Package 'CIMEHR'

October 22, 2026

Type Package

Title Longitudinal EHR Analysis with Informative Visits and Recording

Version 0.2.0

Description Longitudinal outcomes in electronic health records (EHR) are seen only when a patient visits and the outcome is recorded at that visit, and both processes can depend on the patient's latent health. The package fits the CIMEHR estimator for this setting, a three-stage procedure linked by a shared latent factor: a frailty partial likelihood for the visiting process, a pairwise likelihood for the recording (observation) process, and an estimating equation for the outcome. Spline and higher-order Laplace variants are included, together with comparison methods (summary statistics, linear mixed models, Lin-Ying, Liang, pairwise likelihood, joint models, inverse intensity rate ratio weighting and balancing, multiple imputation, linear increments and EHRJoint) behind one common interface. Methods are described in Yang, Shi, and Mukherjee (2026) <doi:10.48550/arXiv.2602.15374>.

License MIT + file LICENSE

Copyright See file inst/COPYRIGHTS.

URL https://github.com/ysph-dsde/CIMEHR

BugReports https://github.com/ysph-dsde/CIMEHR/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports nlme, parallel, Rcpp, splines, stats, tools, utils

LinkingTo Rcpp

Suggests ggplot2 (>= 3.4.0), knitr, mice, pan, rmarkdown

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Collate 'RcppExports.R' 'cimehr_shared.R' 'cimehr.R' 'cimehr_hl.R' 'cimehr_hl_moments.R' 'cimehr_identification.R'

'cimehr_numerics.R' 'cimehr_prediction.R' 'cimehr_spline.R'
'cimehr_spline_hl.R' 'cimehr_zero.R' 'curve_bootstrap.R'
'curve_refit.R' 'dispatch.R' 'ehrjoint.R'
'ehrjoint_prediction.R' 'iirr_balancing_marker.R'
'iirr_balancing_visit.R' 'iirr_weighting_marker.R'
'iirr_weighting_visit.R' 'input_data.R' 'jmvl_g_marker.R'
'jmvl_g_visit.R' 'li_iirr.R' 'li_liang.R' 'li_lmm.R'
'li_reconstruction.R' 'liang_completed_followup.R'
'liang_marker.R' 'liang_trajectory.R' 'liang_visit.R'
'lin_ying_marker.R' 'lin_ying_visit.R' 'lmm.R' 'meanfit.R'
'mi_iirr.R' 'mi_liang.R' 'mi_lmm.R' 'native_adapter.R'
'oa_lmm.R' 'outcome_summary.R' 'pairlik.R' 'registry.R'
'stages.R' 'stages_spline.R' 'summary_max.R' 'summary_mean.R'
'summary_median.R' 'summary_min.R' 'trajectory_api.R'
'va_lmm.R'

Config/roxygen2/version 8.1.1

Author Cheng-Han Yang [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4161-3140>),
Yiren Hou [aut] (ORCID: <https://orcid.org/0009-0005-0422-4268>)

Maintainer Cheng-Han Yang <chenghanyang728@gmail.com>

Repository CRAN

Date/Publication 2026-10-22 13:30:38 UTC

Contents

CIMEHR-package 2
baseline_mean 6
CIMEHR 9
cimehr_example 16
cimehr_fit-methods 17
CIMEHR_spline 20
comparison_methods 26
fit_method 32
meanfit 37
stage_summaries 42

Index **47**

`CIMEHR-package` *Longitudinal Analysis of Electronic Health Records with Informative Visiting and Recording*

Description

Longitudinal outcomes in electronic health records (EHR) are seen only when a patient visits and the outcome is recorded at that visit. Both processes can depend on the patient's latent health. **CIMEHR** fits the CIMEHR estimator for this setting together with comparison methods (summary statistics, linear mixed models, Lin-Ying, Liang, pairwise likelihood, joint models, inverse intensity rate ratio weighting and balancing, multiple imputation, linear increments and EHRJoint). All methods share one interface, `fit_method`, and each method is implemented in its own source file.

Details

Common interface: `fit_method(method, data, ...)` fits one method, where `method` is a method key (the `method` column of `available_comparison_methods()`, for example `"lmm"` or `"cimehr"`) or a label from the `label` column. Methods have no numbers; a numeric `method` is an error. The column names (`id`, `time`, `outcome`, `obs_indicator`, `censor_col`, `entry_col`) and the covariate sets (`visit_covs`, `long_covs`, `outcome_random_covs`, `...`) are common arguments; options specific to one method go in `controls = list(...)`. `method_comparisons` fits several methods to the same data and `cimehr_methods` gives the default comparison set.

Wrapper functions call `fit_method()` for a fixed method or family: `Summary_stat` (`summary_min`, `summary_mean`, `summary_median`, `summary_max`), `LMM` (`lmm`), `Lin_Ying` (`lin_ying_marker`, `lin_ying_visit`), `Liang` (`liang_marker`, `liang_visit`), `Pairwise_likelihood` (`pairlik`), `JMVL_G` (`jmvl_g_marker`, `jmvl_g_visit`), `IIRR` (`iirr_weighting_marker`, `iirr_weighting_visit`), `IIRR_balancing` (`iirr_balancing_marker`, `iirr_balancing_visit`), `Multiple_imputation_IP` (`mi_lmm`, `mi_liang`, `mi_iirr`), `Linear_increment_IP` (`li_lmm`, `li_liang`, `li_iirr`), `EHRJoint` (`ehrjoint`), `CIMEHR_spline` (`cimehr_spline`), `CIMEHR` (`cimehr`), `CIMEHR_HL` (`cimehr_hl`) and `CIMEHR_spline_HL` (`cimehr_spline_hl`).

Methods: Several estimators have a marker version, whose event process is the recorded outcomes (visits with `R = 1`), and a visit version, whose event process is all visits.

Key	Label	Source file
`summary_min`	Summary-Min	'R/summary_min.R'
`summary_mean`	Summary-Mean	'R/summary_mean.R'
`summary_median`	Summary-Median	'R/summary_median.R'
`summary_max`	Summary-Max	'R/summary_max.R'
`lmm`	LMM	'R/lmm.R'
`oa_lmm`	OA-LMM	'R/oa_lmm.R'
`va_lmm`	VA-LMM	'R/va_lmm.R'
`lin_ying_marker`	Lin-Ying (marker)	'R/lin_ying_marker.R'
`liang_marker`	Liang (marker)	'R/liang_marker.R'
`pairlik`	PairLik	'R/pairlik.R'
`jmvl_g_marker`	JMVL-G (marker)	'R/jmvl_g_marker.R'
`iirr_weighting_marker`	IIRR-Weighting (marker)	'R/iirr_weighting_marker.R'
`iirr_balancing_marker`	IIRR-Balancing (marker)	'R/iirr_balancing_marker.R'
`mi_lmm`	MI-LMM	'R/mi_lmm.R'
`mi_liang`	MI-Liang	'R/mi_liang.R'
`mi_iirr`	MI-IIRR	'R/mi_iirr.R'
`li_lmm`	LI-LMM	'R/li_lmm.R'
`li_liang`	LI-Liang	'R/li_liang.R'
`li_iirr`	LI-IIRR	'R/li_iirr.R'

ehrjoint	EHRJoint	'R/ehrjoint.R'
iirr_weighting_visit	IIRR-Weighting (visit)	'R/iirr_weighting_visit.R'
cimehr_spline	CIMEHR spline	'R/cimehr_spline.R'
lin_ying_visit	Lin-Ying (visit)	'R/lin_ying_visit.R'
liang_visit	Liang (visit)	'R/liang_visit.R'
jmvl_g_visit	JMVL-G (visit)	'R/jmvl_g_visit.R'
iirr_balancing_visit	IIRR-Balancing (visit)	'R/iirr_balancing_visit.R'
cimehr	CIMEHR	'R/cimehr.R'
cimehr_hl	CIMEHR-HL	'R/cimehr_hl.R'
cimehr_spline_hl	Spline CIMEHR-HL	'R/cimehr_spline_hl.R'

`summary_min`, `summary_mean`, `summary_median`, `summary_max` least-squares regression of each subject's minimum, mean, median or maximum recorded outcome on the subject-level mean covariates, with HC3 standard errors.

`lmm`, `oa_lmm`, `va_lmm` Gaussian linear mixed model (**nlme**) for the recorded outcomes (`lmm`), adjusted for the cumulative number of recorded outcomes (`oa_lmm`) or of visits (`va_lmm`).

`lin_ying_marker`, `lin_ying_visit` estimating equations for a marginal mean with an unspecified time trend and a proportional-rate event process. Patient-level stacked standard errors propagate the fitted rate regression and empirical profile means, including the nearest-history outcome smoother.

`liang_marker`, `liang_visit` joint model of a proportional-rate gamma-frailty event process and the longitudinal outcome. Patient-level stacked standard errors propagate the rate regression, Breslow baseline, frailty moments, centering weights and outcome equation. At an inactive frailty component, inference conditions on the selected no-frailty branch.

`pairlik` cross-swapped pairwise likelihood of Chen, Ning and Cai.

`jmvl_g_marker`, `jmvl_g_visit` parametric joint model with exponential gap times and bivariate Gaussian random effects for the event process and the outcome.

`iirr_weighting_marker`, `iirr_weighting_visit`, `iirr_balancing_marker`, `iirr_balancing_visit` inverse intensity rate ratio weighting from a proportional-rate event model (`iirr_weighting_marker`, `iirr_weighting_visit`), or entropy-calibrated balancing weights (`iirr_balancing_marker`, `iirr_balancing_visit`).

`mi_lmm`, `mi_liang`, `mi_iirr` multiple imputation of unrecorded outcomes at visits under missing at random (**mice**, `2l.pan`) followed by LMM, Liang or visit-process IIRR, pooled by Rubin's rules. These need the suggested packages **mice** and **pan**.

`li_lmm`, `li_liang`, `li_iirr` linear-increments reconstruction of unrecorded outcomes followed by LMM, Liang or visit-process IIRR. All three default to reverse reconstruction of leading missing outcomes and forward reconstruction after the first observation, using separately estimated increment models. Subjects with visits but no observed outcome are excluded before estimation by default. Patient bootstrap optionally refits the entire procedure.

`ehrjoint` the EHRJoint method of Du, Shi and Mukherjee: a shared visiting frailty and a covariate-based recording model with logit or probit link.

`cimehr` three stages. (1) A proportional-rate visiting model with a shared frailty whose variance is estimated by moments (`gamma.*`, `sigma2_zeta`). (2) A within-patient pairwise probit likelihood for the recording indicators, with fixed effects (`alpha.*`), loadings on the visiting frailty (`delta.*`) and recording random-effect variances (`sigma2_q.*`); the frailty enters through its Laplace (Gaussian) posterior approximation. (3) An estimating equation for the

outcome built from the posterior moments and the fitted recording model, with covariate effects `beta.*` and random-effect link terms `theta.*`. Standard errors come from a patient-level stacked sandwich over all three stages.

`cimehr_hl` CIMEHR with higher-order (third and fourth derivative) corrections to the Laplace posterior moments and an all-stage patient-level stacked sandwich. The approximation and sandwich do not guarantee removal of approximation bias.

`cimehr_spline`, `cimehr_spline_hl` CIMEHR and CIMEHR-HL with a spline baseline time trend and optional covariate-specific time trajectories; fitted curves are available from `predict()` (see `CIMEHR_spline`). The default stacked covariance propagates all three stages, conditional on the chosen spline basis and selected variance face.

Methods `mi_iirr` and `li_iirr` require common follow-up (the same censoring time for every subject). The CIMEHR methods (`cimehr`, `cimehr_hl`, `cimehr_spline`, `cimehr_spline_hl`) require `censor_col`.

Data format: One row per visit with columns `id`, `time`, `Y` (outcome, `NA` when not recorded), `R` (1 if the outcome was recorded at the visit, else 0), subject-level covariates such as `Z` and `X` (numeric and constant within subject), `C` (censoring time, constant within subject) and optionally `E` (entry time). A subject with no visits has exactly one row with `time` = `NA`, `Y` = `NA` and `R` = `NA`. Column names are set through the arguments of `fit_method()`.

Risk sets, entry and censoring: Visits must lie in `[E, C]`. The risk set of the visit process is `(E, C]`: a visit exactly at the entry time is not part of the visit process and is removed automatically; the number removed is returned as `diagnostics$n_visits_at_entry_excluded`. Without `entry_col` there is no left truncation. Delayed entry is supported by `lin_ying_marker`, `lin_ying_visit`, `liang_marker`, `liang_visit`, `jmvl_g_marker`, `jmvl_g_visit`, `iirr_weighting_marker`, `iirr_weighting_visit`, `iirr_balancing_marker`, `iirr_balancing_visit`, `ehrjoint` and the CIMEHR methods.

Results: Every fit is a list of class `"cimehr_fit"` with components `coefficients`, `se`, `vcov` (outcome parameters), `vcov_all` (all stages, when available) and `diagnostics`. `summary_outcome`, `summary_visiting` and `summary_observation` return one table per stage; `extract_coefficient` and `coef_stage` extract estimates; `coef`, `vcov`, `confint`, `summary` and `print` methods are provided.

Inference differs by method. Direct Lin-Ying and Liang methods use `inference = "stacked"`; all four CIMEHR methods also default to stacked inference. Their stacked covariances account for the fitted stages described above, but do not correct model misspecification or approximation bias. Covariance regularity and boundary limitations are recorded in `diagnostics`; a returned standard error alone does not establish valid regular Wald inference. Other methods retain their own model-based, robust, multiple-imputation, conditional or bootstrap inference rules (see `comparison_methods`).

File layout: Each method is implemented in one file, '`R/<key>.R`', named by its method key (see the table above); file names carry no numbers. A named wrapper is defined in the file of a method it fits, for example `Liang()` in '`R/liang_marker.R`'. Shared code is kept in helper files: '`registry.R`' (method registry, keys and `cimehr_methods()`), '`dispatch.R`' (`fit_method()` and `method_comparisons()`), '`input_data.R`' (data checks and entry-visit removal), '`native_adapter.R`' (argument mapping and result standardization), '`outcome_summary.R`' (outcome table and S3 methods), '`stages.R`' and '`stages_spline.R`' (stage tables), '`cimehr_shared.R`' (engine shared by the CIMEHR methods), '`cimehr_hl_moments.R`' and '`cimehr_hl.R`' (higher-order Laplace

moments for `cimehr_hl` and `cimehr_spline_hl`; 'cimehr_hl.R' is also the method file of `cimehr_hl` and holds `CIMEHR_HL()`).

See Also

`fit_method`, `CIMEHR`, `available_comparison_methods`

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

fit <- fit_method("cimehr", dat, visit_covs = c("Z", "X"),
                  obs_fixed_covs = c("Z", "X"), obs_random_covs = "Z",
                  long_covs = c("Z", "X"), outcome_random_covs = "Z",
                  censor_col = "C")
summary_outcome(fit)
summary_visiting(fit)
summary_observation(fit)
# Complete four-method workflow: vignette("cimehr-workflow")
```

---

baseline_mean *Evaluate Baseline and Population Mean Curves*

Description

Data-frame interfaces to the baseline and population mean predictions used by `meanfit`. The data-frame prediction functions use an existing fitted model without refitting it. Direct plotting can optionally request the full patient bootstrap provided by `meanfit()`.

Usage

```
baseline_mean(object, times = NULL, bandwidth = NULL,
              se.fit = FALSE, level = .95)

population_trajectory(object, newdata = NULL, times = NULL,
  bandwidth = NULL, aggregate = TRUE, weights = NULL,
  se.fit = FALSE, level = .95)

## S3 method for class 'cimehr_fit'
predict(object, newdata = NULL, times = NULL,
  type = "population", aggregate = FALSE, weights = NULL,
  bandwidth = NULL, se.fit = FALSE, level = .95, ...)

## S3 method for class 'cimehr_fit'
```

```
plot(x, type = c("trajectory", "baseline"),
  newdata = NULL, times = NULL, bandwidth = NULL, aggregate = is.null(newdata),
  weights = NULL, se.fit = FALSE, level = .95,
  xlab = "Time", ylab = NULL,
  bootstrap = FALSE, B = 50L, seed = NULL, workers = 1L, ...)
```

Arguments

object, x	A supported fitted cimehr_fit; see meanfit for the supported method keys.
newdata	Numeric covariate profiles, a function returning profiles at each requested time, or NULL for the fitted method's stored reference profiles. See meanfit.
times	Finite times within the stored follow-up domain, in the fitted time units. Defaults to 101 equally spaced points in that domain.
bandwidth	Positive local-linear bandwidth in the fitted time units. Defaults to 0.2 times the domain width. Used only for increment-based baseline recovery.
aggregate	Average over profiles at each time. Defaults to TRUE for population_trajectory(), is.null(newdata) for direct plotting, and FALSE for predict().
weights	Nonnegative profile weights with positive sum, for aggregate = TRUE; equal weighting is the default.
se.fit	Request supported analytical curve standard errors and pointwise Wald intervals. The default is FALSE. For direct plotting, bootstrap = TRUE uses bootstrap uncertainty instead. See meanfit for method-specific availability and interpretation. An absent exclusion restriction does not by itself suppress supported CIMEHR curve standard errors. Diagnostic concerns are retained in the returned metadata without suppressing computable intervals.
level	Confidence level strictly between zero and one.
type	For predict(), only "population" is supported. For direct plotting, "trajectory" draws the population mean and "baseline" draws the common baseline mean.
xlab, ylab	Axis labels; see meanfit.
bootstrap, B, seed, workers	For direct plotting only: request an optional patient-cluster bootstrap, set the number of samples (default 50L), an optional seed, and the worker count (default 1L). The default bootstrap = FALSE performs no bootstrap. These arguments are passed to meanfit; they are not arguments of baseline_mean(), population_trajectory(), or predict().
...	Reserved; additional arguments are rejected.

Details

baseline_mean() evaluates $\beta_0(t)$ on the modeled outcome scale. population_trajectory() adds the fitted outcome covariate contribution with random effects averaged to their population mean. predict() is a data-frame interface for separate profiles by default. Use meanfit() to retain an explicit curve object for subsequent summary and plotting.

Spline fits retain their existing predict.cimehr_spline_fit method, including its type = "observed" option and its own defaults; see CIMEHR_spline. Use meanfit(), baseline_mean(), or population_trajectory() for the common mean-curve interface across all supported methods.

For bootstrap intervals, construct a curve with `meanfit(fit, bootstrap = TRUE, B = 50L)` and then call `plot()`. Alternatively, `plot(fit, bootstrap = TRUE, B = 50L)` constructs and plots such a curve directly. Each bootstrap sample resamples whole subjects and refits all stages. Intervals use the original curve plus or minus the normal quantile times the bootstrap standard deviation at each point. The number of finite bootstrap values may differ between points; use a `meanfit` object to inspect the counts and failure diagnostics.

The default reference population, baseline recovery, covariate-path requirements, and uncertainty restrictions are described in `meanfit`. The functions do not automatically transform predictions back from a modeled log outcome.

Value

`baseline_mean()`, `population_trajectory()`, and `predict()` return a data frame containing `time` and `fit`. Separate profile predictions also contain `profile`. Supported uncertainty requests add `se`, `lower`, and `upper`. `plot()` draws a curve and invisibly returns a **ggplot2** object.

See Also

`meanfit`, `fit_method`.

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

fit <- fit_method("cimehr", dat, visit_covs = c("Z", "X"),
                  obs_fixed_covs = c("Z", "X"), obs_random_covs = "Z",
                  long_covs = c("Z", "X"), outcome_random_covs = "Z",
                  censor_col = "C")
times <- seq(0, 60, by = 12)
baseline_mean(fit, times = times)
population_trajectory(fit, times = times)
predict(fit, newdata = data.frame(Z = c(0, 1), X = c(0, 0)), times = times)

## plot() requires the suggested package ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
  file <- tempfile(fileext = ".pdf")
  grDevices::pdf(file)
  plot(fit, type = "baseline", times = times)
  grDevices::dev.off()
  unlink(file)
}
```

CIMEHR *CIMEHR Three-Stage Estimator for Informative Visiting and Recording*

Description

CIMEHR (method key cimehr) fits a longitudinal outcome model to electronic health record data in which both the timing of visits and whether the outcome is recorded at a visit may depend on a shared subject-level latent variable. It combines a shared-frailty visiting model with a Laplace approximation to the latent variable, a within-patient pairwise probit likelihood for recording, and a centred estimating equation for the outcome. CIMEHR_HL (method key cimehr_hl) is the same estimator with higher-order corrected posterior moments of the latent variable.

Usage

```
CIMEHR(data, id_col = "id", time_col = "time", y_col = "Y", r_col = "R",
    censor_col = "C", covars_visit_XV = NULL, covars_outcome_fixed_XY = NULL,
    covars_outcome_random_link_ZY = NULL, covars_obs_fixed_XO = NULL,
    covars_obs_random_link_ZO = NULL, time_precision = 0,
    outcome_fixed_structure = NULL, outcome_random_structure = NULL,
    outcome_random_intercept = TRUE, pairwise_control = list(),
    inference = c("stacked", "none"), derivative_step = 2e-05,
    retain_details = FALSE, entry_col = NULL)

CIMEHR_HL(data, id_col = "id", time_col = "time", y_col = "Y", r_col = "R",
    censor_col = "C", covars_visit_XV = NULL, covars_outcome_fixed_XY = NULL,
    covars_outcome_random_link_ZY = NULL, covars_obs_fixed_XO = NULL,
    covars_obs_random_link_ZO = NULL, time_precision = 0,
    outcome_fixed_structure = NULL, outcome_random_structure = NULL,
    outcome_random_intercept = TRUE, pairwise_control = list(),
    inference = c("stacked", "none"), derivative_step = 2e-05,
    retain_details = FALSE, entry_col = NULL)
```

Arguments

data
: A data frame with one row per visit; see 'Data format'.

id_col, time_col, y_col, r_col, censor_col
: Names of the subject identifier, visit time, outcome, recording indicator (1 if the outcome was recorded at the visit, 0 otherwise) and censoring-time columns.

covars_visit_XV
: Character vector of subject-level covariates X_V of the visiting rate model. No intercept is used (the baseline rate absorbs it). NULL fits the visiting model without covariates.

covars_outcome_fixed_XY
: Character vector of outcome covariates X_Y with fixed coefficients β; at least one is required. Each column must be constant within subject unless it is declared as a time-affine path in outcome_fixed_structure.

`covars_outcome_random_link_ZY`
: Character vector of subject-level columns Z_Y that are multiplied by the latent-variable functional κ_i and enter the outcome model with coefficients θ. Including the name of the time column adds a latent-linked time slope (`theta.time01`). `NULL` adds nothing beyond the intercept controlled by `outcome_random_intercept`.

`covars_obs_fixed_XO`
: Character vector of subject-level covariates X_O of the recording model, added to an intercept that is always included. `NULL` gives an intercept-only design.

`covars_obs_random_link_ZO`
: Character vector of subject-level columns Z_O of the recording model, added to an intercept that is always included. Each column carries a loading δ on the latent variable and a subject-level random-effect variance σ_q^2. `NULL` gives an intercept only.

`time_precision`
: Nonnegative number. If positive, visit, censoring and entry times are rounded to multiples of `time_precision` when the Stage-1 risk sets and baseline grid are formed. `0` (default) uses the exact times. An error is raised if rounding merges two visits of the same subject or moves a visit onto its entry time.

`outcome_fixed_structure`
: `NULL` (default) or a list declaring time-affine outcome covariates; see 'Covariate structures'.

`outcome_random_structure`
: `NULL` (default) or a list declaring the columns of Z_Y; see 'Covariate structures'.

`outcome_random_intercept`
: Logical. If `TRUE` (default), Z_Y contains a constant column (coefficient `theta.Intercept`).

`pairwise_control`
: Named list of Stage-2 optimisation settings: `n_starts` (number of starting values, 1 to 5; default 5), `score_tol` (tolerance on the scaled projected score used to declare convergence; default `1e-10`), `boundary_tol` (variance components below this value are refitted at zero; default `1e-7`), `identification_tol` (relative eigenvalue tolerance of the local identification check; default `1e-7`), `hessian_step` (finite-difference step of the Hessian; default `2e-5`), `iter.max`, `eval.max`, `rel.tol`, `x.tol` (passed to `nlminb`; defaults 250, 600, `1e-10`, `1e-9`) and `optim_control` (a list merged into the `nlminb` control list).

`inference`
: `"stacked"` (default) computes the all-stage stacked sandwich covariance; `"none"` returns point estimates only, with `NA` standard errors.

`derivative_step`
: Relative step, strictly between 0 and 0.1, of the central finite differences used by the stacked sandwich.

`retain_details`
: Logical. If `TRUE`, stage-level intermediate objects are kept in `fit$details` and, with `inference = "stacked"`, the patient-level influence functions in `fit$inference`.

`entry_col`
: Optional name of a subject-level entry-time column E_i for delayed entry. `NULL` (default) means no left truncation.

Details

Data format: One row per visit with the subject identifier, visit time, outcome (`NA` when not recorded), recording indicator, subject-level covariates and a censoring time C_i that is finite and

constant within subject. A subject with no visits has exactly one row with `time`, `Y` and `R` all `NA`. All modelled covariates must be numeric; code factors with `model.matrix` first. X_V, X_O and Z_O must be constant within subject. Visits must lie in $[E_i, C_i]$; the visiting risk set is $(E_i, C_i]$. A visit exactly at E_i is not part of the visit process and is dropped automatically; the number dropped is returned as `diagnostics$n_visits_at_entry_excluded`. At least three subjects are required.

Stage 1: visiting process: Visits of subject i follow a counting process with rate $\eta_i \exp(X_{V,i}^\top \gamma)\, d\Lambda_0(t)$ on $(E_i, C_i]$, where the frailty $\eta_i = \exp(\sigma_\zeta U_i - \sigma_\zeta^2/2)$ is driven by a standard normal latent variable U_i. γ is estimated from the partial-likelihood score, Λ_0 by a Breslow estimator, and σ_ζ^2 by a factorial-moment (method-of-moments) estimator, truncated at zero. The posterior of U_i given the visit count m_i and the expected count is approximated by a Gaussian (Laplace) distribution with the posterior mode as mean and the inverse curvature as variance.

Stage 2: recording process: Given a visit, the outcome is recorded when $X_{O,i}^\top \alpha + Z_{O,i}^\top(\delta U_i + q_i) + e_{ij} > 0$, with $q_i \sim N(0, \mathrm{diag}(\sigma_q^2))$ and $e_{ij} \sim N(0,1)$. With U_i replaced by its Stage-1 posterior, the parameters are estimated by maximising a within-patient pairwise probit likelihood (each pair of visits of a subject weighted $1/(m_i - 1)$; subjects with one visit contribute a marginal probit term) under $\sigma_q^2 \geq 0$. When the estimate of σ_ζ^2 is zero, see 'Zero visiting frailty' below. The fit yields, per subject, the recording probability ω_i and the conditional mean $\kappa_i = E(U_i \mid \mathrm{recorded})$.

Stage 3: outcome: The outcome model is $E(Y_{ij}) = A(t_{ij}) + X_{Y,i}(t_{ij})^\top \beta + \kappa_i Z_{Y,i}(t_{ij})^\top \theta$ with an unspecified baseline mean function $A(t)$. β and θ solve an estimating equation over the recorded outcomes in which A is removed by risk-set centring; each subject at risk is weighted by $\omega_i m_i / \Lambda_{0,i}$, the estimated rate of recorded visits, where $\Lambda_{0,i}$ is the cumulative baseline visiting rate over the subject's risk window. No intercept is estimated.

Inference: With `inference = "stacked"`, the covariance is an all-stage patient-level stacked sandwich that propagates the estimation of γ, Λ_0, σ_ζ^2 and the Stage-2 parameters into the outcome coefficients; derivatives of the posterior moments and of ω_i, κ_i are computed by finite differences with step `derivative_step`. The sandwich targets the Laplace-defined estimator and does not remove approximation bias. When the estimate of σ_ζ^2 or of some σ_q^2 is zero, the covariance is conditional on the selected boundary face. Outcome summary tables and the default `confint` retain the finite SE and checked available interval. `ci_available` reports interval availability separately from `inference_valid`. At zero frailty, successful estimable fits with eligible computed conditional covariance and available intervals receive `inference_status = "conditional_supported"` and `inference_valid = TRUE`. Loading, relevance and collinearity thresholds are descriptive and do not control inference eligibility. Unavailable descriptive diagnostics alone do not force `NA`. Existing fitting, estimability and covariance-availability checks still apply. The intervals are retained, and `regular_inference_valid` remains `FALSE`. `TRUE` denotes support under the fitted model and selected-branch/ face assumptions, not proof of population identification or coverage. The scope and reason are retained: fixed-face intervals omit face-selection uncertainty. `confint(fit, type = "regular")` restricts output to regular intervals. A zero model variance does not imply zero variance for an outcome coefficient; unavailable parameter covariances remain `NA`. Stage-1 visiting-regression and positive frailty-variance intervals use their own covariance checks and do not depend on downstream observation-variance boundaries. The selected-face calculation does not compute influences for fixed boundary variance coordinates, and their missing SEs are not replaced by zero. A finite supplied SE is retained, but an uncalibrated boundary interval is not included in the default limits. An explicit `numerical_wald`

= TRUE request in a visiting or observation summary adds separate numerical limits without establishing boundary calibration or changing inference_valid. The reporting statuses distinguish unavailable inference from a fitting failure; see summary_visiting. When the frailty-variance estimate is positive but within $\sqrt{\log n}$ standard errors of zero, the existing frailty-strength diagnostic flags possible weak identification: diagnostics$frailty_weak_identification is TRUE. This descriptive flag does not suppress an otherwise eligible computed covariance, SE, or Wald interval and does not override the actual fit, estimability and covariance checks. The threshold uses a divergence rate considered for identification-category selection by Andrews and Cheng (2012). This diagnostic does not implement their robust critical values or establish validity of CIMEHR Wald inference under weak identification. Stacked all-stage covariance remains the default; it is not disabled by these reporting distinctions.

Zero visiting frailty: When the estimate of σ_ζ^2 is exactly zero, the posterior of U_i is $N(0,1)$ and the compensation formula is unchanged; a zero frailty variance alone does not remove the recording-outcome link. The check concerns whether the attainable compensation under the original model can be uniquely estimated and separated from the fixed outcome effects after profiling the baseline. It does not require every latent loading to be separately identified. Where an exact quotient is available, only identified coordinates are reported: total observation variances (tau2.<pattern>, for example tau2.(Intercept) for a scalar random intercept or tau2.Z=0 and tau2.Z=1 for binary Z) and compensation coefficients compensation.<Z_Y column>|<Z_O pattern or column> (for example compensation.Intercept|Z=1; the suffix is omitted for a scalar random intercept). These replace individually unidentified delta.*, sigma2_q.* and theta.* coordinates. Saturated Z_O patterns use exact 0/1 indicators. A nonlinear attainable product set is not expanded to an unrestricted linear space.

A unique loading decomposition up to common sign can also support estimation in the original coordinates. The fitted covariance and the outcome design must pass the corresponding identification checks; their numerical rank does not prove regularity at the true parameter. When the three-stage procedure cannot uniquely estimate the compensation, fitting stops with cimehr_zero_frailty_identification_e including reason and stage. The number of random-effect columns is not itself a rejection criterion.

The compensation is estimated only when a relevant exclusion restriction exists: at least one observation-design direction lies outside the outcome design (extra_rank >= 1; see 'Zero-frailty identification diagnostics') and its relevance statistic exceeds $\sqrt{\log n}$. The statistic uses the Stage-2 sandwich covariance of $\hat{\alpha}$ (inverse pairwise Hessian and centered subject scores), which at zero frailty equals its stacked covariance and is computed for every inference option. When this exclusion-relevance criterion is not met, the zero-frailty estimation procedure includes a zero-compensation restriction ($c = 0$), the Stage-3 path also used when the fitted total observation variance is zero. This restriction does not establish the absence of informative observation; the remaining coefficients, stacked SEs and CIs are reported and are interpreted under it, and one cimehr_identification_warning with code compensation_omitted is signalled. A covariate that affects recording but not the outcome can strengthen identification. diagnostics$zero_frailty_compensation records status ("estimated", "omitted_no_relevant_exclusion" or "omitted_zero_total_observation_variance"), reason (for "omitted_no_relevant_exclusion": "no_excluded_direction", "weak_relevance" or "relevance_not_assessed"; otherwise NA), extra_rank and the relevance statistic. The software does not infer an exclusion restriction from covariate names.

Default stacked covariance propagates all remaining identifiable nuisance-stage quantities and their cross-stage covariance, including uncertainty in observation probabilities, compensation

multipliers and centering weights. At zero frailty this calculation is conditional on the selected zero-frailty branch and variance face; it does not assert unconditional or selection-adjusted coverage. Computable covariance, SEs and CIs within that scope are retained.

Zero-frailty identification diagnostics: `diagnostics$identification_diagnostics` contains design and fitted-information diagnostics. Their exclusion and relevance definitions also give the criterion that decides whether the compensation is estimated or restricted to zero (see 'Zero visiting frailty'); apart from that decision, these diagnostics do not change the estimator, covariance, or interval limits. The list contains:

exclusion `extra_rank` is the dimension of the observation fixed-design space outside the intercept and outcome fixed/time/ trajectory modifier space, among subjects with visits. The comparison uses column spaces, not variable names. An additional direction does not establish a valid exclusion restriction.

relevance `statistic` is $\sqrt{b^T V_b^{-1} b/r}$, where b is the fitted observation predictor projected onto the r additional directions and V_b is its covariance obtained from the available stacked covariance of $\hat{\alpha}$. The compensation decision uses the same definition with the Stage-2 sandwich covariance; a value at or below $\sqrt{\log n}$ imposes the zero-compensation restriction. Here n counts eligible independent subjects. This is an Andrews–Cheng-inspired diagnostic, not their calibrated robust critical-value procedure.

compensation `minimum_residual_fraction` is the smallest residual norm fraction of a compensation direction after projection on the outcome design. A value below the descriptive threshold `0.1` gives a `near_collinear` warning. `condition_number` is the condition number of the column-scaled combined design. The ordinary estimator uses its baseline-centered design, whereas the spline estimator explicitly includes baseline and trajectory columns.

loading For a loading decomposition locally unique up to common sign, `third_largest_signal` is the third-largest $|\hat{\delta}_j|/\mathrm{SE}(\hat{\delta}_j)$. A value at or below $\sqrt{\log n}$ gives a `weak_loading` warning. Numerical uniqueness and a large signal do not verify the population loading structure; the `loading_regularity_not_verified` warning remains.

warning_codes, warnings, status Codes and explanations for the applicable diagnostics. An unavailable diagnostic is `not_assessed`. `no_warning_detected` does not certify identification or coverage.

coverage_status `not_established_by_diagnostics`; the checks do not establish nominal confidence-interval coverage.

The warning condition is `cimehr_identification_warning`. It does not suppress a computable covariance, SE, or CI. For the loading, relevance and compensation-collinearity screens, crossing their warning thresholds, or lacking the information to assess them, does not automatically change `inference_valid`; the exclusion-relevance criterion decides only whether the compensation is estimated. The same applies to the `loading_regularity_not_verified` reminder. Conditional support instead requires the existing fit, estimability and covariance-eligibility checks. It remains conditional on the model and selected branch/face assumptions and does not prove population identification or nominal coverage. Visiting parameters retain their own covariance checks. `diagnostics$identification_assessment` stores the overall outcome assessment with `valid`, `status`, `reason` and `scope`. Parameter summaries and curves distinguish numerical availability from conditional inference support; see `summary_outcome`.

Covariate structures: `outcome_fixed_structure` is a list with elements `columns`, `time_origin` (default 0) and `time_scale` (default 1). `columns` is a named list with one data frame per time-varying covariate in `covars_outcome_fixed_XY`, with columns `modifier` (a subject-constant

column name, or "(Intercept)" for 1) and power (0 or 1). The covariate path is the sum over rows of modifier times $((t - \text{origin})/\text{scale})^{\text{power}}$, and it must reproduce the observed column at every visit. Covariates not listed are treated as constant within subject. For example, list(columns = list(age = data.frame(modifier = c("age0", "(Intercept)"), power = c(0, 1)))) declares age = age0 + time.

outcome_random_structure is either a list with static_covs (subject-constant columns of Z_Y), time_covs (columns multiplied by $(t - \text{origin})/\text{scale}$; "(Intercept)" gives a pure time slope named time01), time_origin and time_scale, or a list with columns, time_origin and time_scale in the format of outcome_fixed_structure. When NULL, Z_Y consists of the intercept (if outcome_random_intercept), the non-time columns of covars_outcome_random_link_ZY, and a time slope if the time column is listed there.

CIMEHR-HL: CIMEHR_HL replaces the Laplace posterior mean and variance of U_i by moments corrected with the third and fourth derivatives of the log posterior density (Gaussian moment closure). Stages 2 and 3 and the sandwich are unchanged. The corrected moments remain a local approximation; the sandwich does not remove approximation bias.

CIMEHR(...) and CIMEHR_HL(...) call fit_method with method "cimehr" and "cimehr_hl". Through fit_method, the options time_precision, pairwise_control, inference, derivative_step and retain_details are passed in controls. Note that fit_method uses the visiting covariates as the default recording covariates, whereas these wrappers use an intercept-only recording design when covars_obs_fixed_X0 is NULL.

Value

An object of class c("cimehr_fit", "list") (CIMEHR) or c("cimehr_hl_fit", "cimehr_fit", "list") (CIMEHR_HL), a list with components including

method_key, method, estimand
: Method key ("cimehr" or "cimehr_hl"), label ("CIMEHR" or "CIMEHR-HL") and a description of the target.

coefficients
: Named vector of all estimates: outcome coefficients beta.* and theta.*, visiting coefficients gamma.*, recording parameters alpha.*, delta.* and sigma2_q.*, and the frailty variance sigma2_zeta. At a zero frailty variance compensation.* and tau2.* replace theta.*, delta.* and sigma2_q.*; see 'Zero visiting frailty'.

se
: Standard errors matching coefficients (NA when inference = "none" or unavailable at a boundary).

vcov
: Covariance matrix of the outcome coefficients (beta.*, theta.*).

vcov_all
: Joint stacked covariance of all coefficients, or NULL when inference = "none". Rows and columns of boundary parameters are NA.

vcov_type, variance_type, variance_label
: Description of the covariance.

fit
: The native fit, with components visiting_process (γ, baseline increments Lambda0_hat, σ^2_ζ, posterior means mu_Ui and variances sigma2_Ui), observation_process (α, δ, σ^2_q and their covariance), longitudinal_outcome (β, θ, ω_i, κ_i, estimated baseline $A(t)$), inference (stacked sandwich) and, with retain_details = TRUE, details.

`diagnostics` List including ok, `converged`, `identified`, `regular`, `frailty_boundary`, `observation_variance_bou` `inference_scope`, `n_subjects`, `n_visits`, `n_outcomes`, `n_visits_at_entry_excluded`, optimiser diagnostics (`visiting`, `pairwise`), `warnings` and `elapsed_seconds`; at a zero frailty variance also `compensation_identification`, `identification_diagnostics`, `identification_assessment`, `identification_status` and `identification_warning`.

`parameter_table`
Outcome coefficient table as returned by `summary_outcome`.

`assumptions`, `call`
Model assumptions and the matched call.

The object also records `native_base_method = "cimehr"` and `variant`, which is `"current"` for `CIMEHR` and `"higher_order_gaussian_moments"` for `CIMEHR_HL`. Methods for `coef`, `vcov` (`all = TRUE` gives `vcov_all`), `confint`, `summary` and `print` are available (see `coef.cimehr_fit`); `summary_visiting` and `summary_observation` tabulate the Stage-1 and Stage-2 parameters.

References

Andrews, D. W. K. and Cheng, X. (2012). Estimation and inference with weak, semi-strong, and strong identification. *Econometrica*, **80**, 2153–2211. doi:10.3982/ECTA9456.

See Also

`CIMEHR_spline` for the spline-baseline version, `fit_method`, `available_comparison_methods`, `summary_outcome`.

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

fit <- CIMEHR(dat, covars_visit_XV = c("Z", "X"),
              covars_outcome_fixed_XY = c("Z", "X"),
              covars_outcome_random_link_ZY = "Z",
              covars_obs_fixed_XO = c("Z", "X"),
              covars_obs_random_link_ZO = "Z", censor_col = "C")
summary_outcome(fit)
summary_visiting(fit)
summary_observation(fit)
confint(fit)

fit_hl <- CIMEHR_HL(dat, covars_visit_XV = c("Z", "X"),
                    covars_outcome_fixed_XY = c("Z", "X"),
                    covars_outcome_random_link_ZY = "Z",
                    covars_obs_fixed_XO = c("Z", "X"),
                    covars_obs_random_link_ZO = "Z", censor_col = "C")
extract_coefficient(fit_hl, "Z")
```

`cimehr_example` *Simulated Participant and Biomarker Visit Records*

Description

Simulated data generated from the CIMEHR shared-latent-variable model. They illustrate binary and continuous baseline covariates, unrecorded biomarker values at visits, and participants with no visits. The records do not represent real people or clinical measurements.

Usage

```
data(cimehr_example)
```

Format

A list with three components:

participants A data frame with 1,000 rows and columns `id` (integer participant identifier), `Z` (a binary baseline covariate), `X` (a continuous baseline covariate), and `C` (end of follow-up, 60 months for every participant).

visits A data frame with one row per actual visit. Columns are `id`, `time` (months since baseline), `Y` (the biomarker value), and `R` (the recording indicator). The recording indicator is 1 when a biomarker value is observed and 0 otherwise. Unrecorded values are `NA`. Participants with no visits have no rows in this component.

truth The generating parameters and seed. It contains no participant-specific latent variables or unobserved biomarker values.

Details

The data use seed 200000013. Baseline covariates are independent $Z_i \sim \mathrm{Bernoulli}(0.5)$ and $X_i \sim N(0,1)$. For independent $U_i \sim N(0,1)$, visiting frailty is

$$\eta_i = \exp(0.6U_i - 0.6^2/2).$$

Conditional on the covariates and frailty, visits follow a Poisson process with rate $\exp(-1.6 + 0.4Z_i + 0.2X_i)\eta_i$ over 60 months. Both observation and outcome random-effect designs are $(1, Z_i)$.

The observation coefficients satisfy $q_i = (0.2, 0.8)^\top U_i + e_{qi}$, with independent normal residual coefficients, each with variance 0.2. The probit observation predictor is

$$0.2 - 0.4Z_i + q_{0i} + q_{Zi}Z_i.$$

The coefficient of X_i in the observation model is zero. The outcome coefficients satisfy $b_i = (0.5, 0.2)^\top U_i + e_{bi}$, where the normal residual coefficients have covariance

$$\mathrm{Var}(e_{bi}) = \begin{pmatrix} 0.75 & -0.10 \\ -0.10 & 1.96 \end{pmatrix}.$$

Thus the marginal variances of b_{0i} and b_{Zi} are 1 and 2, and their marginal covariance is zero. The outcome is

$$Y_{ij} = -2 + 0.1t_{ij} - 0.5Z_i + 0.5X_i + b_{0i} + b_{Zi}Z_i + \varepsilon_{ij},$$

where measurement errors are independent standard normal draws. The residual observation and outcome vectors are independent of each other, U_i, and the baseline covariates. Measurement errors are independent of these quantities. Conditional on U_i, the covariates, and q_i, recording indicators are independent Bernoulli draws with the stated probit probabilities and are independent of the outcome residuals and measurement errors. The simulated biomarker is a continuous variable in arbitrary units on the modeled scale; no log transformation is required.

Retain every participant when merging the two components. A participant without visits then contributes a placeholder row with missing `time`, `Y`, and `R`. These rows preserve the risk-set information used in visiting-process estimation. A missing biomarker value at a real visit instead has a finite visit time and `R = 0`.

The reproducible generator is installed at `system.file("examples", "simulate_example.R", package = "CIMEHR")`. It uses only base R and the **stats** package.

Examples

```
data(cimehr_example)
participants <- cimehr_example$participants
visits <- cimehr_example$visits
analysis_data <- merge(participants, visits, by = "id", all.x = TRUE)
analysis_data <- analysis_data[order(analysis_data$id, analysis_data$time), ]
sum(is.na(analysis_data$time))
table(visits$R)
```

---

cimehr_fit-methods *Methods for Fitted CIMEHR and Comparison-Method Objects*

Description

Extract coefficients, covariance matrices and confidence intervals from, summarize, and print objects of class `"cimehr_fit"` returned by `fit_method`, the method wrappers and `CIMEHR` and its variants.

Usage

```
## S3 method for class 'cimehr_fit'
coef(object, ...)

## S3 method for class 'cimehr_fit'
vcov(object, all = FALSE, ...)

## S3 method for class 'cimehr_fit'
confint(object, parm, level = 0.95,
    type = c("available", "regular", "conditional"), ...)
```

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

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

Arguments

object, x	An object of class "cimehr_fit".
all	Logical. If FALSE (default), the covariance of the outcome-model coefficients; if TRUE, the all-stage (joint) covariance matrix when the method provides one.
parm	Parameters for which intervals are computed, given as names from the parameter column of summary_outcome(object) or as row numbers of that table. All outcome parameters if missing.
level	Confidence level, strictly between 0 and 1.
type	For confint, "available" (default) returns available numerical intervals. Validity and scope are reported separately, including an explicit covariance-failure status if applicable. "regular" restricts output to regular inference; "conditional" uses an available conditional covariance, also returning regular intervals for regular fits.
...	Not used.

Details

summary, print and confint work on the outcome table produced by summary_outcome. For CIMEHR and CIMEHR-HL (cimehr and cimehr_hl) this table contains only the outcome parameters (beta.*, theta.*, or identified compensation.* coordinates at zero frailty), whereas coef returns every estimated parameter; use coef_stage or summary_visiting and summary_observation for the other stages.

confint uses qt((1 + level) / 2, df) with the df column of the outcome table (a normal quantile when df = Inf), and, by default, returns available numerical intervals with their inference metadata. A variance boundary does not by itself suppress a computable outcome-coefficient interval. ci_available reports that limits are available; inference_valid separately records support within the stated scope. At zero frailty, an eligible computed conditional covariance gives conditional_supported and inference_valid = TRUE, subject to the existing fit and estimability checks. Descriptive loading, relevance and collinearity warnings, including missing descriptive diagnostics, do not change eligibility or remove available limits. Conditional support assumes the fitted model and selected branch/variance face; it is not a proof of coverage or population identification. regular_inference_valid separately identifies regular inference. A finite SE is reported by se_available. The stage summaries can additionally report numerical Wald limits in separate columns on an explicit numerical_wald = TRUE request; those limits do not change default intervals or make inference_valid true. See summary_visiting. If an object explicitly records a covariance failure, any existing finite numerical limits are retained but are marked inference_status = "covariance_failure" with inference_valid = FALSE; they are not certified regular intervals. Finite numerical values do not override a recorded failure.

When the default output includes conditional intervals or when type = "conditional" is requested, the returned matrix carries inference_type, interval_scope, inference_scope and inference_reason

attributes. Zero-frailty output also retains inference-validity and identification-diagnostic metadata. At a selected variance boundary, fixed-face intervals hold that branch or face fixed and omit uncertainty from selecting it. Their availability does not assert selection-adjusted coverage. `type = "regular"` retains only regular intervals. Nonfinite, unidentified or otherwise unavailable parameter covariances remain `NA`; no interval is fabricated for a zero variance component without an available variance estimate.

With `all = TRUE`, `vcov` returns the first available of `object$vcov_all`, `object$inference$vcov_all` and `object$stacked_sandwich$vcov_joint`.

Value

`coef`: the named numeric vector `object$coefficients`.

`vcov`: with `all = FALSE`, `object$vcov`, the covariance matrix of the outcome-model coefficients; with `all = TRUE`, the all-stage covariance matrix, or `NULL` if the method does not provide one.

`confint`: a matrix with columns `lower` and `upper` and one row per selected outcome parameter.

`summary`: a list with components `method` (method label), `coefficients` (the outcome table, as from `summary_outcome`), `diagnostics`, `variance_label` (description of the variance estimator) and `assumptions`.

`print`: prints the method label and the outcome table and, when the fit has zero-frailty identification warnings, a line listing their codes (the text is in `diagnostics$identification_warning`); returns `x` invisibly.

See Also

`fit_method`, `summary_outcome`, `comparison_methods`.

Examples

```
## Simulated EHR-type data: one row per visit, 150 subjects
set.seed(2)
one_subject <- function(i) {
  Z <- rbinom(1, 1, 0.5); X <- rnorm(1); C <- runif(1, 5, 10); b <- rnorm(1)
  k <- rpois(1, C * exp(-1 + 0.3 * Z + 0.5 * b))
  if (k == 0)
    return(data.frame(id = i, time = NA, Y = NA, R = NA, Z = Z, X = X, C = C))
  time <- sort(runif(k, 0, C))
  R <- rbinom(k, 1, plogis(0.5 + 0.3 * X))
  Y <- 1 + 0.5 * Z - 0.3 * X + 0.1 * time + b + rnorm(k, sd = 0.5)
  data.frame(id = i, time = time, Y = ifelse(R == 1, Y, NA), R = R,
             Z = Z, X = X, C = C)
}
dat <- do.call(rbind, lapply(1:150, one_subject))

fit <- IIRR(dat, process = "visit", visit_covs = c("Z", "X"),
            long_covs = c("Z", "X"), censor_col = "C")
fit
coef(fit)
vcov(fit)
dim(vcov(fit, all = TRUE))   # outcome and visiting-process parameters
```

```
confint(fit)
confint(fit, parm = "Z", level = 0.90)
s <- summary(fit)
s$variance_label
s$coefficients[, c("parameter", "estimate", "se", "inference_valid")]

## A linear mixed model has no all-stage covariance
fit_lmm <- LMM(dat, visit_covs = c("Z", "X"), long_covs = c("Z", "X"))
is.null(vcov(fit_lmm, all = TRUE))
```

CIMEHR_spline *CIMEHR with a Spline Baseline and Spline Trajectories*

Description

CIMEHR_spline (method key cimehr_spline) is the CIMEHR estimator with a regression-spline baseline mean and optional spline trajectory contrasts for selected covariates, so that the outcome mean over time can be estimated and predicted. Stages 1 and 2 are those of CIMEHR. CIMEHR_spline_HL (method key cimehr_spline_hl) uses the higher-order corrected posterior moments of CIMEHR_HL. The predict method returns fitted mean trajectories.

Usage

```
CIMEHR_spline(data, id_col = "id", time_col = "time", y_col = "Y", r_col = "R",
    censor_col = "C", covars_visit_XV = NULL, covars_outcome_fixed_XY = NULL,
    outcome_fixed_structure = NULL, covars_outcome_random_link_ZY = NULL,
    outcome_random_structure = NULL, outcome_random_intercept = TRUE,
    covars_obs_fixed_XO = NULL, covars_obs_random_link_ZO = NULL,
    trajectory_covars = NULL, spline_type = c("natural", "bspline"),
    baseline_df = 5L, trajectory_df = 3L, baseline_knots = NULL,
    trajectory_knots = NULL, boundary_knots = NULL, reference_time = NULL,
    spline_degree = 3L, stage3_chunk_size = 100000L,
    inference = c("stacked", "conditional", "none"), nuisance_fit = NULL,
   retain_nuisance_fit = FALSE, retain_influence = FALSE, entry_col = NULL, ...)

CIMEHR_spline_HL(data, id_col = "id", time_col = "time", y_col = "Y", r_col = "R",
    censor_col = "C", covars_visit_XV = NULL, covars_outcome_fixed_XY = NULL,
    outcome_fixed_structure = NULL, covars_outcome_random_link_ZY = NULL,
    outcome_random_structure = NULL, outcome_random_intercept = TRUE,
    covars_obs_fixed_XO = NULL, covars_obs_random_link_ZO = NULL,
    trajectory_covars = NULL, spline_type = c("natural", "bspline"),
    baseline_df = 5L, trajectory_df = 3L, baseline_knots = NULL,
    trajectory_knots = NULL, boundary_knots = NULL, reference_time = NULL,
    spline_degree = 3L, stage3_chunk_size = 100000L,
    inference = c("stacked", "conditional", "none"), nuisance_fit = NULL,
   retain_nuisance_fit = FALSE, retain_influence = FALSE, entry_col = NULL, ...)

## S3 method for class 'cimehr_spline_fit'
```

```
predict(object, newdata = NULL, times = NULL,
    type = c("population", "observed"), kappa = NULL, se.fit = TRUE,
    level = 0.95, aggregate = FALSE, weights = NULL, bandwidth = NULL, ...)
```

Arguments

`data`	A data frame with one row per visit, in the format described in CIMEHR.
`id_col`, `time_col`, `y_col`, `r_col`, `censor_col`	Names of the subject identifier, visit time, outcome, recording indicator (1 if recorded, 0 otherwise) and censoring-time columns.
`covars_visit_XV`	Character vector of subject-level covariates X_V of the visiting rate model (no intercept), or `NULL`.
`covars_outcome_fixed_XY`	Character vector of outcome covariates X_Y with fixed coefficients β, or `NULL`. Columns must be constant within subject unless declared in `outcome_fixed_structure`. Do not include a constant column; the baseline spline contains the intercept.
`outcome_fixed_structure`	`NULL` or a list declaring time-affine outcome covariates, in the format described in CIMEHR. Here `columns` must list every covariate of `covars_outcome_fixed_XY`, in the same order (use `data.frame(modifier = "X", power = 0)` for a constant covariate `X`), and `time_origin` and `time_scale` must be given.
`covars_outcome_random_link_ZY`	Character vector of subject-level columns Z_Y multiplied by κ_i, with coefficients θ. Including the name of the time column adds a latent-linked time slope (`theta.time`) in the time scale $(t - t_{ref})/w$, where w is the width of `boundary_knots`.
`outcome_random_structure`	`NULL` or a list declaring the columns of Z_Y, in the format described in CIMEHR; `time_origin` and `time_scale` must be given.
`outcome_random_intercept`	Logical. If `TRUE` (default), Z_Y contains a constant column (`theta.(Intercept)`).
`covars_obs_fixed_XO`, `covars_obs_random_link_ZO`	Character vectors of subject-level covariates X_O and Z_O of the recording model, each added to an intercept; see CIMEHR.
`trajectory_covars`	Subset of `covars_outcome_fixed_XY` whose association with the outcome may change over time. Each receives spline contrasts $S(t) - S(t_{ref})$ multiplied by the covariate, so its `beta` coefficient is the association at `reference_time`. A covariate with a time-varying path in `outcome_fixed_structure` cannot be listed.
`spline_type`	`"natural"` (ns, default) or `"bspline"` (bs).
`baseline_df`, `trajectory_df`	Integer basis dimensions (at least 2) of the baseline and trajectory splines. Ignored when the corresponding knots are supplied.
`baseline_knots`, `trajectory_knots`	Optional internal knots, strictly inside `boundary_knots`.

`boundary_knots` Two increasing boundary knots. Default: the range of 0, the visit times and the censoring times.

`reference_time` Time at which the trajectory contrasts are zero and the origin of the default latent-linked time slope; must lie within `boundary_knots`. Default: the lower boundary knot.

`spline_degree` Polynomial degree of B-splines; used only when `spline_type = "bspline"` (natural splines are cubic).

`stage3_chunk_size` Positive integer; number of recorded outcome rows processed at a time when the Stage-3 design is built (controls memory).

`inference` `"stacked"` (default): all-stage stacked sandwich; `"conditional"`: Stage-3 patient sandwich that holds Stages 1 and 2 fixed; `"none"`: no covariance.

`nuisance_fit` Optional Stages 1-2 fit (class `"cimehr_laplace_nuisance"`) from an earlier fit with `retain_nuisance_fit = TRUE`, found in `fit$fit$nuisance_fit`. It is reused instead of refitting Stages 1 and 2, for example to change the spline settings. The data, the visiting and recording designs, `time_precision`, `pairwise_control` and the method variant (ordinary or HL) must match the original fit, otherwise an error is raised.

`retain_nuisance_fit` Logical; keep the Stages 1-2 fit in `fit$fit$nuisance_fit`.

`retain_influence` Logical; keep patient-level influence functions in `fit$fit$stacked_sandwich` and Stage-3 internals in `fit$fit$stage3`.

`entry_col` Optional name of a subject-level entry-time column for delayed entry; `NULL` means no left truncation.

`...` For `CIMEHR_spline` and `CIMEHR_spline_HL`: the named Stage 1-2 controls `time_precision` (default 0), `pairwise_control` (default `list()`) and `derivative_step` (default `2e-5`), with the meanings given in `CIMEHR`; any other name is an error. For `predict`: additional arguments are rejected for `type = "population"` and ignored for `type = "observed"`.

`object` A fit from `CIMEHR_spline` or `CIMEHR_spline_HL`.

`newdata` Data frame with one row per covariate profile. It must contain the fixed covariates (or their modifiers when `outcome_fixed_structure` is used), the trajectory covariates and, for `type = "observed"`, the random-link covariates. For `type = "population"`, `NULL` uses one stored reference profile per retained subject; a function returning profiles at each requested time is also accepted, as in `meanfit`. For `type = "observed"`, an explicit data frame is required.

`times` Finite numeric prediction times in the fitted time units. For `type = "population"`, times must lie within the stored follow-up domain, and `NULL` selects 101 equally spaced points in that domain. For `type = "observed"`, supply the times explicitly.

`type` `"population"` (default) sets the latent term $\kappa_i Z_Y^\top \theta$ to zero; `"observed"` includes it using `kappa`.

`kappa` For `type = "observed"`: one finite value of κ per row of `newdata`, for example taken from `fit$fit$outcome_functionals$kappa`, or from `fit$fit$outcome_functionals$compens` at a zero frailty variance.

se.fit	Logical; add standard errors and pointwise confidence limits. Requires a fit with a covariance (inference not "none").
level	Confidence level of the pointwise limits.
aggregate	For type = "population", average over profiles at each time. The default FALSE returns one curve per profile, including each stored subject profile when newdata = NULL. With TRUE, equal profile weights are used unless weights is supplied. Not used for type = "observed".
weights	For population predictions with aggregate = TRUE, one finite nonnegative weight per profile, with positive sum. Weights are normalized to sum to one. Not used for type = "observed".
bandwidth	Accepted for the common population-trajectory interface; if supplied, must be positive and finite. The fitted spline basis is evaluated directly, so spline predictions do not use smoothing. Not used for type = "observed".

Details

Stages 1 and 2 (shared-frailty visiting model with Laplace posterior for the latent variable U_i; pairwise probit recording model giving ω_i and κ_i) are described in CIMEHR. Stage 3 fits

$$E(Y_{ij}) = B(t_{ij})^\top b + X_{Y,i}(t_{ij})^\top \beta + \sum_g G_{ig}\{S(t_{ij}) - S(t_{ref})\}^\top \phi_g + \kappa_i Z_{Y,i}(t_{ij})^\top \theta$$

by least squares over all recorded outcomes, where B is the baseline basis (with intercept; coefficients baseline.S0.k), G are the trajectory_covars and S is the trajectory basis without intercept (coefficients trajectory.<covariate>.Sc.k).

Choosing the spline: The baseline B and the trajectory basis S are chosen by the user. spline_type selects natural cubic splines (ns) or B-splines (bs of degree spline_degree); baseline_df and trajectory_df set the number of basis functions, with interior knots at quantiles of the observed visit times; baseline_knots and trajectory_knots place the interior knots explicitly (the corresponding df is then ignored); boundary_knots fixes the boundary knots, which also set the time scale w of random slopes in time. Through fit_method the same arguments are given in controls, for example fit_method("cimehr_spline", dat, ..., controls = list(spline_type = "bspline", spline_degree = 2L, baseline_knots = c(10, 20, 30))).

The basis dimensions are fixed; unless boundary_knots, baseline_knots and (with trajectory covariates) trajectory_knots are all supplied, inference is conditional on the knots selected from the observed visit times.

With inference = "stacked" the sandwich propagates the estimation of Stages 1 and 2 through κ_i. At a variance boundary (estimated σ^2_ζ or some σ^2_q equal to zero) the covariance is conditional on the selected face and diagnostics$regular is FALSE. Summary tables distinguish a finite reported SE, available intervals and regular inference. The default confint retains a checked available fixed-face interval and its scope; that interval omits face-selection uncertainty. confint(fit, type = "regular") selects regular intervals only. Stage-1 summaries use their own covariance checks rather than downstream variance-face regularity. At a zero estimate of σ^2_ζ, the compensation coordinates and identification checks in CIMEHR apply (see its 'Zero visiting frailty'). The column-space, relevance, compensation-collinearity and loading diagnostics in CIMEHR also apply. Diagnostic warnings retain computable covariance, SEs and interval limits. Successful estimable fits with

eligible computed conditional covariance give `inference_status = "conditional_supported"` and `inference_valid = TRUE` under the fitted model and selected-face assumptions. Apart from the compensation decision described in CIMEHR, loading, relevance and collinearity screens remain descriptive; their warnings or unavailability do not control eligibility. Existing fit and covariance checks still apply; regular inference remains distinct. No CI, curve SE, or bootstrap interval is suppressed solely because an exclusion restriction is absent. A covariate-name comparison does not determine inference availability. Zero-frailty nuisance fits retained by earlier versions must be refitted for the current identification checks.

Observed-conditional predictions use the saved compensation basis and check whether the fitted model uniquely determines compensation at the requested covariates. A new observation random-link pattern is not itself a reason to stop. Predictions whose compensation can differ between equivalent fitted decompositions or coefficient representations stop with `cimehr_zero_frailty_identification_error` and reason `compensation_not_identified_by_fitted_design`. For example, compensation at `Z = 0.5` need not be determined by a fit that observed only binary `Z`. Population predictions do not include this observation-selection compensation term. Warnings raised during fitting (for example for `inference = "conditional"`) are stored in `diagnostics$warnings`. Visits exactly at the entry time are excluded as for CIMEHR.

`CIMEHR_spline_HL` differs only in the posterior moments of U_i; see CIMEHR_HL. The corrected moments remain a local approximation; the sandwich does not remove approximation bias.

Value

`CIMEHR_spline` returns an object of class `c("cimehr_spline_fit", "cimehr_fit", "list")`; `CIMEHR_spline_HL` returns class `c("cimehr_hl_fit", "cimehr_spline_fit", "cimehr_fit", "list")`. Both record `native_base_method = "cimehr_spline"` and `variant` (`"current"` or `"higher_order_gaussian_moments"`). Components include

`method_key`, `method`, `estimand`
: Method key (`"cimehr_spline"` or `"cimehr_spline_hl"`), label and target description.

`coefficients`, `se`
: Stage-3 estimates and standard errors (`baseline.*`, `beta.*`, `trajectory.*`, `theta.*`, or `compensation.*` at a zero frailty variance); `se` is `NULL` when `inference = "none"`.

`beta`, `trajectory`, `baseline_spline`, `theta`
: The coefficient blocks.

`vcov`
: Covariance of `coefficients` for the chosen `inference` (`NULL` for `"none"`).

`vcov_all`
: With `inference = "stacked"`, the joint covariance of `gamma.*`, `sigma2_zeta`, the recording parameters and the Stage-3 coefficients; otherwise `NULL`.

`vcov_type`, `variance_type`, `variance_label`
: Description of the covariance.

`fit`
: Native fit with `spline_spec` (bases, knots, `reference_time`), `nuisance_summary` (Stage-1 and Stage-2 estimates), `outcome_functionals` (ω_i and κ_i by subject), `stacked_sandwich`, `vcov_conditional`, `model_info` and, if requested, `nuisance_fit` and `stage3`.

`diagnostics`
: List including `ok`, `converged`, `identified`, `regular`, `conditional_inference_valid`, `frailty_boundary`, `observation_variance_boundary`, `inference_scope`, `stage3_design_rank`,

stage3_scaled_condition_number, nuisance_fit_reused, n_subjects, n_visits, n_outcomes, n_visits_at_entry_excluded and warnings; zero-frailty fits also include identification_diagnostics, identification_assessment, identification_status and identification_warning.

parameter_table, assumptions, call
: Outcome table (as from summary_outcome), assumptions and the matched call.

summary_visiting and summary_observation tabulate the Stage-1 and Stage-2 parameters.

predict returns a data frame with columns time and fit, plus profile for separate profile predictions. With se.fit = TRUE, it adds se, lower and upper (normal-based pointwise limits). For an aggregated population curve, the weighted prediction design is formed before applying the joint coefficient covariance. Population predictions use the common interface described in meanfit. Observed predictions retain attributes trajectory_type, uncertainty_scope, inference_scope and regular; their uncertainty is conditional on the supplied kappa. inference_scope describes the covariance; when the fit's inference assessment (at zero frailty, or after a recorded fit or covariance failure) states a different scope, that scope is in attribute inference_assessment_scope.

See Also

CIMEHR, fit_method (methods "cimehr_spline" and "cimehr_spline_hl"), summary_outcome.

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

fit <- CIMEHR_spline(dat, covars_visit_XV = c("Z", "X"),
                     covars_outcome_fixed_XY = c("Z", "X"),
                     covars_outcome_random_link_ZY = "Z",
                     covars_obs_fixed_XO = c("Z", "X"),
                     covars_obs_random_link_ZO = "Z", censor_col = "C",
                     baseline_df = 5L, trajectory_covars = NULL,
                     boundary_knots = c(0, 60), reference_time = 0)
summary_outcome(fit)
profiles <- data.frame(Z = c(0, 1), X = c(0, 0))
predict(fit, newdata = profiles, times = c(12, 36, 60))
curve <- meanfit(fit, newdata = profiles, times = seq(0, 60, by = 12),
                 se.fit = TRUE)
summary(curve)

fit_hl <- CIMEHR_spline_HL(dat, covars_visit_XV = c("Z", "X"),
                           covars_outcome_fixed_XY = c("Z", "X"),
                           covars_outcome_random_link_ZY = "Z",
                           covars_obs_fixed_XO = c("Z", "X"),
                           covars_obs_random_link_ZO = "Z", censor_col = "C",
                           baseline_df = 5L, trajectory_covars = NULL,
                           boundary_knots = c(0, 60), reference_time = 0)
```

```
extract_coefficient(fit_hl, "Z")
```

`comparison_methods` *Named Wrappers for the Comparison Methods*

Description

Convenience functions that fit one of the comparison methods by name. Each wrapper selects a method key from its own arguments and then calls `fit_method(method, data, ...)`, so every result is a `"cimehr_fit"` object with the same structure as a direct `fit_method` call.

Usage

```
Summary_stat(data, stat = c("mean", "median", "min", "max"), ...)

LMM(data, ...)
Linear_mixed_model(data, ...)

Lin_Ying(data, process = c("marker", "visit"), ...)

Liang(data, process = c("marker", "visit"), ...)
Joint_modeling_visiting_and_longitudinal_Liang(data,
    process = c("marker", "visit"), ...)

Pairwise_likelihood(data, ...)

JMVL_G(data, process = c("marker", "visit"), ...)

IIRR(data, process = c("marker", "visit"), ...)
Inverse_intensity_rate_ratio_weighting(data, process = c("marker", "visit"), ...)

IIRR_balancing(data, process = c("marker", "visit"), ...)
Inverse_intensity_rate_ratio_balancing(data, process = c("marker", "visit"), ...)

Multiple_imputation_IP(data, ip_method = c("LMM", "Liang", "IIRR"), ...)

Linear_increment_IP(data, ip_method = c("LMM", "Liang", "IIRR"), ...)

EHRJoint(data, ...)
```

Arguments

`data`	A `data.frame` with one row per visit (see `fit_method` for the required layout).
`stat`	Subject-level summary of the recorded outcomes used by `Summary_stat`: `"mean"` (default), `"median"`, `"min"` or `"max"`.

process	Event process that drives the method: "marker" uses the recorded-marker process (visits with R = 1); "visit" uses the clinic-visit process (all visits). The default is "marker".
ip_method	Analysis applied after imputation (Multiple_imputation_IP) or linear increments (Linear_increment_IP): "LMM" (default), "Liang" or "IIRR".
...	Further arguments passed to fit_method (everything except method and data): the column names id, time, outcome, obs_indicator, censor_col and entry_col; the covariate sets visit_covs, long_covs and outcome_random_covs; outcome_random_structure, outcome_random_intercept, seed, controls; and named method-specific controls, which fit_method merges into controls.

Details

The wrappers map to method keys as follows (keys and labels as in available_comparison_methods()):

Wrapper	Selector	Method key
Summary_stat	stat = "min", "mean", "median", "max"	summary_mi
LMM, Linear_mixed_model	–	lmm (LMM)
Lin_Ying	process = "marker", "visit"	lin_ying_m
Liang, Joint_modeling_visiting_and_longitudinal_Liang	process = "marker", "visit"	liang_mark
Pairwise_likelihood	–	pairlik (Pa
JMVL_G	process = "marker", "visit"	jmvl_g_mar
IIRR, Inverse_intensity_rate_ratio_weighting	process = "marker", "visit"	iirr_weigh
IIRR_balancing, Inverse_intensity_rate_ratio_balancing	process = "marker", "visit"	iirr_balan
Multiple_imputation_IP	ip_method = "LMM", "Liang", "IIRR"	mi_lmm, mi_
Linear_increment_IP	ip_method = "LMM", "Liang", "IIRR"	li_lmm, li_
EHRJoint	–	ehrjoint (E

Paired names (for example Liang and Joint_modeling_visiting_and_longitudinal_Liang) are the same function. Lin_Ying(data, process = "visit", ...) is therefore equivalent to fit_method("lin_ying_visi data, ...). oa_lmm (OA-LMM) and va_lmm (VA-LMM) have no wrapper and are fitted with fit_method; the CIMEHR estimators (cimehr, cimehr_hl, cimehr_spline, cimehr_spline_hl) have their own functions CIMEHR, CIMEHR_HL, CIMEHR_spline and CIMEHR_spline_HL.

At an exactly zero estimated visiting log-frailty variance, the four CIMEHR methods retain a standard normal visiting-only posterior and the original observation/outcome compensation formula. The check concerns the attainable compensation space, not separate identification of every loading and residual variance. There is no rejection based only on the number of observation random-effect columns or the absence of a covariate excluded from the outcome model. When the exclusion-relevance criterion of CIMEHR is not met, the zero-frailty estimation procedure includes a zero-compensation restriction ($c = 0$); this does not establish the absence of informative observation, and the estimates and SEs are interpreted under that restriction. The software does not infer an exclusion restriction from covariate names or use those names to determine interval availability. See 'Zero visiting frailty' in CIMEHR.

Exact total-variance and compensation coordinates are used when supported by the original model. A scalar observation random intercept can retain product coefficients without separately estimating its shared loading. Saturated observation groups use group coordinates only when the original

outcome random-effect design supports the same attainable compensation space. A nonlinear attainable set is not replaced by a larger unrestricted linear span. The compensation must also be separable from fixed outcome effects after profiling the baseline. Otherwise fitting stops with a `cimehr_zero_frailty_identification_error` and a specific reason, rather than returning estimates from an arbitrary latent decomposition. A locally unique loading decomposition up to common sign is not rejected solely because it uses the original loading coordinates. Numerical rank at a fitted covariance does not prove regularity at the true parameter.

Default stacked covariance propagates the remaining identifiable visiting, observation, and outcome quantities, including cross-stage covariance and the estimated probabilities, compensation multipliers, and centering weights. This inference is conditional on the selected zero-frailty branch and variance face; it does not assert unconditional or selection-adjusted coverage. Individual latent parameters that cannot be separated are not reported as identified. The `zero_frailty_compensation` component and diagnostics record the compensation coordinates and identification scope. Numerical rank at a fitted value does not prove regularity at the true parameter.

The covariate arguments of `fit_method` default to `NULL` (no covariates), so `visit_covs` (visit-process covariates) and `long_covs` (outcome-model covariates) should normally be supplied. `outcome_random_covs` gives the random-effect (or random-link) covariates for the methods that model them: `lmm`, `liang_marker`, `liang_visit`, the multiple-imputation methods (`mi_lmm`, `mi_liang`, `mi_iirr`), `li_lmm`, `li_liang` and `ehrjoint`.

For `EHRJoint`, `obs_fixed_covs` specifies the recording-model covariates separately from `visit_covs` and `long_covs`. These predictors must be numeric and constant within subject. Its default is `visit_covs`, after any `controls` override; `character()` gives an intercept-only recording model. The stacked covariance uses this recording design and propagates recording-parameter uncertainty into the outcome estimates.

Method-specific controls are given either in `controls = list(...)` or directly as named arguments; names that the selected method does not accept give an error. The available controls (with defaults) are:

`Lin_Ying` `optimizer_control = list(maxit = 1000L)`, `inference = "stacked"`, `return_subject_scores = FALSE`.

`Liang` `optimizer_control = list(maxit = 1000L)`, `estimate_intercept = FALSE`, `baseline_formula = ~1`, `inference = "stacked"`, `retain_influence = FALSE`.

`Pairwise_likelihood` `time_formula = ~time` (a one-sided formula in the time column such as `~time + I(time^2)`, or `NULL`), `max_subject_pairs = 20000L`, `optimizer_control = list(maxit = 2000L)`; all cross observation pairs of each sampled subject pair are used, and the pair sampling seed is `fit_method`'s `seed`.

`JMVL_G` `include_time = TRUE`, `gh_order = 9L`, `optimizer_control = list(maxit = 1500L, reltol = 1e-9)`.

`IIRR` `include_time = TRUE`, `optimizer_control = list(maxit = 1000L)`, `return_subject_scores = FALSE`.

`IIRR_balancing` `include_time = TRUE`, `balance_time_interactions = TRUE`, `calibration_tol = 1e-8`, `optimizer_control = list(maxit = 2000L)`, `return_subject_scores = FALSE`.

`Multiple_imputation_IP` `m = 5L` imputations, `maxit = 5L`, `keep_completed = FALSE`; for `"LMM"` also `lmm_method = c("REML", "ML")` and `optimizer = c("nlminb", "optim")`; for `"Liang"` also `sigma_floor = 1e-8`, `estimate_intercept = FALSE`, `baseline_formula = ~1`. The imputation seed is `fit_method`'s `seed`. When calling `fit_method()` directly, give the number

of imputations as `controls = list(m = 5)` or name the first argument (`method = "mi_lmm"`); otherwise R matches `m` to `method`.

`Linear_increment_IP` `li_covariance = c("randomwalk", "rate_independence")`, `inference = c("conditional", "patient_bootstrap")`, `bootstrap_reps = 200L`, `keep_completed = FALSE`, `leading_imputation = c("reverse_li", "error")`, and `unobserved_subjects = c("exclude", "error")`; for `"LMM"` also `lmm_method` and `optimizer`; for `"Liang"` also `sigma_floor`, `estimate_intercept`, and `baseline_formula`. The bootstrap seed is `fit_method`'s `seed`.

`EHRJoint` `obs_link = c("logit", "probit")`, `sigma_floor = 1e-6`, `retain_influence = FALSE`, `rank_tol = 1e-9`.

`Summary_stat` and `LMM` have no further controls; `LMM` uses `outcome_random_structure` and `outcome_random_intercept`.

Both marker and visit versions of `Lin_Ying` and `Liang` use `inference = "stacked"`; this is the only supported inference setting for these four methods. The point estimators are unchanged by this covariance calculation. Lin-Ying's patient-level sandwich propagates the fitted rate regression and both empirical profile means, including the nearest-history outcome smoother. Validity still requires the nearest-history mean assumptions and adequate within-subject support. Liang's patient-level sandwich propagates the rate regression, Breslow baseline, frailty moments, centering weights and outcome equation. With `estimate_intercept = TRUE`, Liang inference conditions on the specified outcome baseline basis. At an inactive frailty component, inference conditions on the selected no-frailty branch; it does not include branch-selection uncertainty, and regular Wald inference for the boundary variance is unavailable.

The primary outcome covariance is returned in `vcov`; `vcov(fit, all = TRUE)` also gives the available joint parameter covariance. `return_subject_scores = TRUE` for Lin-Ying and `retain_influence = TRUE` for Liang retain additional patient-level quantities for inspection. These inference settings apply to the direct Lin-Ying and Liang methods. Multiple-imputation and linear-increment methods retain their separately documented inference rules.

For all three `Linear_increment_IP()` methods, the default `leading_imputation = "reverse_li"` reconstructs outcomes before the first observed outcome using a separately fitted reverse increment-rate regression. Only original observed adjacent pairs train that model; its predictors include the next outcome and requested visit/outcome covariates. LI-LMM uses original covariates and random-structure modifiers, avoiding generated random-effect basis columns; LI-IIRR uses the union of visit, outcome and random-link covariates. LI-Liang additionally includes its visit-count frailty terms. It recurses backward only over the leading missing run. The subsequent forward reconstruction is unchanged. Use `leading_imputation = "error"` to reject missing starting outcomes. No reverse model is fitted if leading reconstruction is unnecessary.

The reverse procedure is a derived extension requiring a correctly specified reverse conditional-mean model and representative observed pairs conditional on the visit process. These assumptions do not follow from forward LI alone. Coefficients are pooled across time and visit gaps; no additional time basis is fitted for either increment model.

The default `unobserved_subjects = "exclude"` excludes a whole person with one or more actual visits (finite time) and no originally observed outcome before either LI model or the downstream analysis is fitted. An observed outcome requires original `R = 1` and finite `Y`. `unobserved_subjects = "error"` retains the rejection policy. Public input validation still applies before this eligibility rule. Zero-visit placeholders and people with one observed visit remain. Multiple-visit subjects with at least one observed outcome remain eligible for forward/reverse reconstruction.

Every actual visit of a retained person must have a finite outcome before fitting LMM, Liang or IIRR; this includes both Liang baseline modes. Insufficient or unidentified increment regressions, unresolved outcomes and invalid covariate designs cause errors. Original `R` values and all visits of retained people are preserved; all applicable visit, frailty, centering and outcome equations use the same cohort. Exclusion changes the analysis population and may change the visit/frailty distribution. LI and downstream model assumptions must hold in the retained cohort; this policy does not establish unbiasedness for the original input cohort. Diagnostics record `unobserved_subjects`, `n_subjects_input`, `n_visits_input`, `n_subjects_excluded_unobserved`, `n_visits_excluded_unobserved`, `analysis_population`, and `bootstrap_population`, as well as forward/backward imputation counts. Exclusions are reported as counts without subject identifiers.

All three LI methods default to `inference = "conditional"`, which conditions on LI reconstruction; LI-Liang also conditions on its Stage 1 estimates. Explicitly choosing `inference = "patient_bootstrap"` refits both required increment regressions and the downstream analysis in subject bootstrap samples drawn from the retained cohort, conditional on eligibility. LI-Liang additionally re-estimates its visit-count frailty terms. Bootstrap includes their estimation uncertainty, but does not correct selection bias or validate the model or missing-data assumptions. The same finite-outcome requirement applies to each refit.

Delayed entry (`entry_col` with a nonzero entry time) is supported by `lin_ying_marker`, `lin_ying_visit`, `liang_marker`, `liang_visit`, `jmvl_g_marker`, `jmvl_g_visit`, `iirr_weighting_marker`, `iirr_weighting_visit`, `iirr_balancing_marker`, `iirr_balancing_visit` and `ehrjoint`; the other wrapped methods stop with an error. `mi_liang` and `li_liang` support subject-specific follow-up through the ordinary Liang visiting-process estimator and individual risk sets. `mi_iirr` and `li_iirr` use a common follow-up window: when `censor_col` is supplied, the censoring time must be the same for every subject. `mi_lmm`, `mi_liang` and `mi_iirr` require the suggested packages **mice** and **pan**.

Value

An object of class `"cimehr_fit"` as returned by `fit_method`: a list with, among others, `method_key`, `method`, `coefficients` (named numeric vector), `se`, `vcov`, `diagnostics` (including `n_visits_at_entry_excluded`, `elapsed_seconds` and `warnings`), `variance_type`, `variance_label`, `assumptions`, `parameter_table` and `call`. Methods with a visiting or observation model also return nuisance estimates, which `summary_visiting` and `summary_observation` tabulate, and an all-stage covariance where available (`vcov(fit, all = TRUE)`).

See Also

`fit_method`, `available_comparison_methods`, `method_comparisons`, `summary_outcome`, `coef.cimehr_fit`.

Examples

```
## Simulated EHR-type data: one row per visit, 150 subjects
set.seed(1)
one_subject <- function(i) {
  Z <- rbinom(1, 1, 0.5); X <- rnorm(1); C <- runif(1, 5, 10); b <- rnorm(1)
  k <- rpois(1, C * exp(-1 + 0.3 * Z + 0.5 * b))
  if (k == 0)
    return(data.frame(id = i, time = NA, Y = NA, R = NA, Z = Z, X = X, C = C))
  time <- sort(runif(k, 0, C))
  R <- rbinom(k, 1, plogis(0.5 + 0.3 * X))
```

```
  Y <- 1 + 0.5 * Z - 0.3 * X + 0.1 * time + b + rnorm(k, sd = 0.5)
  data.frame(id = i, time = time, Y = ifelse(R == 1, Y, NA), R = R,
             Z = Z, X = X, C = C)
}
dat <- do.call(rbind, lapply(1:150, one_subject))
vc <- c("Z", "X")

Summary_stat(dat, "mean", visit_covs = vc, long_covs = vc)

fit_lmm <- Linear_mixed_model(dat, visit_covs = vc, long_covs = vc,
                              outcome_random_covs = "Z")
coef(fit_lmm)

fit_liang <- Liang(dat, process = "visit", visit_covs = vc, long_covs = vc,
                   outcome_random_covs = "Z", censor_col = "C")
fit_liang$method_key
summary_visiting(fit_liang)

fit_lv <- fit_method("liang_visit", dat, visit_covs = vc, long_covs = vc,
                     outcome_random_covs = "Z", censor_col = "C")
all.equal(coef(fit_liang), coef(fit_lv))

coef(Lin_Ying(dat, visit_covs = vc, long_covs = vc, censor_col = "C"))
coef(JMVL_G(dat, visit_covs = vc, long_covs = vc, censor_col = "C"))
coef(Pairwise_likelihood(dat, visit_covs = vc, long_covs = vc))
coef(Inverse_intensity_rate_ratio_weighting(dat, visit_covs = vc, long_covs = vc,
                                            censor_col = "C", include_time = TRUE))
coef(IIRR_balancing(dat, process = "visit", visit_covs = vc, long_covs = vc,
                    censor_col = "C"))
coef(EHRJoint(dat, visit_covs = vc, long_covs = vc, outcome_random_covs = "Z",
              censor_col = "C", controls = list(obs_link = "probit")))
coef(Linear_increment_IP(dat, "LMM", visit_covs = vc, long_covs = vc,
                         outcome_random_covs = "Z"))

## LI-IIRR needs a common censoring time
dat_common <- dat
dat_common$C <- 10
coef(Linear_increment_IP(dat_common, "IIRR", visit_covs = vc, long_covs = vc,
                         censor_col = "C"))

## Multiple imputation requires the suggested packages mice and pan;
## m = 2 imputations only keeps the example fast
if (requireNamespace("mice", quietly = TRUE) &&
    requireNamespace("pan", quietly = TRUE)) {
  fit_mi <- Multiple_imputation_IP(dat, "LMM", visit_covs = vc, long_covs = vc,
                                   outcome_random_covs = "Z", m = 2)
  summary_outcome(fit_mi)
}

## The long names are aliases of the short ones
identical(Linear_mixed_model, LMM)
identical(Joint_modeling_visiting_and_longitudinal_Liang, Liang)
identical(Inverse_intensity_rate_ratio_weighting, IIRR)
```

```
identical(Inverse_intensity_rate_ratio_balancing, IIRR_balancing)
```

`fit_method` *Fit CIMEHR or a Comparison Method Through the Common Interface*

Description

`fit_method()` fits one method, chosen by method key or label, to visit-level electronic health record data with a common set of column and covariate arguments. `method_comparisons()` fits several methods to the same data. `available_comparison_methods()` returns the method registry and `cimehr_methods()` returns a default or complete set of method keys.

Usage

```
fit_method(method, data, id = "id", time = "time", outcome = "Y",
           obs_indicator = "R", visit_covs = NULL, long_covs = NULL,
           outcome_random_covs = NULL, obs_random_covs = outcome_random_covs,
           censor_col = NULL, entry_col = NULL, obs_fixed_covs = visit_covs,
           cimehr_visit_covs = visit_covs, outcome_fixed_structure = NULL,
           outcome_random_structure = NULL, outcome_random_intercept = TRUE,
           focal = if (length(long_covs)) long_covs[1L] else NULL,
           seed = 1L, controls = list(), ...)

method_comparisons(data, methods = cimehr_methods("selected"),
                   controls = list(), ...)

available_comparison_methods()

cimehr_methods(mode = c("selected", "all"))
```

Arguments

`method`	a single method: a method key from the `method` column of `available_comparison_methods()` (for example `"lmm"`, `"liang_visit"`, `"cimehr"`), a label from the `label` column (for example `"LMM"`, `"Liang (visit)"`, `"CIMEHR"`) or one of the aliases `"IIRR-M"` (`iirr_weighting_marker`), `"IIRR-V"` (`iirr_weighting_visit`), `"Liang-M"` (`liang_marker`), `"Liang-V"` (`liang_visit`), `"LI+Liang"` (`li_liang`), `"LI+IIRR"` (`li_iirr`), `"CIMEHR (updated)"` (`cimehr`), `"CIMEHR_spline"` (`cimehr_spline`), `"CIMEHR_HL"` (`cimehr_hl`) and `"CIMEHR_spline_HL"` (`cimehr_spline_hl`). Numeric input is an error.
`data`	a data frame with one row per visit; see 'Data format'.
`id, time, outcome, obs_indicator`	names of the subject identifier, visit time, outcome and recording indicator columns.
`visit_covs`	character vector of subject-level covariates for the visiting (or recorded-marker) event process. It is also the default for `obs_fixed_covs` and `cimehr_visit_covs`.

`long_covs` character vector of covariates with fixed effects in the outcome model. Their coefficients are the main results.

`outcome_random_covs` character vector of covariates that carry outcome random effects in addition to the random intercept, for methods that model outcome random effects. `NULL` means none.

`obs_random_covs` character vector of covariates that link the random effects in the recording (observation) model of the CIMEHR methods (`cimehr`, `cimehr_hl`, `cimehr_spline`, `cimehr_spline_hl`).

`censor_col` name of the censoring-time column (constant within subject). Required by the CIMEHR methods (`cimehr`, `cimehr_hl`, `cimehr_spline`, `cimehr_spline_hl`). When `NULL`, methods with a censoring argument use the largest observed visit time as a common censoring time.

`entry_col` optional name of the entry-time column (constant within subject). When `NULL` there is no left truncation. Only `lin_ying_marker`, `lin_ying_visit`, `liang_marker`, `liang_visit`, `jmvl_g_marker`, `jmvl_g_visit`, `iirr_weighting_marker`, `iirr_weighting_visit`, `iirr_balancing_marker`, `iirr_balancing_visit`, `ehrjoint` and the CIMEHR methods accept delayed entry; the others stop if any entry time differs from 0.

`obs_fixed_covs` character vector of fixed covariates in the recording model of the CIMEHR methods and `ehrjoint`. For EHRJoint these must be subject-level, time-invariant numeric covariates. The default uses `visit_covs`, for EHRJoint after any `controls` override; `character()` specifies an intercept-only recording model. EHRJoint's stacked covariance uses this recording design separately from the visiting design.

`cimehr_visit_covs` character vector of visiting-process covariates used by the CIMEHR methods.

`outcome_fixed_structure` optional list declaring fixed outcome covariates as affine functions of time (CIMEHR methods); see `CIMEHR`.

`outcome_random_structure` optional list describing the outcome random-effect design (`lmm`, `oa_lmm`, `va_lmm`, `mi_lmm`, `li_lmm` and the CIMEHR methods).

`outcome_random_intercept` logical; include an outcome random intercept (`lmm`, `oa_lmm`, `va_lmm`, `mi_lmm`, `li_lmm` and the CIMEHR methods).

`focal` name of the focal outcome coefficient. The summary-statistic and mixed-model methods (`summary_min`, `summary_mean`, `summary_median`, `summary_max`, `lmm`, `oa_lmm`, `va_lmm`) set `diagnostics$ok = FALSE` when it is missing or not finite.

`seed` integer seed for methods that use random numbers: subject-pair sampling in `pairlik`, imputation in `mi_lmm`, `mi_liang` and `mi_iirr` and the optional bootstrap in `li_lmm`, `li_liang` and `li_iirr`. The global random number state is restored afterwards.

`controls` for `fit_method()`, a named list of method-specific arguments passed to the method (for example `list(obs_link = "probit")` for `ehrjoint` or `list(inference`

= "none") for cimehr). Names must be unique, must be arguments of the chosen method, and may not repeat the data or column arguments above. For method_comparisons(), a list of such lists named by the value supplied in methods or by the method key (for example "cimehr" or "liang_marker"); a name equal to the supplied value takes precedence.

... for fit_method(), further method-specific arguments, merged into controls. For method_comparisons(), arguments passed to fit_method() for every method.

methods character vector of method keys, labels or aliases, as for method. Numbers and unknown names are an error for the whole call.

mode "selected" for the default comparison set, "all" for every registered method.

Details

fit_method() checks the data, removes visits that coincide with the entry time, resolves the method, passes the common arguments that the method accepts together with controls, and returns a standardized result. Warnings raised while fitting are collected in diagnostics$warnings and are not printed, except that a fit with zero-frailty identification warnings signals one warning of class cimehr_identification_warning. Methods are identified only by key, label or alias: a numeric method is an error, and so is a string that is neither a key, a label nor an alias.

Fits from the CIMEHR variants, Liang-V, Liang-M, MI+Liang, LI+Liang, and EHRJoint retain input data and evaluated fitting arguments in refit for the optional participant bootstrap in meanfit. Request it with meanfit(fit, bootstrap = TRUE, B = 50L); it is disabled by default and separate from method-specific coefficient inference. The retained data include any zero-visit participant rows supplied to the fit. Older saved fits without this component must be refitted before requesting bootstrap mean curves.

Several estimators come in a marker version, whose event process is the recorded outcomes (visits with R = 1), and a visit version, whose event process is all visits: lin_ying_marker/lin_ying_visit, liang_marker/liang_visit, jmvl_g_marker/jmvl_g_visit, iirr_weighting_marker/iirr_weighting_visit, iirr_balancing_marker/iirr_balancing_visit. The key fixes the process; setting process in controls to the other process is an error. mi_iirr and li_iirr require common follow-up and stop when the censoring time varies between subjects. mi_lmm, mi_liang and mi_iirr need the suggested packages **mice** and **pan**.

li_lmm, li_liang and li_iirr default to leading_imputation = "reverse_li": a separately fitted reverse increment-rate model reconstructs leading missing outcomes from the subject's first observed outcome. It uses only original observed adjacent pairs for fitting, with the next outcome and requested visit/outcome covariates as predictors. LI-LMM uses original covariates and random-structure modifiers instead of generated random-effect basis columns; LI-IIRR uses the union of visit, outcome and random-link covariates. LI-Liang additionally includes its visit-count frailty terms. Forward reconstruction after the first observation is unchanged. Set controls = list(leading_imputation = "error") to reject missing starting outcomes. No reverse model is fitted unless needed.

Reverse reconstruction requires additional reverse conditional-mean and missingness assumptions conditional on the visit process. It is not an algebraic inversion or an automatic consequence of forward LI. Both models pool increment-rate coefficients across time and gaps.

The default controls = list(unobserved_subjects = "exclude") excludes a whole person with one or more actual visits (finite time) and no originally observed outcome before fitting LI or the

downstream analysis. An observed outcome requires original `R = 1` and finite `Y`. Set `controls = list(unobserved_subjects = "error")` to retain the rejection policy. Public input validation still applies before this eligibility rule. Zero-visit placeholders and people with one observed visit are retained; multiple-visit subjects with an observed anchor remain eligible for LI. Every actual visit of every retained person must have a finite outcome before LMM, Liang or IIRR is fitted, including both Liang baseline modes. Insufficient or unidentified increment regressions, unresolved outcomes and invalid covariate designs cause errors. Visits and original `R` values of retained people are preserved across all stages. Selection changes the analysis population and can change the visit/frailty distribution. LI and downstream model assumptions must hold in the retained cohort; this policy does not establish unbiasedness for the original input cohort.

Diagnostics report `unobserved_subjects`, `n_subjects_input`, `n_visits_input`, `n_subjects_excluded_unobserved`, `n_visits_excluded_unobserved`, `analysis_population`, and `bootstrap_population`. Exclusions are reported as counts without subject identifiers.

The default covariance for all three LI methods conditions on LI reconstruction; LI-Liang also conditions on its Stage 1 estimates. Explicitly request `controls = list(inference = "patient_bootstrap")` to refit both required increment regressions and the downstream analysis in each subject bootstrap sample from the retained cohort, conditional on eligibility. LI-Liang additionally re-estimates its visit-count frailty terms. This includes their estimation uncertainty but does not correct selection bias or validate the missing-data assumptions. Every refit must meet the same finite-outcome requirement. Imputation diagnostics report forward and backward counts, the leading-imputation policy and whether a backward model was used.

`method_comparisons()` calls `fit_method()` for each element of `methods`. An error while fitting one method does not stop the others.

`cimehr_methods("selected")` returns `lmm`, `summary_median`, `iirr_weighting_visit`, `liang_visit`, `iirr_weighting_marker`, `liang_marker`, `li_iirr`, `li_liang`, `ehrjoint`, `cimehr`, `cimehr_spline`, `cimehr_hl` and `cimehr_spline_hl`.

Value

`fit_method()` returns a list of class `"cimehr_fit"` (with `"cimehr_spline_fit"` added for `cimehr_spline` and `cimehr_spline_hl` and `"cimehr_hl_fit"` for `cimehr_hl` and `cimehr_spline_hl`). Its main components are

`method_key`, `method`, `estimand`
: the method key, name and a description of its target.

`coefficients`
: named numeric vector of estimates. For `cimehr` and `cimehr_hl` it contains all stages: outcome (`beta.*`, `theta.*`), visiting (`gamma.*`, `sigma2_zeta`) and recording (`alpha.*`, `delta.*`, `sigma2_q.*`) parameters; `cimehr_spline` and `cimehr_spline_hl` also report spline coefficients (`baseline.*`).

`se`, `vcov`
: standard errors and covariance matrix of the outcome parameters.

`covariance_factor`
: for CIMEHR methods, a compact factor of the outcome participant influences, when uncertainty is computed. It supports stable contrast calculations without storing every participant influence.

`vcov_all`
: joint covariance of all estimated stages, for methods that provide one (otherwise absent); also available as `vcov(fit, all = TRUE)`.

variance_type, variance_label
: standardized description of the variance estimator.

diagnostics
: list including ok, warnings, elapsed_seconds, package_version and n_visits_at_entry_excluded, plus method-specific entries.

parameter_table
: outcome-stage table as returned by summary_outcome.

assumptions, call
: working assumptions of the method and the matched call.

Further method-specific components (for example fit, nuisance, inference) may be present.

method_comparisons() returns a list of such fits named by method label. A method that fails is returned as a list of class "cimehr_failure" with components method and error.

available_comparison_methods() returns a data frame with one row per method; columns include method (the method key), label, native_estimand, requires_common_followup, sampling_event_source, se_scope, variance_label, base_method and native_base_method (method keys; for example "cimehr" for cimehr_hl).

cimehr_methods() returns a character vector of method keys.

Data format

One row per visit with columns id, time, Y (outcome, NA when not recorded), R (1 if the outcome was recorded at the visit, else 0), numeric subject-level covariates (constant within subject), C (censoring time) and optionally E (entry time). A subject without visits has exactly one row with time, Y and R equal to NA. Visits must lie in [E, C]. Covariates must be numeric; encode factors with model.matrix.

The risk set of the visit process is (E, C]. A visit exactly at the entry time is not part of the visit process and is dropped before fitting (a subject left without visits becomes a zero-visit row); the number dropped is returned in diagnostics$n_visits_at_entry_excluded.

See Also

summary_outcome, summary_visiting, summary_observation, extract_coefficient; CIMEHR, CIMEHR_spline and the method wrappers such as LMM and Liang.

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

available_comparison_methods()[, c("method", "label")]

fit <- fit_method("cimehr", dat, visit_covs = c("Z", "X"),
                  obs_fixed_covs = c("Z", "X"), obs_random_covs = "Z",
                  long_covs = c("Z", "X"), outcome_random_covs = "Z",
                  censor_col = "C")
```

```
summary_outcome(fit)
dim(vcov(fit, all = TRUE))

fit_lmm <- fit_method("lmm", dat, long_covs = c("Z", "X"),
                      outcome_random_covs = "Z", censor_col = "C")
coef(fit_lmm)
cimehr_methods()

## Fit several methods to the same data in one call
fits <- method_comparisons(dat, methods = c("lmm", "liang_visit"),
                           visit_covs = c("Z", "X"), long_covs = c("Z", "X"),
                           censor_col = "C")
lapply(fits, coef)
```

`meanfit` *Construct and Plot Fitted Longitudinal Mean Curves*

Description

Constructs a fitted mean-curve object from a CIMEHR, Liang, MI/LI+Liang, or EHRJoint model. The curve object can be summarized, converted to a data frame, or plotted. An optional patient bootstrap refits the model to estimate curve uncertainty; it is disabled by default. Once constructed, the curve object can be summarized or plotted without further refitting.

Usage

```
meanfit(object, ...)

## S3 method for class 'cimehr_fit'
meanfit(object, newdata = NULL, times = NULL,
  type = c("population", "baseline"), aggregate = is.null(newdata),
  weights = NULL, bandwidth = NULL, se.fit = FALSE, conf.int = .95,
  bootstrap = FALSE, B = 50L, seed = NULL, workers = 1L, ...)

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

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

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

## S3 method for class 'cimehr_meanfit'
plot(x, conf.int = !is.null(x$conf.int),
  xlab = "Time", ylab = NULL, ...)
```

Arguments

object	For meanfit(), a cimehr_fit returned by fit_method. Supported method keys are cimehr, cimehr_hl, cimehr_spline, cimehr_spline_hl, liang_visit, liang_marker, mi_liang, li_liang, and ehrjoint. For summary(), a cimehr_meanfit object.
x	A cimehr_meanfit object.
newdata	A data frame with one row per covariate profile and the numeric covariates or modifiers required by the fitted outcome design. With NULL, uses one stored reference profile per subject in the population retained by the fitted method. A function of one numeric time may instead return a nonempty profile data frame at each requested time, where the fitted model supports the specified covariate paths. Keep the row ordering consistent across times for separate profile curves. Not used for type = "baseline".
times	Finite prediction times in the fitted time units and within the stored follow-up domain. The default is 101 equally spaced points in that domain. Supplied times are sorted and duplicates removed.
type	"population" for the outcome mean with random effects averaged over their population distribution, or "baseline" for the common baseline mean function $\beta_0(t)$.
aggregate	Average predictions over covariate profiles at each time. The default is TRUE when newdata is NULL, and FALSE when profiles are supplied. With FALSE, returns one curve per profile. Not used for baseline mean curves.
weights	Optional finite, nonnegative profile weights with positive sum, used only when aggregate = TRUE. Provide one weight per profile; weights are normalized to sum to one. The default gives equal weight to each profile. Not used for baseline mean curves.
bandwidth	Positive local-linear smoothing bandwidth in the fitted time units. The default is 0.2 times the width of the stored follow-up domain. Used for increment-based baseline recovery; fitted spline or parametric baseline bases are evaluated directly.
se.fit	Compute analytical curve standard errors and pointwise Wald intervals. The default is FALSE. Currently supported for the CIMEHR spline methods when their joint coefficient covariance is available. Ignored when bootstrap = TRUE, which uses bootstrap standard errors instead. See Details.
conf.int	For meanfit(), a confidence level strictly between zero and one. For plot(), a logical indicating whether to draw intervals already stored in the curve object; defaults to whether intervals were requested when constructing the object.
bootstrap	Use a patient-cluster bootstrap to estimate curve standard errors and pointwise Wald intervals. The default, FALSE, performs no bootstrap. When TRUE, overrides analytical inference requested through se.fit. Requires a fit with stored refitting data and settings.
B	Number of bootstrap samples, an integer of at least two. The default is 50L. Increase this, for example to 200L, to reduce Monte Carlo variability in the bootstrap standard errors.

seed	Optional seed for the bootstrap.
workers	Number of bootstrap workers, a positive integer. The default is 1L. Multicore execution is available on non-Windows systems; Windows uses sequential execution with a warning when more than one worker is requested.
row.names, optional	Standard as.data.frame arguments. row.names, when supplied, replaces the returned row names.
xlab, ylab	Axis labels. The default vertical label is 'Population-average outcome' or 'Baseline mean outcome'.
...	Reserved for S3 methods. Additional arguments to meanfit(), summary(), or plot() are rejected.

Details

For a supplied profile $x(t)$, the population mean is

$$m(t; x) = \beta_0(t) + x(t)^\top \beta$$

with the corresponding fitted spline design used by the spline methods. Random effects are averaged to their population mean; subject-specific empirical Bayes predictions are not used. All predictions remain on the modeled outcome scale, including the log scale when a log-transformed outcome was fitted.

With no newdata, the default averages one profile per subject in the reference population retained by the fitted method. The reference population can differ across methods, particularly after LI reconstruction eligibility exclusions. Supply the same newdata and weights to compare curves standardized to a common population. Stored profiles must be suitable for the requested time path; CIMEHR fits with time-varying modifiers that cannot be represented by stored subject profiles require explicit newdata.

For ordinary CIMEHR and EHRJoint, the baseline is recovered by local-linear smoothing of compensated baseline increments against visiting-baseline increments. The cumulative numerator $A(t) = \int_0^t \beta_0(u)\, d\Lambda_0(u)$ is an intermediate quantity; type = "baseline" returns the recovered $\beta_0(t)$. Liang uses its stored outcome-event normalization or evaluates its fitted parametric baseline basis. MI+Liang averages the baseline curves from its completed-data fits. The CIMEHR spline methods evaluate the fitted baseline spline directly. Local recovery returns NA where there are fewer than three distinct usable event times in the bandwidth or the local design is numerically singular.

For CIMEHR spline curves, se.fit = TRUE uses the stored joint covariance of baseline-spline and outcome coefficients. For a prediction design row $d(t)$, the variance is $d(t)Vd(t)^\top$. For an aggregated curve, the weighted prediction design is formed before applying this covariance. Intervals are pointwise, conditional on the specified profiles, weights, and fitted spline basis.

Without a bootstrap, ordinary CIMEHR, Liang, MI/LI+Liang, and EHRJoint provide point curves through this interface. Their outcome-coefficient standard errors alone do not include the uncertainty in baseline recovery. Newly fitted CIMEHR spline models retain an untruncated factor of their participant influences. Analytical curve variances use that factor directly to reduce cancellation between large covariance entries, retaining the fitted conditional or all-stage stacked scope. This numerical calculation does not establish identification or confidence-interval coverage.

Requesting se.fit = TRUE without a supported curve covariance raises an error unless bootstrap = TRUE is also specified.

At zero visiting frailty, the absence of an exclusion restriction does not by itself prevent supported curve standard errors or bootstrap intervals. The fitted compensation must be identified, and the method-specific curve covariance requirements above still apply. See CIMEHR for conditional inference scope at a selected variance boundary. Returned curve metadata distinguish numerical uncertainty availability from conditional inference support; identification warnings do not remove computable limits. At zero frailty, `conditional_supported` gives `inference_valid = TRUE` for otherwise eligible computed uncertainty under the fitted-model and selected-branch/face assumptions. Descriptive identification warnings and missing descriptive diagnostics do not control eligibility. Existing fitting, estimability and uncertainty-availability requirements still apply. An explicitly recorded covariance failure also prevents an analytical positive-frailty curve interval from being reported as valid inference: computed numerical values remain, with failure status and its reason. A full-refit bootstrap uses its own uncertainty calculation, so failure of the parent analytical covariance alone does not invalidate an otherwise available bootstrap interval; the parent failure remains visible in the curve-data attributes `parent_covariance_failed` and `parent_covariance_warning`, and in the inference reason. For zero-frailty CIMEHR output, curve attributes include `ci_available`, `inference_valid`, `inference_status`, `identification_status` and `identification_warning`; the `meanfit` object also retains this metadata at the top level.

With `bootstrap = TRUE`, subjects are sampled with replacement from the stored input data. Each sampled copy receives a new subject identifier and retains its complete record, including a placeholder for a subject without visits. Each bootstrap fit repeats all estimation stages and any MI or LI operations using the stored evaluated settings. Supplied model controls are retained, cached nuisance fits are discarded, and automatic data-dependent defaults are recalculated from the bootstrap data. The method's eligibility rules are applied in each refit. For zero-frailty CIMEHR fits, bootstrap curves retain the `parent_identification_assessment` separately. Their diagnostic support is inherited from the parent fit and does not calibrate bootstrap coverage. The bootstrap `inference_scope` describes refitting all stages, rather than the selected-face analytical covariance used for Wald intervals.

For `newdata = NULL` and `aggregate = TRUE`, each bootstrap curve averages over the reference population retained by that bootstrap fit. With `aggregate = FALSE`, the original fitted reference profiles are held fixed. Explicit `newdata`, including a profile-generating function, is held fixed across bootstrap fits. Bootstrap requests with `newdata = NULL` and non-`NULL` `weights` are rejected; supply explicit profiles and their weights for weighted standardization.

The original prediction grid and smoothing bandwidth are used throughout. At each time and profile, the bootstrap standard error is the sample standard deviation of the finite bootstrap curve estimates, without division by the square root of `B`. The pointwise Wald interval is centered at the original fitted curve:

$$\widehat{m}(t) \ \pm \ z_{(1+\mathrm{conf.int})/2}\, \widehat{\mathrm{SE}}_{\mathrm{boot}}(t).$$

The same calculation applies to a baseline mean curve. Failed fits and nonfinite or unsupported curve points are recorded; failed attempts are not replaced by new samples. A warning reports incomplete bootstrap output. Each curve row reports its finite replicate count in `n_boot`; fewer than two finite values produce `NA` for its standard error and interval, while retaining the original point estimate.

Older fitted objects without the required prediction metadata may need to be refitted with the updated package. Bootstrap additionally requires the stored `refit` data and settings; older objects without this component must be refitted. Plotting requires **ggplot2**; it draws stored intervals by default when available. Use `plot(curve, conf.int = FALSE)` to hide them without refitting.

Value

`meanfit()` returns an object of class `cimehr_meanfit` with components `data`, `time`, `method`, `method_key`, `type`, `aggregate`, `newdata`, `standardization`, `bandwidth`, `conf.int`, `outcome_scale`, and `call`. Its `data` component contains `time` and `fit`, plus `profile` for separate profile curves. Supported uncertainty requests add `se`, `lower`, and `upper`. Bootstrap output also adds `n_boot` to `data` and a `bootstrap` component containing the B-by-curve-row matrix `replicates`, pointwise finite counts `n_success`, the number of `successful_fits`, and failure diagnostics, including the stage and reason for failed fits or incomplete curves.

`summary()` and `as.data.frame()` return this curve data frame. Choose prediction times when calling `meanfit()`. `plot()` draws the curves and invisibly returns a **ggplot2** object, which can be passed to `ggplot2::ggsave()` or modified with additional ggplot layers. `print()` invisibly returns its input.

See Also

`fit_method`, `baseline_mean`, `population_trajectory`.

Examples

```
data("cimehr_example", package = "CIMEHR")
d <- merge(cimehr_example$participants, cimehr_example$visits,
           by = "id", all.x = TRUE)
d <- d[d$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
d <- d[order(d$id, d$time, na.last = TRUE), ]
fit <- fit_method("cimehr", d, visit_covs = c("Z", "X"),
                  long_covs = c("Z", "X"), obs_fixed_covs = c("Z", "X"),
                  outcome_random_covs = "Z", obs_random_covs = "Z",
                  censor_col = "C")
profiles <- data.frame(Z = c(0, 1), X = c(0, 0))
times <- seq(0, 60, length.out = 13)

## Separate curves for two covariate profiles
curve <- meanfit(fit, newdata = profiles, times = times)
summary(curve)

## Baseline mean curve
as.data.frame(meanfit(fit, type = "baseline", times = times))

## Weighted standardization over the two profiles
as.data.frame(meanfit(fit, newdata = profiles, times = times,
                      aggregate = TRUE, weights = c(.4, .6)))

## Participant bootstrap. B = 5 only keeps the example fast; use B >= 50
## in practice.
fit_liang <- fit_method("liang_visit", d, visit_covs = c("Z", "X"),
                        long_covs = c("Z", "X"), outcome_random_covs = "Z",
                        censor_col = "C")
boot <- meanfit(fit_liang, newdata = profiles, bootstrap = TRUE,
                B = 5L, seed = 2026)
summary(boot)
```

```
## Plotting requires the suggested package ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
  file <- tempfile(fileext = ".pdf")
  grDevices::pdf(file)
  plot(curve)
  plot(boot, conf.int = FALSE)
  grDevices::dev.off()
  unlink(file)
}
```

`stage_summaries` *Stage-Specific Parameter Tables*

Description

Tabulate the estimates of a fitted model by stage: the longitudinal outcome model, the visiting (event) process and the observation (recording) process. `coef_stage` returns the estimates of one stage as a named vector and `extract_coefficient` returns the table row of a single parameter.

Usage

```
summary_outcome(object, ...)
summary_visiting(object, numerical_wald = FALSE, ...)
summary_observation(object, numerical_wald = FALSE, ...)

coef_stage(object, stage = c("outcome", "visiting", "observation"))

extract_coefficient(object, term,
    stage = c("outcome", "visiting", "observation"))
```

Arguments

`object`	A fitted object of class `"cimehr_fit"`, as returned by `fit_method`, the method wrappers or `CIMEHR` and its variants.
`...`	Not used.
`numerical_wald`	Logical; default `FALSE`. For visiting and observation summaries, explicitly request separate arithmetic Wald limits from finite SEs of valid point estimates. This is a diagnostic calculation, not a calibrated boundary confidence interval. It does not change default intervals or validity flags.
`stage`	Stage to report: `"outcome"` (default), `"visiting"` or `"observation"`.
`term`	Character string naming one parameter, as in the `parameter` column of the stage table (for example `"Z"`, `"gamma.Z"` or `"alpha.X"`). If it is not found, `paste0("beta.", term)` is tried, so `"Z"` selects `"beta.Z"` in a CIMEHR fit.

Details

summary_outcome(object), summary_visiting(object) and summary_observation(object) are the tables for the stages "outcome", "visiting" and "observation". The stages contain:

outcome The outcome-model coefficients in object$coefficients; for CIMEHR and CIMEHR-HL (cimehr and cimehr_hl) only the beta.* and theta.* entries. This is the table printed by print and returned by summary for a "cimehr_fit" object.

visiting For cimehr and cimehr_hl, the gamma.* coefficients and the frailty variance sigma2_zeta; for the spline methods (cimehr_spline and cimehr_spline_hl), the same parameters taken from the spline fit. For the other methods, the visiting-process coefficients reported by the method (gamma.*) and, when reported, the frailty variance (frailty_variance) and a scalar baseline Lambda.

observation For the CIMEHR methods (cimehr, cimehr_hl, cimehr_spline, cimehr_spline_hl), the recording-model coefficients alpha.*, the random-link coefficients delta.* and the variance components sigma2_q.*. For the other methods, the recording-model coefficients alpha.* when the method fits one (for example ehrjoint).

A stage that the method does not estimate gives a data frame with no rows (and coef_stage gives numeric(0)).

Intervals in the outcome table are 95% intervals based on qt(0.975, df) (normal quantiles when df = Inf); intervals in the visiting and observation tables are 95% normal (Wald) intervals. Use confint for another level of the outcome parameters. By default, intervals use every checked available covariance, including conditional covariance at a variance boundary. confint(object, type = "regular") restricts output to regular inference. Unavailable or unidentified parameter covariances do not acquire intervals; a boundary marker describes the fitted model and does not alone suppress an available outcome interval. If the fitted object explicitly records a covariance failure, existing finite numerical limits are retained with inference_valid = FALSE and inference_status = "covariance_failure", rather than treated as validated inference.

The selected-face CIMEHR sandwich does not estimate the sampling uncertainty of variance coordinates held fixed at zero. Their missing SEs are not replaced by zero and do not indicate failed point estimation. A finite supplied boundary SE is retained even when it does not support a calibrated interval. Numerical limits from such an SE are available only in separate columns when numerical_wald = TRUE; a warning identifies requested limits lacking a supported reference distribution. A missing SE is never imputed. Reporting fallbacks require finite, nonnegative covariance diagonals rather than replacing negative values by zero. No calibrated boundary-variance test or interval is implemented by this reporting option.

Value

summary_outcome, summary_visiting and summary_observation return a data frame with one row per parameter and the columns

parameter	Parameter name.
stage	"outcome", "visiting" or "observation".
estimate, se	Estimate and standard error (NA when not available).
df	Degrees of freedom used for the interval (Inf for a normal reference).

lower, upper
: Available numerical 95% interval limits, with validity reported separately; NA when ci_available is FALSE.

point_valid
: TRUE when the estimate is finite and the fit diagnostics report success, convergence and identification.

inference_valid
: Whether inference is supported within its stated scope, separately from numerical availability. At zero frailty, conditional_supported gives TRUE for eligible computed conditional covariance under the fitted-model and selected-branch/face assumptions. Descriptive loading, relevance and collinearity screens do not control this flag, and missing descriptive diagnostics alone do not assign NA. Existing fit, estimability and covariance checks apply. FALSE denotes unavailable or invalid inference under those checks. Conditional support does not prove population identification, nominal coverage, or selection-adjusted coverage.

inference_status
: For example regular, conditional, conditional_supported, not_assessed, unavailable, covariance_failure, or fit_invalid, according to covariance eligibility and fit checks. A descriptive warning does not replace conditional_supported for otherwise eligible zero-frailty inference.

identification_status, identification_warning
: Diagnostic status and explanation of identification concerns. These describe design and fitted-information checks, not a proof of identification at the true parameter or a guarantee of coverage.

ci_available
: TRUE when limits from the stated regular or conditional calculation are present, including retained numerical limits whose validity is FALSE or NA. This flag reports availability, not nominal coverage. Uncalibrated numerical boundary-variance limits requested separately do not set this flag. A missing covariance or SE is not replaced by zero or otherwise imputed.

regular_inference_valid
: TRUE specifically when regular inference is available for that parameter and stage. Fixed-effect inference in an LMM is checked separately from the covariance-parameter Hessian. Visiting-regression inference is checked separately from downstream observation or outcome variance boundaries.

regular_lower, regular_upper
: 95% regular interval limits; NA when regular_inference_valid is FALSE.

se_available
: TRUE for a finite, nonnegative reported SE, including an SE whose validity is limited to a stated conditional scope. This flag alone does not assert nominal interval coverage.

se_status
: Availability and, when known from the fitted object, the reason for a missing or invalid SE: available, disabled, not_computed, boundary_not_computed, numerical_failure, identifiability_failure, or unknown. A boundary flag alone does not establish a numerical failure. unknown is used when saved information cannot identify the cause.

ci_status
: regular, conditional, numerical_only, covariance_failure, boundary_uncalibrated, se_unavailable, fit_invalid, reference_unavailable, or inference_unavailable, according to the point estimate, SE, reference distribution, and stated calculation scope. A conditional CI status describes the covariance calculation;

conditional_supported inference status describes eligibility under the stated assumptions. After a recorded covariance failure, retained finite numerical limits have numerical_only status; this does not certify their covariance calculation. Descriptive concerns are recorded separately in identification_status and identification_warning.

conditional_inference_available
: TRUE when a valid point estimate and finite SE support an interval in the reported conditional scope, or regular inference is available.

conditional_lower, conditional_upper
: 95% limits using that covariance; NA when it is unavailable. At a selected variance boundary they hold the chosen branch or face fixed, omit selection uncertainty and are not selection-adjusted intervals.

inference_scope, inference_reason
: Description of the covariance and the availability or limitation of regular inference. For a CIMEHR fit at a selected zero-frailty branch, stacked inference includes the remaining identifiable nuisance quantities and is conditional on that branch and variance face (see CIMEHR).

parameter_boundary
: TRUE for a reported variance parameter on its zero boundary. It does not imply that other parameters or the entire fit failed.

fit_boundary
: TRUE when the fitted model reports a variance or frailty boundary. This is separate from convergence and SE availability.

boundary
: Legacy flag retained for compatibility. In ordinary stage and outcome tables this describes the fit; in spline nuisance tables it describes the parameter. Use parameter_boundary and fit_boundary for consistent meanings across methods.

variance_type
: Type of variance estimate, for example "hc3", "model_based", "plug_in_conditional", "stacked_sandwich" or "stacked_branch_conditional".

The visiting and observation tables of the spline methods (cimehr_spline, cimehr_spline_hl) add the columns scale, variance_label, conditional_inference_valid, and se_source. The legacy spline column conditional_inference_valid is an alias for conditional interval availability, not a claim of selection-adjusted coverage. Tables for cimehr_hl and cimehr_spline_hl add method_key, variant and native_base_method.

With numerical_wald = TRUE, visiting and observation tables also contain numerical_lower, numerical_upper, numerical_wald_available, and numerical_wald_scope. These are separately labelled arithmetic limits; they never alter lower, upper, or inference-validity flags, and do not establish boundary calibration.

coef_stage returns the estimate column as a numeric vector named by parameter.

extract_coefficient returns the one-row data frame of the requested parameter and stops with an error if it is not in the stage.

See Also

fit_method, coef.cimehr_fit, comparison_methods.

Examples

```
data("cimehr_example", package = "CIMEHR")
dat <- merge(cimehr_example$participants, cimehr_example$visits,
             by = "id", all.x = TRUE)
dat <- dat[dat$id <= 200, ]  # 200 of the 1,000 participants keeps the example fast
dat <- dat[order(dat$id, dat$time, na.last = TRUE), ]
rownames(dat) <- NULL

fit <- fit_method("cimehr", dat, visit_covs = c("Z", "X"),
                  obs_fixed_covs = c("Z", "X"), obs_random_covs = "Z",
                  long_covs = c("Z", "X"), outcome_random_covs = "Z",
                  censor_col = "C")
summary_outcome(fit)
summary_visiting(fit)
summary_observation(fit)
coef_stage(fit, "visiting")
extract_coefficient(fit, "Z")
extract_coefficient(fit, "alpha.X", stage = "observation")

# A model without an observation stage returns an empty table.
fit_lmm <- LMM(dat, long_covs = c("Z", "X"), censor_col = "C")
nrow(summary_observation(fit_lmm))
```

Index

∗ **datasets**
cimehr_example, 16
∗ **methods**
cimehr_fit-methods, 17
∗ **models**
baseline_mean, 6
CIMEHR, 9
cimehr_fit-methods, 17
CIMEHR_spline, 20
comparison_methods, 26
fit_method, 32
meanfit, 37
stage_summaries, 42
∗ **package**
CIMEHR-package, 2
∗ **regression**
baseline_mean, 6
CIMEHR, 9
CIMEHR_spline, 20
comparison_methods, 26
fit_method, 32
meanfit, 37

as.data.frame.cimehr_meanfit *(*meanfit*)*, 37
available_comparison_methods, *3*, *6*, *15*, *27*, *30*
available_comparison_methods *(*fit_method*)*, 32

baseline_mean, 6, *41*
bs, *21*, *23*

CIMEHR, *3*, *6*, 9, *17*, *20–25*, *27*, *33*, *36*, *40*, *42*, *45*
CIMEHR-package, 2
cimehr_example, 16
cimehr_fit-methods, 17
CIMEHR_HL, *3*, *20*, *24*, *27*
CIMEHR_HL *(*CIMEHR*)*, 9
cimehr_meanfit *(*meanfit*)*, 37
cimehr_methods, *3*
cimehr_methods *(*fit_method*)*, 32
CIMEHR_spline, *3*, *5*, *7*, *15*, 20, *27*, *36*
CIMEHR_spline_HL, *3*, *27*
CIMEHR_spline_HL *(*CIMEHR_spline*)*, 20
coef.cimehr_fit, *15*, *30*, *45*
coef.cimehr_fit *(*cimehr_fit-methods*)*, 17
coef_stage, *5*, *18*
coef_stage *(*stage_summaries*)*, 42
comparison_methods, *5*, *19*, 26, *45*
confint, *43*
confint.cimehr_fit *(*cimehr_fit-methods*)*, 17

EHRJoint, *3*
EHRJoint *(*comparison_methods*)*, 26
extract_coefficient, *5*, *36*
extract_coefficient *(*stage_summaries*)*, 42

fit_method, *3*, *6*, *8*, *14*, *15*, *17*, *19*, *23*, *25–27*, *30*, 32, *38*, *41*, *42*, *45*

IIRR, *3*
IIRR *(*comparison_methods*)*, 26
IIRR_balancing, *3*
IIRR_balancing *(*comparison_methods*)*, 26
Inverse_intensity_rate_ratio_balancing *(*comparison_methods*)*, 26
Inverse_intensity_rate_ratio_weighting *(*comparison_methods*)*, 26

JMVL_G, *3*
JMVL_G *(*comparison_methods*)*, 26
Joint_modeling_visiting_and_longitudinal_Liang *(*comparison_methods*)*, 26

Liang, *3*, *36*
Liang *(*comparison_methods*)*, 26
Lin_Ying, *3*

Lin_Ying (comparison_methods), 26
Linear_increment_IP, 3
Linear_increment_IP
        (comparison_methods), 26
Linear_mixed_model
        (comparison_methods), 26
LMM, 3, 36
LMM (comparison_methods), 26

meanfit, 6–8, 22, 25, 34, 37
method wrappers, 17, 42
method_comparisons, 3, 30
method_comparisons (fit_method), 32
model.matrix, 11, 36
Multiple_imputation_IP, 3
Multiple_imputation_IP
        (comparison_methods), 26

nlminb, 10
ns, 21, 23

Pairwise_likelihood, 3
Pairwise_likelihood
        (comparison_methods), 26
plot.cimehr_fit (baseline_mean), 6
plot.cimehr_meanfit (meanfit), 37
population_trajectory, 41
population_trajectory (baseline_mean), 6
predict.cimehr_fit (baseline_mean), 6
predict.cimehr_spline_fit
        (CIMEHR_spline), 20
print.cimehr_fit (cimehr_fit-methods),
        17
print.cimehr_meanfit (meanfit), 37

stage_summaries, 42
summary.cimehr_fit
        (cimehr_fit-methods), 17
summary.cimehr_meanfit (meanfit), 37
summary_observation, 5, 15, 18, 25, 30, 36
summary_observation (stage_summaries),
        42
summary_outcome, 5, 13, 15, 18, 19, 25, 30, 36
summary_outcome (stage_summaries), 42
Summary_stat, 3
Summary_stat (comparison_methods), 26
summary_visiting, 5, 12, 15, 18, 25, 30, 36
summary_visiting (stage_summaries), 42

vcov.cimehr_fit (cimehr_fit-methods), 17