils::zip(), *299*

validate_mfrm_calibration *(*mfrm_calibration_workflow*)*, 537
validate_mfrm_calibration(), *537*
vcov(), *180*

vcov.mfrm_cml_fit
(mfrm_fit_extractors), 588
vcov.mfrm_comparison_fit
(mfrm_fit_extractors), 588
vcov.mfrm_facet_intervals
(mfrm_facet_intervals), 572
vcov.mfrm_fit (mfrm_fit_extractors), 588
vcov.mfrm_jeffreys_fit
(summary.mfrm_jeffreys_fit),
865
vcov.mfrm_multivariate_gstudy, 877
vcov.mfrm_multivariate_gstudy(), *204*,
216, *217*, *320*, *480*, *489*, *599*
vcov.mfrm_network_ergm
(fit_rater_network), 407
vcov.mfrm_population_intervals
(mfrm_population_intervals),
617
visual_reporting_template, 880
visual_reporting_template(), *464*, *493*,
496, *508*, *513*

write_mfrm_residual_file, 881
write_mfrm_residual_file(), *447*, *456*,
501, *884*
write_mfrm_subset_file, 883
write_mfrm_subset_file(), *447*, *456*, *501*,
882