Package 'rolescry'

October 28, 2026

Title Name-Blind Variable-Role Detection by Data Signature

Version 0.3.1

Description Deterministic, name-blind detection of variable roles (group, outcome, survival or duration time and event, coupled binary-continuous pairs, calendar time index, paired and agreement measurements, repeated measures and panels, scale and rating items, subject or entity identifier, covariate) in tabular data from any domain -- clinical, financial, social, experimental. Roles are assigned from each column's information-theoretic signature -- Shannon entropy, normalized mutual information, and distributional shape -- rather than from column names, so renaming columns to 'col_1', 'col_2', ... does not change the result (``Data inspice, non nomen''). An optional, capped name-based hint and automatic header-row detection are also provided. No large language models and no external data transmission. Extracted from the 'MDStatR' biostatistics engine; see Boynukara (2026) <doi:10.5281/zenodo.20707791>.

License Apache License (== 2.0)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, utils

Suggests moments, diptest, stringdist, readxl, openxlsx, haven, testthat (>= 3.0.0), knitr, rmarkdown, spelling

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/canboynukara/rolescry

BugReports https://github.com/canboynukara/rolescry/issues

Language en-US

NeedsCompilation no

Author Can Boynukara [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6075-3923>)

Maintainer Can Boynukara <canboynukara1@gmail.com>

Repository CRAN

Date/Publication 2026-10-28 18:40:02 UTC

Contents

compute_nmi 2
detect_header 3
detect_roles 4
harmonize_roles 6
read_data 7
rolescry_default_name_bonus 8
verify_roles 9

Index **11**

`compute_nmi` *Normalized mutual information*

Description

Computes the normalized mutual information (NMI) between two discrete variables, a name-blind, information-theoretic measure of association in $[0, 1]$. NMI is the mutual information divided by the smaller of the two marginal Shannon entropies; it is 0 for independent variables and 1 for a perfect (deterministic) association, and unlike a raw chi-squared it is comparable across variables with different numbers of levels.

Usage

```
compute_nmi(x, y = NULL)
```

Arguments

`x`	Either a two-way contingency table / matrix of counts, or a vector (factor, character, or numeric) of the first variable.
`y`	Optional. If `x` is a vector, the second variable's vector; a contingency table is formed via `table(x, y)` on complete cases. Ignored when `x` is already a table/matrix.

Value

A single numeric in $[0, 1]$. Returns 0 for degenerate input (fewer than two rows/columns, zero total, or near-zero marginal entropy).

Examples

```
set.seed(1)
g <- sample(c("A", "B", "C"), 200, replace = TRUE)
y <- ifelse(g == "A", "yes", sample(c("yes", "no"), 200, replace = TRUE))
compute_nmi(g, y)            # > 0: g carries information about y
compute_nmi(g, sample(g))    # ~0: shuffled -> independent
```

`detect_header` *Detect the header row of a raw, unparsed table*

Description

Given a raw table read with *no* header (every cell character), scores each of the first rows with a 7-signal weighted heuristic (alphabetic ratio, non-numeric ratio, uniqueness, normalized Shannon entropy, median string length, alpha-vs-next-row transition, fill completeness) and returns the most header-like row plus repaired, unique column names. Empty cells are upward-filled (merged-cell repair) and any still-empty name becomes `col_<j>`. Base + stats only; no file-format dependencies.

Usage

```
detect_header(raw, verbose = FALSE)
```

Arguments

`raw`	A data.frame or matrix of the raw sheet, read with `header = FALSE` so the header row appears as data.
`verbose`	Logical; emit the chosen row via `message()` if TRUE.

Value

A list with `header_row` (integer), `score` (numeric), `names` (repaired character vector, `length == ncol(raw)`), and `all_scores`.

See Also

`read_data()`

Examples

```
raw <- data.frame(
  V1 = c("age", "34", "51"),
  V2 = c("sex", "M", "F"),
  V3 = c("score", "8.1", "7.4"),
  stringsAsFactors = FALSE
)
detect_header(raw)$names
```

`detect_roles` *Detect variable roles by data signature, not by name*

Description

Inspects an already-loaded data frame and assigns each column (or group of columns) to statistical roles – group variable, continuous/binary outcome, survival time and event, paired and agreement measurement pairs, repeated measures, scale items, subject id, and covariates – using only the data information-theoretic signature (Shannon entropy, distributional shape, inter-column structure) and never the column names. Renaming columns to `col_1, col_2, ...` does not change the result (the name-blindness, or "turnusol", invariant).

Usage

```
detect_roles(data, name_bonus = NULL, verbose = FALSE)

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

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

Arguments

`data`	A `data.frame` (already loaded; for header-aware loading see `read_data()`).
`name_bonus`	Optional named list mapping role keys to character vectors of case-insensitive keyword regex fragments, e.g. `list(group_var = c("treat", "arm"), outcome_binary = c("death"))`. Recognized keys: `group_var`, `outcome_continuous`, `outcome_binary`, `subject_id`, `time_variable`, `event_variable`. `NULL` (default) = pure signature detection.
`verbose`	Logical; if `TRUE`, emit per-role progress via `message()`. Default `FALSE` (silent).
`x`	A `role_detection` object.
`...`	Ignored.
`object`	A `role_detection` object.

Details

Detection is purely mathematical by default (`name_bonus = NULL`). When a keyword dictionary is supplied via `name_bonus`, column names act only as a small, capped tie-breaker (at most a +10 point nudge, i.e. <= 10 percent, applied to candidate selection for the group, outcome, subject-id and survival roles) – the reported confidence stays the mathematical signature. See `rolescry_default_name_bonus()` for a ready-made dictionary.

A column can hold more than one role (e.g. a measurement in both `paired_pairs` and `agreement_pairs`). The `covariate` role is residual: it lists the continuous, categorical and binary columns that no other role claimed. A role whose evidence does not single out a column is reported as not found rather than guessed.

The coupled_pair role (0.3.0) reports a dependent pair of one binary and one positive continuous column whose reading is ambiguous: it may be a survival (time, event) pair or a (cause, binary outcome) pair, and the data alone cannot say which. The survival roles are assigned instead only when censoring geometry singles the survival reading out. The entry carries continuous_col, binary_col and the two interpretations; the choice is the user's.

The time_index role (0.3.0) reports calendar columns: Date/POSIXct classes, or character columns whose every value parses as a date. The class and format are properties of the values, so name-blindness is preserved. On a balanced long panel whose entity/time axes are value-symmetric, the axes are reported as ambiguous (repeated_measures$panel_axes) unless serial dependence of the measures singles one reading out.

Value

An S3 object of class "role_detection": a list with

var_info data.frame(column, type) – value-based column typing.

roles named list; each entry has found, columns, score, max_score, pct, detected_by and a components score breakdown. Meaning of score/pct per role: group_var: score = pct = 100 * B (normalized-entropy ranking; max_score = 100; there is no description-length test). Paired/agreement pairs, ordinal scale items and the outcomes: score is in bits, max_score is NA, and pct is the display map 100 * (1 - 2^(-bits/8)). Survival, wide repeated measures and binary scale items (ported point scorers): score and max_score are points and pct is the points ratio.

value_types named character vector of per-column value-type labels.

potential_pairs list of candidate continuous column pairs with MDL pair bits and shift bits. The paired/agreement split is the sign of the shift bits, a BIC-type test of a mean difference between the two columns: agreement_pairs means "no detectable systematic shift", so a method-comparison pair with a real calibration bias is labeled paired_pairs, and the label depends on n. Exactly one pair is reported across the two roles.

n_obs, n_var dataset dimensions.

See Also

read_data(), compute_nmi(), rolescry_default_name_bonus()

Examples

```
set.seed(1)
d <- data.frame(
  arm  = rep(c(0, 1), each = 50),
  pre  = rnorm(100, 10, 2),
  post = rnorm(100, 11, 2),
  resp = rbinom(100, 1, 0.4)
)
res <- detect_roles(d)
res
res$roles$group_var$columns
```

`harmonize_roles` *Align detected variable roles across sites*

Description

Runs `detect_roles()` name-blind on each site and reports, per role, which column each site holds, plus descriptive consistency checks (value type, event coding, group-label overlap, range overlap, battery size). Alignment is by role only; column names appear only in the returned cells. The residual `covariate` role is excluded from `map`, `checks` and `site_only`. `aligned` means the detector found the role at every site; it does not establish that the columns measure the same thing; read `checks` (levels, Jaccard, ranges) before pooling. Cell order is alphabetical and carries no pairing direction.

Usage

```
harmonize_roles(sites)

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

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

Arguments

`sites`	A named list (unique, non-empty names) of at least two data.frames.
`x, object`	A `role_harmonization` object.
`...`	Ignored.

Value

An object of class `"role_harmonization"`: a list with `map` (role, status, one column per site), `checks` (role, check, one column per site, flag), `site_only` (site, role, columns), `n_sites` and `sites`. Flags are `type_mismatch`, `coding_mismatch` and `size_mismatch`; no numeric threshold decides a flag. `summary` returns `map`.

Examples

```
set.seed(1)
mk <- function(n) data.frame(id = seq_len(n), arm = sample(c("a", "b"), n, TRUE),
                             x = rnorm(n), y = rnorm(n))
harmonize_roles(list(north = mk(60), south = mk(80)))
```

read_data *Read a data file with automatic header detection*

Description

Reads a tabular file into a `data.frame`, detecting the header row with `detect_header()` (the data is first read with no header so the header row is visible as data). Delimited text (.csv/.tsv) is read with base R and always works; spreadsheet and statistical formats use optional packages and degrade gracefully with an actionable error if the package is absent.

Usage

```
read_data(
  path,
  header = NULL,
  sheet = NULL,
  na_strings = c("", "NA", "N/A", "n/a", "na", "NULL", "null", "."),
  verbose = FALSE
)
```

Arguments

`path`	Path to the file.
`header`	Optional integer giving the 1-based header row to use directly, bypassing detection. `NULL` (default) auto-detects.
`sheet`	Optional sheet name/index for Excel files.
`na_strings`	Character vector of tokens mapped to `NA` before type conversion.
`verbose`	Logical; emit the detected header row via `message()`.

Details

Supported: `.csv`, `.tsv`/`.tab` (base); `.xlsx`/`.xls`/`.xlsm` (Suggests: readxl or openxlsx); `.sav`/`.sas7bdat`/`.dta` (Suggests: haven, header intrinsic); `.rds` (base, returned as stored).

Value

A `data.frame` with detected column names and per-column types inferred via `type.convert`.

See Also

`detect_header()`, `detect_roles()`

Examples

```
tmp <- tempfile(fileext = ".csv")
writeLines(c("age,sex,score", "34,M,8.1", "51,F,7.4"), tmp)
read_data(tmp)
file.remove(tmp)
```

`rolescry_default_name_bonus` *Default name-bonus keyword dictionary*

Description

Returns a ready-made, ASCII-English keyword dictionary suitable for the `name_bonus` argument of `detect_roles()`. It externalizes the hard-coded keyword lists that lived inside the original MDStatR engine (group/treatment, outcome, survival-time and event, subject-id terms) into a plain, inspectable, locale-neutral list.

Usage

```
rolescry_default_name_bonus(domain = c("clinical", "finance", "all"))
```

Arguments

`domain`	Keyword domain: `"clinical"` (default), `"finance"`, or `"all"` (the union). The dictionaries differ only in vocabulary; the capped tie-break mechanics are identical.

Details

Passing this turns column names into a small, capped tie-breaker only ($\leq$ 10 percent of the selection score); the mathematical signature still dominates (>= 90 percent), satisfying the name-blindness contract. Detection without it (`name_bonus = NULL`) is purely mathematical.

Value

A named list of character vectors (regex fragments), with keys `group_var`, `outcome_continuous`, `outcome_binary`, `subject_id`, `time_variable`, `event_variable`.

Examples

```
nb <- rolescry_default_name_bonus()
names(nb)
set.seed(1)
d <- data.frame(
  treatment_arm = rep(c("A", "B"), each = 60),
  biomarker     = rnorm(120),
  death         = rbinom(120, 1, 0.3)
```

```
)
detect_roles(d, name_bonus = nb)$roles$group_var$columns
```

`verify_roles` *Verify a declared analysis plan against the data signature*

Description

Checks a user-declared assignment of variable roles against what the data themselves can support. The claim is mapped to column positions once and the data are then examined name-blind (the same information-theoretic signature used by `detect_roles()`). Each claimed role gets one verdict: `contradicted` (a falsification backed by a value statistic), `consistent` (the data support it) or `unverifiable` (`sub_reason` `identifiability` or `coverage`). Structure the blind detector finds that the claim does not mention is listed separately.

Usage

```
verify_roles(data, claim)

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

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

Arguments

`data`	A `data.frame` with unique, non-empty column names.
`claim`	A named list. Names are role keys: scalar roles (`subject_id`, `group_var`, `outcome_continuous`, `outcome_binary`, `time_variable`, `event_variable`, `time_index`) take one column name; set roles (`repeated_measures`, `scale_items`, `covariate`) take one or more; pair roles (`coupled_pair`, `paired_pairs`, `agreement_pairs`) take a length-2 character vector or a list of them.
`x`, `object`	A `role_verification` object.
`...`	Ignored.

Details

`contradicted` is issued only for these falsifications, each backed by a value statistic: `subject_id` whose values repeat while a detected long panel has a different entity cluster; a `paired_pairs`/`agreement_pairs` pair that shares no information beyond the independence noise floor, whose members differ more than they agree, or with a constant member; a `coupled_pair` whose members are independent; an event column that is not a two-level or 0/1/2 coding, or negative follow-up times; `repeated_measures` that are not the measures of a detected long panel, or a wide set with no common factor; ordinal `scale_items` with no common factor, or binary items forming a one-hot partition; a `time_index` that is not a calendar column; a `group_var` that is all-distinct or calendar; a non-numeric `outcome_continuous`;

an outcome_binary without exactly two observed values. Absence of a time-event association is not a falsification (constant-hazard censoring makes the event independent of the observed time) and is reported as unverifiable(coverage). A consistent pair means the two columns share information and their difference is smaller than their mean. Each claimed pair is judged on its own. A survival claim (time_variable plus event_variable) is one claim shown as two rows and is counted once in counts.

Value

An object of class "role_verification": a list with table (role, columns, verdict, sub_reason, statistic, reason, blind), unclaimed (role, columns, statistic), counts (claims, testable, contradicted, consistent, unverifiable), n_obs and n_var.

Examples

```
set.seed(1)
rct <- data.frame(id = 1:60, arm = rep(c("a", "b"), 30),
                  pre = rnorm(60), post = rnorm(60))
verify_roles(rct, list(subject_id = "id", group_var = "arm",
                       paired_pairs = c("pre", "post")))
```

Index

compute_nmi, 2
compute_nmi(), *5*

detect_header, 3
detect_header(), *7*
detect_roles, 4
detect_roles(), *6–9*

harmonize_roles, 6

print.role_detection *(*detect_roles*)*, 4
print.role_harmonization
        *(*harmonize_roles*)*, 6
print.role_verification *(*verify_roles*)*,
        9

read_data, 7
read_data(), *3–5*
rolescry_default_name_bonus, 8
rolescry_default_name_bonus(), *4, 5*

summary.role_detection *(*detect_roles*)*, 4
summary.role_harmonization
        *(*harmonize_roles*)*, 6
summary.role_verification
        *(*verify_roles*)*, 9

type.convert, *7*

verify_roles, 9