Package: matrixCorr (via r-universe)

October 28, 2026

Type Package

Title Collection of Correlation, Agreement, and Reliability Estimators

Version 0.12.3

Author Thiago de Paula Oliveira [aut, cre]
(<https://orcid.org/0000-0002-4555-2584>)

Maintainer Thiago de Paula Oliveira <thiago.paula.oliveira@gmail.com>

Description Compute correlation, association, agreement, and reliability measures for small to high-dimensional datasets through a consistent matrix-oriented interface. Supports classical correlations (Pearson, Spearman, Kendall, Chatterjee's rank correlation), distance correlation, partial correlation with regularised estimators, shrinkage correlation for p >= n settings, robust correlations including biweight mid-correlation, percentage-bend, Winsorized, and skipped correlation, latent-variable methods for binary and ordinal data, pairwise and overall intraclass correlation for wide data, repeated-measures correlation, and agreement/reliability analyses based on Cohen's kappa, weighted kappa, multi-rater kappa, Gwet's AC1/AC2, Krippendorff's alpha, Bland-Altman methods, Lin's concordance correlation coefficient, Poisson GLMM concordance for count data, and repeated-measures intraclass/concordance correlation, including robust concordance based on minimum covariance determinant estimates. Implemented with optimized C++ backends using BLAS/OpenMP and memory-aware symmetric updates, and returns standard R objects with print/summary/plot methods plus optional Shiny viewers for matrix inspection. Methods based on Ledoit and Wolf (2004) <doi:10.1016/S0047-259X(03)00096-4>; high-dimensional shrinkage covariance estimation <doi:10.2202/1544-6115.1175>; Lin (1989) <doi:10.2307/2532051>; Wilcox (1994) <doi:10.1007/BF02294395>; Wilcox (2004) <doi:10.1080/0266476032000148821>; Hayes and Krippendorff (2007) <doi:10.1080/19312450709336664>; weighted repeated-measures correlation by Kondo et al. (2025) <doi:10.1002/sim.70046>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.4.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp (>= 1.1.0), ggplot2 (>= 3.5.2), Matrix (>= 1.7.2), robustbase, cli, generics, rlang

Suggests knitr, rmarkdown, MASS, mnormt, shiny, shinyWidgets, viridisLite, testthat (>= 3.0.0)

VignetteBuilder knitr

Enhances plotly

Roxygen list(markdown = TRUE)

URL https://github.com/Prof-ThiagoOliveira/matrixCorr

BugReports https://github.com/Prof-ThiagoOliveira/matrixCorr/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Repository https://prof-thiagooliveira.r-universe.dev

Date/Publication 2026-09-28 07:17:23 UTC

RemoteUrl https://github.com/prof-thiagooliveira/matrixcorr

RemoteRef HEAD

RemoteSha 1811a8cfbedffde8d81d5b683b591d568dde2bc5

Contents

[[.summary.corr_result . . . 4
$.summary.corr_result . . . 4
as.data.frame.corr_edge_list . . . 5
ba . . . 5
ba_rm . . . 11
bcdcor . . . 22
bicor . . . 26
biserial . . . 33
ccc . . . 37
ccc_glmm . . . 41
ccc_rm_reml . . . 46
ccc_rm_ustat . . . 59
cia . . . 63
cia_rm . . . 69
cohen_kappa . . . 76
dcor . . . 80
deprecated-matrixCorr . . . 85
estimate . . . 90
gwet_ac . . . 95
hsic . . . 101

icc . . . 105
icc_rm_reml . . . 110
kendall_tau . . . 118
krippendorff_alpha . . . 124
multirater_kappa . . . 128
pbcor . . . 133
pcorr . . . 138
pearson_corr . . . 146
plot.corr_edge_list . . . 151
plot.corr_matrix . . . 152
plot.corr_sparse . . . 153
plot.prob_agree . . . 153
polychoric . . . 154
polyserial . . . 159
print.ccc_ci . . . 163
print.corr_edge_list . . . 164
print.matrixCorr_ccc . . . 164
print.matrixCorr_ccc_ci . . . 165
print.rmcorr . . . 166
print.rmcorr_matrix . . . 167
print.summary.corr_result . . . 169
print.summary.latent_corr . . . 170
print.summary.matrixCorr . . . 171
prob_agree . . . 172
rmcorr . . . 175
robust_ccc . . . 179
robust_dcor . . . 183
shrinkage_corr . . . 188
skipped_corr . . . 192
spearman_rho . . . 198
summary.ccc_rm_reml . . . 203
summary.corr_edge_list . . . 205
summary.corr_matrix . . . 205
summary.corr_sparse . . . 206
tetrachoric . . . 207
view_corr_shiny . . . 211
view_rmcorr_shiny . . . 212
weighted_kappa . . . 214
wincor . . . 217
xi_corr . . . 222

Index **230**

[[.summary.corr_result *Summary Accessor for Correlation Summaries*

Description

Summary Accessor for Correlation Summaries

Usage

```
## S3 method for class 'summary.corr_result'
x[[i, ...]]
```

Arguments

x	A summary.corr_result object.
i	A column name or summary metadata key.
...	Unused.

Value

A summary column (if present) or summary metadata entry.

$.summary.corr_result *Summary Accessor for Correlation Summaries*

Description

Summary Accessor for Correlation Summaries

Usage

```
## S3 method for class 'summary.corr_result'
x$name
```

Arguments

x	A summary.corr_result object.
name	A column name or summary metadata key.

Value

A summary column (if present) or summary metadata entry.

as.data.frame.corr_edge_list *Edge-list data-frame view*

Description

Edge-list data-frame view

Usage

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

Arguments

x	A corr_edge_list object.
row.names	Ignored.
optional	Ignored.
...	Unused.

Value

A data frame with columns row, col, value.

ba *Bland-Altman statistics with confidence intervals*

Description

Computes Bland-Altman mean difference and limits of agreement (LoA) between two numeric measurement vectors, including t-based confidence intervals for the mean difference and for each LoA using 'C++' backend. If group2 is omitted and group1 is a numeric matrix or data frame with at least two numeric columns, ba() computes all pairwise contrasts across methods and returns a pairwise Bland-Altman matrix object.

Note: Lin's concordance correlation coefficient (CCC) is a complementary, single-number summary of agreement (precision + accuracy). It is useful for quick screening or reporting an overall CI, but may miss systematic or magnitude-dependent bias; consider reporting CCC alongside Bland-Altman.

Usage

```
ba(
  group1,
  group2,
  loa_multiplier = 1.96,
  mode = 1L,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE
)

## S3 method for class 'ba'
print(
  x,
  digits = 3,
  ci_digits = 3,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ba'
summary(object, digits = 3, ci_digits = 3, ...)

## S3 method for class 'summary.ba'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ba'
plot(
  x,
  title = "Bland-Altman Plot",
  subtitle = NULL,
  point_alpha = 0.7,
  point_size = 2.2,
  line_size = 0.8,
  shade_ci = TRUE,
```

```
  shade_alpha = 0.08,
  smoother = c("none", "loess", "lm"),
  symmetrize_y = TRUE,
  show_value = TRUE,
  ...
)

## S3 method for class 'ba_matrix'
print(
  x,
  digits = 3,
  ci_digits = 3,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  style = c("pairs", "matrices"),
  ...
)

## S3 method for class 'ba_matrix'
summary(object, digits = 3, ci_digits = 3, ...)

## S3 method for class 'summary.ba_matrix'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ba_matrix'
plot(
  x,
  pairs = NULL,
  against = NULL,
  facet_scales = c("free_y", "fixed"),
  title = "Bland-Altman (pairwise)",
  point_alpha = 0.6,
  point_size = 1.8,
  line_size = 0.7,
  shade_ci = TRUE,
  shade_alpha = 0.08,
```

```
  smoother = c("none", "loess", "lm"),
  show_value = TRUE,
  ...
)
```

Arguments

group1	Numeric vector of paired measurements, or a numeric matrix/data frame with at least two numeric columns when group2 is omitted.
group2	Optional numeric vector of paired measurements. If omitted, group1 must be a numeric matrix or data frame and all unordered numeric column pairs are analysed.
loa_multiplier	Positive scalar; the multiple of the standard deviation used to define the LoA (default 1.96 for nominal 95\ intervals always use $t_{n-1,\,1-\alpha/2}$ regardless of this choice.
mode	Integer; 1 uses group1 - group2, 2 uses group2 - group1.
conf_level	Confidence level for CIs (default 0.95).
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
verbose	Logical; if TRUE, prints how many OpenMP threads are used.
x	A "ba_matrix" object.
digits	Number of digits for estimates (default 3).
ci_digits	Number of digits for CI bounds (default 3).
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Passed to ggplot2::theme() (ggplot path) or plot().
object	A "ba" object.
title	Plot title.
subtitle	Optional subtitle. If NULL, shows n and LoA summary.
point_alpha	Point transparency.
point_size	Point size.
line_size	Line width for mean/LoA.
shade_ci	Logical; if TRUE, draw shaded CI bands instead of 6 dashed lines.
shade_alpha	Transparency of CI bands.
smoother	One of "none", "loess", "lm" to visualize proportional bias.
symmetrize_y	Logical; if TRUE, y-axis centered at mean difference with symmetric limits.

show_value	Logical; included for a consistent plotting interface. Bland-Altman plots do not overlay numeric cell values, so this argument currently has no effect.
style	Show the pairwise result as "pairs" or "matrices".
pairs	Optional character vector of pair labels to display.
against	Optional single method name; if supplied, only contrasts involving that method are plotted.
facet_scales	Either "free_y" (default) or "fixed".

Details

Given paired measurements (x_i, y_i), Bland-Altman analysis uses $d_i = x_i - y_i$ (or $y_i - x_i$ if mode = 2) and $m_i = (x_i + y_i)/2$. The mean difference $\bar{d}$ estimates bias. The limits of agreement (LoA) are $\bar{d} \pm z \cdot s_d$, where s_d is the sample standard deviation of d_i and z (argument loa_multiplier) is typically 1.96 for nominal 95% LoA.

When group2 is omitted and group1 is a wide numeric matrix or data frame, the same two-method calculation is applied to every unordered column pair. The returned pairwise matrices follow the requested mode: with mode = 1, upper-triangle entries represent row minus column; with mode = 2, upper-triangle entries represent column minus row.

Confidence intervals use Student's t distribution with $n - 1$ degrees of freedom, with

- Mean-difference CI given by $\bar{d} \pm t_{n-1,\,1-\alpha/2}\, s_d/\sqrt{n}$; and
- LoA CI given by $(\bar{d} \pm z\, s_d) \;\pm\; t_{n-1,\,1-\alpha/2}\, s_d\, \sqrt{3/n}$.

Assumptions include approximately normal differences and roughly constant variability across the measurement range; if differences increase with magnitude, consider a transformation before analysis. Missing values are removed pairwise (rows with an NA in either input are dropped before calling the C++ backend).

Probability of agreement, available through prob_agree, is a tolerance-based companion to Bland-Altman analysis. Bland-Altman reports bias and limits of agreement for paired differences. prob_agree() instead uses the sampling distribution of estimated differences to quantify the probability that two estimated quantities or curves agree within a user-specified practical tolerance.

Value

If group1 and group2 are both supplied, an object of class "ba" (list) with elements:

- means, diffs: numeric vectors
- groups: data.frame used after NA removal
- n_obs: integer, number of complete pairs used.
- based.on: compatibility alias for n_obs.
- lower.limit, mean.diffs, upper.limit
- lines: named numeric vector (lower, mean, upper)
- CI.lines: named numeric vector for CIs of those lines
- loa_multiplier, critical.diff

If group2 is omitted and group1 is a wide numeric matrix or data frame, an object of class "ba_matrix" with pairwise components:

- bias: pairwise matrix of Bland-Altman mean differences.
- sd_loa: pairwise matrix of SDs of differences.
- loa_lower, loa_upper: pairwise matrices of LoA endpoints.
- width: pairwise matrix of LoA widths.
- n: integer pairwise matrix of complete-case counts.
- mean_ci_low, mean_ci_high: pairwise matrices of CI bounds for the mean difference.
- loa_lower_ci_low, loa_lower_ci_high: pairwise matrices of CI bounds for the lower LoA.
- loa_upper_ci_low, loa_upper_ci_high: pairwise matrices of CI bounds for the upper LoA.
- methods: character vector of analysed method names.
- loa_multiplier, mode: calculation settings reused for every pair.

Author(s)

Thiago de Paula Oliveira

References

Bland JM, Altman DG (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, 307-310.

Bland JM, Altman DG (1999). Measuring agreement in method comparison studies. *Statistical Methods in Medical Research*, 8(2), 135-160.

See Also

print.ba, plot.ba, ccc, prob_agree, ccc_rm_ustat, ccc_rm_reml

Examples

```
set.seed(1)
x <- rnorm(100, 100, 10)
y <- x + rnorm(100, 0, 8)
fit_ba <- ba(x, y)
print(fit_ba)
estimate(fit_ba)
tidy(fit_ba)
confint(fit_ba)
plot(fit_ba)

# Pairwise Bland-Altman across 3 methods
set.seed(7)
wide3 <- data.frame(
  ref = rnorm(80, 100, 8),
  m2  = rnorm(80, 101, 8),
  m3  = rnorm(80,  99, 9)
)
```

```
fit_ba3 <- ba(wide3)
print(fit_ba3)
summary(fit_ba3)
tidy(fit_ba3)
plot(fit_ba3)
```

---

ba_rm *Bland-Altman for repeated measurements*

Description

Repeated-measures Bland-Altman (BA) analysis for method comparison based on a mixed-effects model fitted to **subject-time matched paired differences**. The fitted model includes a subject-specific random intercept and, optionally, an AR(1) residual correlation structure within subject.

The function accepts either exactly two methods or ≥ 3 methods. With exactly two methods it returns a single fitted BA object. With ≥ 3 methods it fits the same model to every unordered method pair and returns pairwise matrices of results.

Required variables

- `response`: numeric measurements.
- `subject`: subject identifier.
- `method`: method label with at least two distinct levels.
- `time`: replicate/time key used to form within-subject pairs.

For any analysed pair of methods, only records where both methods are present for the same `subject` and integer-coerced `time` contribute to the fit. Rows with missing values in any required field are excluded for that analysed pair.

Usage

```
ba_rm(
  data = NULL,
  response,
  subject,
  method,
  time,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE,
  loa_multiplier = 1.96,
  include_slope = FALSE,
  use_ar1 = FALSE,
  ar1_rho = NA_real_,
  max_iter = 200L,
  tol = 1e-06
```

```
)

## S3 method for class 'ba_repeated'
print(
  x,
  digits = 3,
  ci_digits = 3,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ba_repeated_matrix'
print(
  x,
  digits = 3,
  ci_digits = 3,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  style = c("pairs", "matrices"),
  ...
)

## S3 method for class 'ba_repeated'
plot(
  x,
  title = "Bland-Altman (repeated measurements)",
  subtitle = NULL,
  point_alpha = 0.7,
  point_size = 2.2,
  line_size = 0.8,
  shade_ci = TRUE,
  shade_alpha = 0.08,
  smoother = c("none", "loess", "lm"),
  symmetrize_y = TRUE,
  show_points = TRUE,
  show_value = TRUE,
  ...
)

## S3 method for class 'ba_repeated_matrix'
plot(
```

```
  x,
  pairs = NULL,
  against = NULL,
  facet_scales = c("free_y", "fixed"),
  title = "Bland-Altman (repeated, pairwise)",
  point_alpha = 0.6,
  point_size = 1.8,
  line_size = 0.7,
  shade_ci = TRUE,
  shade_alpha = 0.08,
  smoother = c("none", "loess", "lm"),
  show_points = TRUE,
  show_value = TRUE,
  ...
)
```

Arguments

`data`	Optional data frame-like object. If supplied, `response`, `subject`, `method`, and `time` may be column names. Objects not already inheriting from `data.frame` are first coerced with `as.data.frame()`.
`response`	Numeric response vector, or a single character string naming the response column in `data`.
`subject`	Subject identifier vector (integer, numeric, or factor), or a single character string naming the subject column in `data`.
`method`	Method label vector (character, factor, integer, or numeric), or a single character string naming the method column in `data`. At least two distinct method levels are required.
`time`	Replicate/time index vector (integer or numeric), or a single character string naming the time column in `data`. Values are coerced to integer before pairing and before AR(1) contiguity checks.
`conf_level`	Confidence level for Wald confidence intervals for the reported bias and both LoA endpoints. Must lie in `(0, 1)`. Default `0.95`.
`n_threads`	Integer ≥ 1. Number of OpenMP threads. Defaults to `getOption("matrixCorr.threads", 1L)`.
`verbose`	Logical. If `TRUE`, print progress for the pairwise ≥ 3-method path. It has no material effect in the exactly-two-method path.
`loa_multiplier`	Positive scalar giving the SD multiplier used to form the limits of agreement. Default is `1.96`. In the exported `ba_rm()` interface this default is fixed and does not depend on `conf_level`.
`include_slope`	Logical. If `TRUE`, the model includes the paired mean as a fixed effect and estimates a proportional-bias slope. The reported BA centre remains the fitted mean difference at the centred reference paired mean used internally by the backend; the returned LoA remain horizontal bands and are not regression-adjusted curves.

use_ar1	Logical. If TRUE, request an AR(1) residual structure within subject over contiguous integer time blocks.
ar1_rho	Optional AR(1) parameter. Must satisfy abs(ar1_rho) < 0.999 when supplied. If NA and use_ar1 = TRUE, the backend estimates rho separately for each analysed pair.
max_iter	Maximum number of EM/GLS iterations used by the backend.
tol	Convergence tolerance for the backend EM/GLS iterations.
x	A "ba_repeated_matrix" object.
digits	Number of digits for estimates (default 3).
ci_digits	Number of digits for CI bounds (default 3).
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Additional theme adjustments passed to ggplot2::theme(...) (e.g., plot.title.position = "plot", axis.title.x = element_text(size=11)).
style	Show as pairs or matrix format?
title	Plot title (character scalar). Defaults to "Bland–Altman (repeated measurements)" for two methods and "Bland–Altman (repeated, pairwise)" for the faceted matrix plot.
subtitle	Optional subtitle (character scalar). If NULL, a compact summary is shown using the fitted object.
point_alpha	Numeric in [0, 1]. Transparency for scatter points drawn at (pair mean, pair difference) when point data are available. Passed to ggplot2::geom_point(alpha = ...). Default 0.7.
point_size	Positive numeric. Size of scatter points; passed to ggplot2::geom_point(size = ...). Default 2.2.
line_size	Positive numeric. Line width for horizontal bands (bias and both LoA) and, when requested, the proportional-bias line. Passed to ggplot2::geom_hline(linewidth = ...) (and geom_abline). Default 0.8.
shade_ci	Logical. If TRUE and confidence intervals are available in the object (CI.lines for two methods; *_ci_* matrices for the pairwise case), semi-transparent rectangles are drawn to indicate CI bands for the bias and each LoA. If FALSE, dashed horizontal CI lines are drawn instead. Has no effect if CIs are not present. Default TRUE.
shade_alpha	Numeric in [0, 1]. Opacity of the CI shading rectangles when shade_ci = TRUE. Passed to ggplot2::annotate("rect", alpha = ...). Default 0.08.
smoother	One of "none", "loess", or "lm". Adds an overlaid trend for differences vs means when points are drawn, to visualise proportional bias. "lm" fits a straight line with no SE ribbon; "loess" draws a locally-smoothed curve (span 0.9) with no SE ribbon; "none" draws no smoother. Ignored if show_points = FALSE or if no point data are available.

`symmetrize_y` Logical (two-method plot only). If `TRUE`, the y-axis is centred at the estimated bias and expanded symmetrically to cover all elements used to compute the range (bands, CIs, and points if shown). Default `TRUE`.

`show_points` Logical. If `TRUE`, per-pair points are drawn for the exactly-two-method path from the stored `means` and `diffs` vectors. Pairwise matrix plots draw points reconstructed from the canonical long-form columns stored by `ba_rm()`. If `FALSE` or if point data are unavailable, only the bands (and optional CI indicators) are drawn. Default `TRUE`.

`show_value` Logical; included for a consistent plotting interface. Repeated-measures Bland-Altman plots do not overlay numeric cell values, so this argument currently has no effect.

`pairs` (Faceted pairwise plot only.) Optional character vector of labels specifying which method contrasts to display. Labels must match the "row - column" convention used by `print()`/`summary()` (e.g., `"B - A"`). Defaults to all upper-triangle pairs.

`against` (Faceted pairwise plot only.) Optional single method name. If supplied, facets are restricted to contrasts of the chosen method against all others. Ignored when `pairs` is provided.

`facet_scales` (Faceted pairwise plot only.) Either `"free_y"` (default) to allow each facet its own y-axis limits, or `"fixed"` for a common scale across facets. Passed to `ggplot2::facet_wrap(scales = ...)`.

Details

For a selected pair of methods (a, b), the backend first forms complete within-subject pairs at matched `subject` and integer-coerced `time`. Let

$$d_{it} = y_{itb} - y_{ita}, \qquad m_{it} = \frac{y_{ita} + y_{itb}}{2},$$

where d_{it} is the paired difference and m_{it} is the paired mean for subject i at time/replicate t. Only complete subject-time matches contribute to that pairwise fit.

If multiple rows are present for the same `subject-time-method` combination within an analysed pair, the backend keeps the last encountered value for that combination when forming the pair. The function therefore implicitly assumes at most one observation per `subject-time-method` cell for each analysed contrast.

The fitted model for each analysed pair is

$$d_{it} = \beta_0 + \beta_1 x_{it} + u_i + \varepsilon_{it},$$

where $x_{it} = m_{it}$ if `include_slope = TRUE` and the term is omitted otherwise; $u_i \sim \mathcal{N}(0, \sigma_u^2)$ is a subject-specific random intercept; and the within-subject residual vector satisfies $\mathrm{Cov}(\varepsilon_i) = \sigma_e^2 R_i$.

When `use_ar1 = FALSE`, $R_i = I$. When `use_ar1 = TRUE`, the backend works with the residual *precision* matrix $C_i = R_i^{-1}$ over contiguous time blocks within subject and uses $\sigma_e^2 C_i^{-1}$ as the residual covariance.

AR(1) residual structure: Within each subject, paired observations are ordered by integer-coerced `time`. AR(1) correlation is applied only over strictly contiguous runs satisfying $t_{k+1} =$

$t_k + 1$. Gaps break the run. Negative times, and any isolated positions not belonging to a contiguous run, are treated as independent singletons.

For a contiguous run of length L and correlation parameter ρ, the block precision matrix is

$$C = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & & & \\ -\rho & 1+\rho^2 & -\rho & & \\ & \ddots & \ddots & \ddots & \\ & & -\rho & 1+\rho^2 & -\rho \\ & & & -\rho & 1 \end{bmatrix},$$

with a very small ridge added to the diagonal for numerical stability.

If `use_ar1 = TRUE` and `ar1_rho` is supplied, that value is used after validation and clipping to the admissible numerical range handled by the backend.

If `use_ar1 = TRUE` and `ar1_rho = NA`, the backend estimates `rho` separately for each analysed pair by:

1. fitting the corresponding iid model;
2. computing a moments-based lag-1 estimate from detrended residuals within contiguous blocks, used only as a seed; and
3. refining that seed by a short profile search over `rho` using the profiled REML log-likelihood.

In the exported `ba_rm()` wrapper, if an AR(1) fit for a given analysed pair fails specifically because the backend EM/GLS routine did not converge to admissible finite variance-component estimates, the wrapper retries that pair with iid residuals. If the iid refit succeeds, the final reported residual model for that pair is `"iid"` and a warning is issued. Other AR(1) failures are not simplified and are propagated as errors.

Internal centring and scaling for the proportional-bias slope: When `include_slope = TRUE`, the paired mean regressor is centred and scaled internally before fitting. Let $\bar{m}$ be the mean of the observed paired means. The backend chooses a scaling denominator from:

- the sample SD;
- the IQR-based scale $\mathrm{IQR}(m)/1.349$;
- the MAD-based scale $1.4826\,\mathrm{MAD}(m)$.

It uses the first of these that is not judged near-zero relative to the largest finite positive candidate scale, under a threshold proportional to $\sqrt{\epsilon_{\mathrm{mach}}}$. If all candidate scales are treated as near-zero, the fit stops with an error because the proportional-bias slope is not estimable on the observed paired-mean scale.

The returned `beta_slope` is back-transformed to the original paired-mean scale. The returned BA centre is the fitted mean difference at the centred reference paired mean $\bar{m}$, not the original-scale intercept coefficient.

Estimation: The backend uses a stabilised EM/GLS scheme.

Conditional on current variance components, the fixed effects are updated by GLS using the marginal precision of the paired differences after integrating out the random subject intercept. The resulting fixed-effect covariance used in the confidence-interval calculations is the GLS covariance

$$\mathrm{Var}(\hat{\beta} \mid \hat{\theta}) = \left(\sum_i X_i^\top V_i^{-1} X_i\right)^{-1}.$$

Given updated fixed effects, the variance components are refreshed by EM using the conditional moments of the subject random intercept and the residual quadratic forms. Variance updates are ratio-damped and clipped to admissible ranges for numerical stability.

Reported BA centre and limits of agreement: The reported BA centre is always model-based.

When `include_slope = FALSE`, it is the fitted intercept of the paired- difference mixed model.

When `include_slope = TRUE`, it is the fitted mean difference at the centred reference paired mean used internally by the backend.

The reported limits of agreement are

$$\mu_0 \pm \texttt{loa_multiplier}\sqrt{\sigma_u^2 + \sigma_e^2},$$

where μ_0 is the reported model-based BA centre. These LoA are for a single new paired difference from a random subject under the fitted model.

Under the implemented parameterisation, AR(1) correlation affects the off-diagonal within-subject covariance structure and therefore the estimation of the model parameters and their uncertainty, but not the marginal variance of a single paired difference. Consequently `rho` does not appear explicitly in the LoA point-estimate formula.

Confidence intervals: The backend returns Wald confidence intervals for the reported BA centre and for both LoA endpoints.

These intervals combine:

- the conditional GLS uncertainty in the fixed effects at the fitted covariance parameters; and
- a delta-method propagation of covariance-parameter uncertainty from the observed information matrix of the profiled REML log-likelihood.

The covariance-parameter vector is profiled on transformed scales: log-variances for σ_u^2 and σ_e^2, and, when `rho` is estimated internally under AR(1), a transformed correlation parameter mapped back by $\rho = 0.95\tanh(z)$.

Numerical central finite differences are used to approximate both the observed Hessian of the profiled REML log-likelihood and the gradients of the reported derived quantities. The resulting variances are combined and the final intervals are formed with the normal quantile corresponding to `conf_level`.

Exactly two methods versus ≥ 3 methods: With exactly two methods, at least two complete subject-time pairs are required; otherwise the function errors.

With ≥ 3 methods, the function analyses every unordered pair of method levels. For a given pair with fewer than two complete subject-time matches, that contrast is skipped and the corresponding matrix entries remain `NA`.

For a fitted contrast between methods in matrix positions (j, k) with $j < k$, the stored orientation is:

$$\texttt{bias}[j, k] \approx \text{method}_k - \text{method}_j.$$

Hence the transposed entry changes sign, while `sd_loa` and `width` are symmetric.

Identifiability and safeguards: Separate estimation of the residual and subject-level variance components requires sufficient complete within-subject replication after pairing. If the paired data are not adequate to separate these components, the fit stops with an identifiability error.

If the model is conceptually estimable but no finite positive pooled within-subject variance can be formed during initialisation, the backend uses $0.5 \times v_{\text{ref}}$ only as a temporary positive starting value

for the EM routine and records a warning string in the backend output. The exported wrapper does not otherwise modify the final estimates.

If the EM/GLS routine fails to reach admissible finite variance-component estimates, the backend throws an explicit convergence error rather than returning fallback estimates.

Value

Either a "ba_repeated" object (exactly two methods) or a "ba_repeated_matrix" object (≥ 3 methods).

If exactly two methods are supplied, the returned "ba_repeated" object is a list with components:

- means: numeric vector of paired means $(y_1 + y_2)/2$ used for plotting helpers.
- diffs: numeric vector of paired differences $y_2 - y_1$ used for plotting helpers.
- n_obs: integer number of complete subject-time pairs used.
- based.on: compatibility alias for n_obs.
- mean.diffs: scalar model-based BA centre. When include_slope = FALSE, this is the fitted intercept of the paired- difference model. When include_slope = TRUE, this is the fitted mean difference at the centred reference paired mean used internally by the backend.
- lower.limit, upper.limit: scalar limits of agreement, computed as $\mu_0 \pm$ loa_multiplier $\sqrt{\sigma_u^2 + \sigma_e^2}$.
- lines: named numeric vector with entries lower, mean, and upper.
- CI.lines: named numeric vector containing Wald confidence interval bounds for the bias and both LoA endpoints: mean.diff.ci.lower, mean.diff.ci.upper, lower.limit.ci.lower, lower.limit.ci.upper, upper.limit.ci.lower, upper.limit.ci.upper.
- loa_multiplier: scalar LoA multiplier actually used.
- critical.diff: scalar LoA half-width loa_multiplier $\times$ sd_loa.
- include_slope: logical, copied from the call.
- beta_slope: proportional-bias slope on the original paired-mean scale when include_slope = TRUE; otherwise NA.
- sigma2_subject: estimated variance of the subject-level random intercept on paired differences.
- sigma2_resid: estimated residual variance on paired differences.
- use_ar1: logical, copied from the call.
- residual_model: either "ar1" or "iid", indicating the final residual structure actually used.
- ar1_rho: AR(1) correlation actually used in the final fit when residual_model == "ar1"; otherwise NA.
- ar1_estimated: logical indicating whether ar1_rho was estimated internally (TRUE) or supplied by the user (FALSE) when the final residual model is AR(1); otherwise NA.

The confidence level is stored as attr(x, "conf.level").

If ≥ 3 methods are supplied, the returned "ba_repeated_matrix" object is a list with components:

- bias: numeric $m \times m$ matrix of model-based BA centres. For indices (j, k) with $j < k$, bias[j, k] estimates $\text{method}_k - \text{method}_j$. Thus the matrix orientation is **column minus row**, not row minus column. The diagonal is NA.

- sd_loa: numeric $m \times m$ matrix of LoA SDs, $\sqrt{\sigma_u^2 + \sigma_e^2}$. This matrix is symmetric.
- loa_lower, loa_upper: numeric $m \times m$ matrices of LoA endpoints corresponding to bias. These satisfy loa_lower$[j, k] = -$loa_upper$[k, j]$ and loa_upper$[j, k] = -$loa_lower$[k, j]$.
- width: numeric $m \times m$ matrix of LoA widths, loa_upper - loa_lower. This matrix is symmetric.
- n: integer $m \times m$ matrix giving the number of complete subject-time pairs used for each analysed contrast. Pairs with fewer than two complete matches are left as NA in the estimate matrices.
- mean_ci_low, mean_ci_high: numeric $m \times m$ matrices of Wald confidence interval bounds for bias.
- loa_lower_ci_low, loa_lower_ci_high: numeric $m \times m$ matrices of Wald confidence interval bounds for the lower LoA.
- loa_upper_ci_low, loa_upper_ci_high: numeric $m \times m$ matrices of Wald confidence interval bounds for the upper LoA.
- slope: optional numeric $m \times m$ matrix of proportional-bias slopes on the original paired-mean scale when include_slope = TRUE; otherwise NULL. This matrix is antisymmetric in sign because each fitted contrast is reversed across the transpose.
- methods: character vector of method levels defining matrix row and column order.
- loa_multiplier: scalar LoA multiplier actually used.
- conf_level: scalar confidence level used for the reported Wald intervals.
- use_ar1: logical, copied from the call.
- ar1_rho: scalar equal to the user-supplied common ar1_rho when use_ar1 = TRUE and a value was supplied; otherwise NA. This field does *not* store the per-pair estimated AR(1) parameters.
- residual_model: character $m \times m$ matrix whose entries are "ar1", "iid", or NA, indicating the final residual structure used for each pair.
- sigma2_subject: numeric $m \times m$ matrix of estimated subject-level random-intercept variances.
- sigma2_resid: numeric $m \times m$ matrix of estimated residual variances.
- ar1_rho_pair: optional numeric $m \times m$ matrix giving the AR(1) correlation actually used for each pair when the final residual model is "ar1"; otherwise NA for that entry. Present only when use_ar1 = TRUE.
- ar1_estimated: optional logical $m \times m$ matrix indicating whether the pair-specific ar1_rho_pair was estimated internally (TRUE) or supplied by the user (FALSE) for entries whose final residual model is "ar1"; otherwise NA. Present only when use_ar1 = TRUE.
- data_long: canonical long-form data frame with columns .response, .subject, .method, and .time retained for pairwise plotting helpers.
- mapping: named list mapping response, subject, method, and time to the canonical stored columns in data_long.

Author(s)

Thiago de Paula Oliveira

Examples

```
# -------- Simulate repeated-measures data --------
set.seed(1)

# design (no AR)
# subjects
S   <- 30L
# replicates per subject
Tm  <- 15L
subj <- rep(seq_len(S), each = Tm)
time <- rep(seq_len(Tm), times = S)

# subject signal centered at 0 so BA "bias" won't be driven by the mean level
mu_s  <- rnorm(S, mean = 0, sd = 8)
# constant within subject across replicates
true  <- mu_s[subj]

# common noise (no AR, i.i.d.)
sd_e <- 2
e0   <- rnorm(length(true), 0, sd_e)

# --- Methods ---
# M1: signal + noise
y1 <- true + e0

# M2: same precision as M1; here identical so M3 can be
#     almost perfectly the inverse of both M1 and M2
y2 <- y1 + rnorm(length(true), 0, 0.01)

# M3: perfect inverse of M1 and M2
y3 <- -y1   # = -(true + e0)

# M4: unrelated to all others (pure noise, different scale)
y4 <- rnorm(length(true), 3, 6)

data <- rbind(
  data.frame(y = y1, subject = subj, method = "M1", time = time),
  data.frame(y = y2, subject = subj, method = "M2", time = time),
  data.frame(y = y3, subject = subj, method = "M3", time = time),
  data.frame(y = y4, subject = subj, method = "M4", time = time)
)
data$method <- factor(data$method, levels = c("M1","M2","M3","M4"))

# quick sanity checks
with(data, {
  Y <- split(y, method)
  round(cor(cbind(M1 = Y$M1, M2 = Y$M2, M3 = Y$M3, M4 = Y$M4)), 3)
})

# Run BA (no AR)
ba4 <- ba_rm(
  data = data,
```

```
  response = "y", subject = "subject", method = "method", time = "time",
  loa_multiplier = 1.96, conf_level = 0.95,
  include_slope = FALSE, use_ar1 = FALSE
)
summary(ba4)
estimate(ba4)
tidy(ba4)
confint(ba4)
plot(ba4)

# -------- Simulate repeated-measures with AR(1) data --------
set.seed(123)
S <- 40L                      # subjects
Tm <- 50L                     # replicates per subject
methods <- c("A","B","C")     # N = 3 methods
rho <- 0.4                    # AR(1) within-subject across time

ar1_sim <- function(n, rho, sd = 1) {
  z <- rnorm(n)
  e <- numeric(n)
  e[1] <- z[1] * sd
  if (n > 1) for (t in 2:n) e[t] <- rho * e[t-1] + sqrt(1 - rho^2) * z[t] * sd
  e
}

# Subject baseline + time trend (latent "true" signal)
subj <- rep(seq_len(S), each = Tm)
time <- rep(seq_len(Tm), times = S)
# subject effects
mu_s  <- rnorm(S, 50, 7)
trend <- rep(seq_len(Tm) - mean(seq_len(Tm)), times = S) * 0.8
true  <- mu_s[subj] + trend

# Method-specific biases (B has +1.5 constant; C has slight proportional bias)
bias  <- c(A = 0, B = 1.5, C = -0.5)
# proportional component on "true"
prop  <- c(A = 0.00, B = 0.00, C = 0.10)

# Build long data: for each method, add AR(1) noise within subject over time
make_method <- function(meth, sd = 3) {
  e <- unlist(lapply(split(seq_along(time), subj),
                     function(ix) ar1_sim(length(ix), rho, sd)))
  y <- true * (1 + prop[meth]) + bias[meth] + e
  data.frame(y = y, subject = subj, method = meth, time = time,
             check.names = FALSE)
}

data <- do.call(rbind, lapply(methods, make_method))
data$method <- factor(data$method, levels = methods)

# -------- Repeated BA (pairwise matrix) ----------------------
baN <- ba_rm(
  response = data$y, subject = data$subject, method = data$method, time = data$time,
```

```
  loa_multiplier = 1.96, conf_level = 0.95,
  include_slope = FALSE,         # estimate proportional bias per pair
  use_ar1 = TRUE, ar1_rho = rho
)

# Matrices (row - column orientation)
print(baN)
summary(baN)
tidy(baN)

# Faceted BA scatter by pair
plot(baN, smoother = "lm", facet_scales = "free_y")

# -------- Two-method AR(1) path (A vs B only) ------------------------------
data_AB <- subset(data, method %in% c("A","B"))
baAB <- ba_rm(
  response = data_AB$y, subject = data_AB$subject,
  method = droplevels(data_AB$method), time = data_AB$time,
  include_slope = FALSE, use_ar1 = TRUE, ar1_rho = 0.4
)
print(baAB)
plot(baAB)
```

---

bcdcor *Pairwise Bias-Corrected Squared Distance Correlation*

Description

Computes the U-centred bias-corrected squared distance-correlation statistic for numeric columns using a high-performance C++ backend. Unlike conventional distance correlation, this finite-sample statistic may be negative. It is the statistic used by the Székely-Rizzo bias-corrected distance-correlation t-test.

Usage

```
bcdcor(
  data,
  na_method = c("error", "pairwise", "complete"),
  p_value = FALSE,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'bcdcor'
print(
```

```
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'bcdcor'
plot(
  x,
  title = "Bias-corrected squared distance correlation heatmap",
  low_color = "firebrick3",
  mid_color = "white",
  high_color = "steelblue1",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'bcdcor'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.bcdcor'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns are dropped. Columns must be numeric.

na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each association on its own pairwise complete-case overlap. "complete" performs listwise deletion once across the retained numeric columns.
p_value	Logical (default FALSE). If TRUE, attach pairwise p-values, test statistics, and degrees of freedom from the bias-corrected distance-correlation t-test.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates: "matrix", "sparse", or "edge_list".
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
x	An object of class summary.dcor.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
title	Plot title. Default is "Distance correlation heatmap".
low_color, mid_color, high_color	Colours used in the bcdcor() heatmap.
value_text_size	Font size for displaying values. Default is 4.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class dcor.

Details

Let $A^{(x)}$ and $A^{(y)}$ be U-centred distance matrices. The U-centred construction provides an unbiased estimator of squared distance covariance,

$$V_{n,U}^2(x,y) = \frac{1}{n(n-3)} \sum_{i \neq j} A_{ij}^{(x)} A_{ij}^{(y)}.$$

The bias-corrected squared distance-correlation statistic is

$$R_n^*(x,y) = \frac{V_{n,U}^2(x,y)}{\sqrt{V_{n,U}^2(x,x) V_{n,U}^2(y,y)}}.$$

This ratio is signed and may be negative in finite samples.

When `p_value = TRUE`, the t statistic is

$$T = \sqrt{M-1}\frac{R_n^*}{\sqrt{1-(R_n^*)^2}}, \quad M = \frac{n(n-3)}{2},$$

referenced to a Student t distribution with $M-1$ degrees of freedom. The signed statistic is not clipped before inference.

Value

A symmetric correlation result where the `(i, j)` entry is the signed U-centred bias-corrected squared distance-correlation statistic. The object has class `bcdcor`. When `p_value = TRUE`, the `inference` attribute contains `bcdcor`, `statistic`, `parameter`, and `p_value`.

Difference between `dcor()` and `bcdcor()`

`dcor()` returns conventional nonnegative distance correlation R_n. `dcor(squared = TRUE)` returns conventional R_n^2. `bcdcor()` returns the U-centred bias-corrected squared statistic used by the t-test. In finite samples, `bcdcor(x)` is not generally equal to `dcor(x)^2`.

Author(s)

Thiago de Paula Oliveira

References

Szekely, G. J., & Rizzo, M. L. (2013). The distance correlation t-test of independence. *Journal of Multivariate Analysis*, 117, 193-213.

See Also

`dcor()`, `robust_dcor()`

Examples

```
# Bias-corrected versus conventional squared dCor
set.seed(100)
n <- 150
x <- rnorm(n)
y <- 0.6 * x + sqrt(1 - 0.6^2) * rnorm(n)
X <- cbind(x = x, y = y)

# These are related but are not the same finite-sample estimator:
# dcor(..., squared = TRUE) uses the conventional V-statistic,
# whereas bcdcor() uses U-centering and bias correction.
dcor(X, squared = TRUE)
bcdcor(X)

# Bias-corrected squared dCor can be negative in finite samples,
# particularly under weak dependence or independence.
```

```
# This is not a negative-dependence estimate.
set.seed(3)
x <- rnorm(80)
y <- rnorm(80)
X <- cbind(x = x, y = y)

bcdcor(X)

# Bias-corrected dCor t-test
set.seed(2026)
n <- 80
x <- rnorm(n)
y <- x^2 + rnorm(n, sd = 0.25)
X <- cbind(x = x, y = y)

fit <- bcdcor(X, p_value = TRUE)

fit
summary(fit)

inf <- attr(fit, "inference")
inf$bcdcor
inf$statistic
inf$parameter
inf$p_value

# bcdcor() supports the same missing-data modes as dcor().
X_miss <- cbind(
  a = c(1, 2, NA, 4, 5, 6),
  b = c(2, 3, 4, 5, NA, 7),
  c = c(6, 5, 4, 3, 2, 1)
)

bcdcor(X_miss, na_method = "complete")
bcdcor(X_miss, na_method = "pairwise")

# Edge-list thresholding uses absolute magnitude and preserves signs.
set.seed(3)
X <- cbind(x = rnorm(80), y = rnorm(80), z = rnorm(80))

bcdcor(
  X,
  output = "edge_list",
  threshold = 0.001,
  diag = FALSE
)
```

---

bicor *Biweight mid-correlation (bicor)*

Description

Computes pairwise biweight mid-correlations for numeric data. Bicor is a robust, Pearson-like correlation that down-weights outliers and heavy-tailed observations. Optional large-sample confidence intervals are available as a derived feature.

Usage

```
bicor(
  data,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  c_const = 9,
  max_p_outliers = 1,
  pearson_fallback = c("hybrid", "none", "all"),
  mad_consistent = FALSE,
  w = NULL,
  sparse_threshold = NULL
)

diag.bicor(x, ...)

## S3 method for class 'bicor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  na_print = "NA",
  ...
)

## S3 method for class 'bicor'
plot(
  x,
  title = "Biweight mid-correlation heatmap",
  reorder = c("none", "hclust"),
  triangle = c("full", "lower", "upper"),
  low_color = "indianred1",
  mid_color = "white",
```

```
  high_color = "steelblue1",
  value_text_size = 3,
  ci_text_size = 2.5,
  show_value = TRUE,
  na_fill = "grey90",
  ...
)

## S3 method for class 'bicor'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.bicor'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

`data`	A numeric matrix or a data frame containing numeric columns. Factors, logicals and common time classes are dropped in the data-frame path. Missing values are not allowed unless `na_method = "pairwise"`.
`na_method`	Character scalar controlling missing-data handling. `"error"` rejects missing, `NaN`, and infinite values. `"pairwise"` recomputes each estimate on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. `"complete"` performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
`ci`	Logical (default `FALSE`). If `TRUE`, attach approximate Fisher-z confidence intervals for the off-diagonal biweight mid-correlations. This confidence interval is provided as an additional large-sample approximation.
`conf_level`	Confidence level used when `ci = TRUE`. Default is `0.95`.

`n_threads` Integer ≥ 1. Number of OpenMP threads. Defaults to `getOption("matrixCorr.threads", 1L)`.

`output` Output representation for the computed estimates.

- `"matrix"` (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code.
- `"sparse"`: sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding.
- `"edge_list"`: long-form data frame with columns `row`, `col`, `value`; convenient for filtering, joins, and network-style workflows.

`threshold` Non-negative absolute-value filter for non-matrix outputs: keep entries with `abs(value) >= threshold`. Use `threshold > 0` when you want only stronger associations (typically with `output = "sparse"` or `"edge_list"`). Keep `threshold = 0` to retain all values. Must be `0` when `output = "matrix"`.

`diag` Logical; whether to include diagonal entries in `"sparse"` and `"edge_list"` outputs.

`c_const` Positive numeric. Tukey biweight tuning constant applied to the *raw* MAD; default `9` (Langfelder & Horvath's convention).

`max_p_outliers` Numeric in `(0, 1]`. Optional cap on the maximum proportion of outliers *on each side*; if `< 1`, side-specific rescaling maps those quantiles to `|u|=1`. Use `1` to disable.

`pearson_fallback`
Character scalar indicating the fallback policy. One of:

- `"hybrid"` (default): if a column has MAD = 0, that column uses Pearson standardisation, yielding a hybrid correlation.
- `"none"`: return `NA` if a column has MAD = 0 or becomes degenerate after weighting.
- `"all"`: force ordinary Pearson for all columns.

`mad_consistent` Logical; if `TRUE`, use the normal-consistent MAD (`MAD_raw * 1.4826`) in the bicor weights. Default `FALSE` to match Langfelder & Horvath (2012).

`w` Optional non-negative numeric vector of length `nrow(data)` giving *row weights*. When supplied, weighted medians/MADs are used and Tukey weights are multiplied by `w` before normalisation.

`sparse_threshold`
Optional numeric ≥ 0. If supplied, sets entries with `|r| < sparse_threshold` to 0 and returns a sparse `"ddiMatrix"` (requires **Matrix**).

`x` An object of class `summary.bicor`.

`...` Additional arguments passed to `ggplot2::theme()` or other layers.

`digits` Integer; number of decimal places used for the matrix.

`n` Optional row threshold for compact preview output.

`topn` Optional number of leading/trailing rows to show when truncated.

`max_vars` Optional maximum number of visible columns; `NULL` derives this from console width.

`width` Optional display width; defaults to `getOption("width")`.

ci_digits	Integer; digits for bicor confidence limits in the pairwise summary.
show_ci	One of "yes" or "no".
na_print	Character; how to display missing values.
title	Plot title. Default is "Biweight mid-correlation heatmap".
reorder	Character; one of "none" (default) or "hclust". If "hclust", variables are reordered by complete-linkage clustering on the distance $d = 1-r$, after replacing NA by 0 for clustering purposes only.
triangle	One of "full" (default), "lower", or "upper" to display the full matrix or a single triangle.
low_color, mid_color, high_color	Colours for the gradient in scale_fill_gradient2. Defaults are "indianred1", "white", "steelblue1".
value_text_size	Numeric; font size for cell labels. Set to NULL to suppress labels (recommended for large matrices).
ci_text_size	Text size for confidence-interval labels in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
na_fill	Fill colour for NA cells. Default "grey90".
object	An object of class bicor.
p_digits	Integer; digits for bicor p-values in the pairwise summary.

Details

For a column $x = (x_a)_{a=1}^m$, let $\mathrm{med}(x)$ be the median and $\mathrm{MAD}(x) = \mathrm{med}(|x - \mathrm{med}(x)|)$ the (raw) median absolute deviation. If mad_consistent = TRUE, the consistent scale $\mathrm{MAD}^\star(x) = 1.4826\,\mathrm{MAD}(x)$ is used. With tuning constant $c > 0$, define

$$u_a = \frac{x_a - \mathrm{med}(x)}{c\,\mathrm{MAD}^{(\star)}(x)}.$$

The Tukey biweight gives per-observation weights

$$w_a = (1 - u_a^2)^2\,\mathbf{1}\{|u_a| < 1\}.$$

Robust standardisation of a column is

$$\tilde{x}_a = \frac{(x_a - \mathrm{med}(x))\,w_a}{\sqrt{\sum_{b=1}^m \left[(x_b - \mathrm{med}(x))\,w_b\right]^2}}.$$

For two columns x, y, the biweight mid-correlation is

$$\mathrm{bicor}(x, y) = \sum_{a=1}^m \tilde{x}_a\,\tilde{y}_a \in [-1, 1].$$

Capping the maximum proportion of outliers (max_p_outliers). If max_p_outliers < 1, let $q_L = Q_x(\text{max_p_outliers})$ and $q_U = Q_x(1 - \text{max_p_outliers})$ be the lower/upper quantiles of x.

If the corresponding $|u|$ at either quantile exceeds 1, u is rescaled *separately* on the negative and positive sides so that those quantiles land at $|u| = 1$. This guarantees that all observations between the two quantiles receive positive weight. Note the bound applies per side, so up to 2 max_p_outliers of observations can be treated as outliers overall.

Fallback when for zero MAD / degeneracy (`pearson_fallback`). If a column has $\mathrm{MAD} = 0$ or the robust denominator becomes zero, the following rules apply:

- `"none"` when correlations involving that column are `NA` (diagonal remains 1).
- `"hybrid"` when only the affected column switches to Pearson standardisation $\bar{x}_a = (x_a - \overline{x})/\sqrt{\sum_b (x_b - \overline{x})^2}$, yielding the hybrid correlation

$$\mathrm{bicor}_{\mathrm{hyb}}(x, y) = \sum_a \bar{x}_a \, \tilde{y}_a,$$

  with the other column still robust-standardised.
- `"all"` when all columns use ordinary Pearson standardisation; the result equals `stats::cor(..., method="pearson")` when the NA policy matches.

Handling missing values (`na_method`).

- `"error"` (default): inputs must be finite; this yields a symmetric, positive semidefinite (PSD) matrix since $R = \tilde{X}^\top \tilde{X}$.
- `"pairwise"`: each R_{jk} is computed on the intersection of rows where both columns are finite. Pairs with fewer than 5 overlapping rows return `NA` (guarding against instability). Pairwise deletion can break PSD, as in the Pearson case.

Row weights (`w`). When `w` is supplied (non-negative, length m), the weighted median $\mathrm{med}_w(x)$ and weighted MAD $\mathrm{MAD}_w(x) = \mathrm{med}_w(|x - \mathrm{med}_w(x)|)$ are used to form u. The Tukey weights are then multiplied by the observation weights prior to normalisation:

$$\tilde{x}_a = \frac{(x_a - \mathrm{med}_w(x))\, w_a \, w_a^{(\mathrm{obs})}}{\sqrt{\sum_b \left[(x_b - \mathrm{med}_w(x))\, w_b \, w_b^{(\mathrm{obs})}\right]^2}},$$

where $w_a^{(\mathrm{obs})} \geq 0$ are the user-supplied row weights and w_a are the Tukey biweights built from the weighted median/MAD. Weighted pairwise behaves analogously on each column pair's overlap.

MAD choice (`mad_consistent`). Setting `mad_consistent = TRUE` multiplies the raw MAD by 1.4826 inside u. Equivalently, it uses an effective tuning constant $c^\star = c \times 1.4826$. The default `FALSE` reproduces the convention in Langfelder & Horvath (2012).

Optional sparsification (`sparse_threshold`). If provided, entries with $|r| <$ sparse_threshold are set to 0 and the result is returned as a `"ddiMatrix"` (diagonal is forced to 1). This is a post-processing step that does not alter the per-pair estimates.

Computation and threads. Columns are robust-standardised in parallel and the matrix is formed as $R = \tilde{X}^\top \tilde{X}$. `n_threads` selects the number of OpenMP threads; by default it uses `getOption("matrixCorr.threads", 1L)`.

Large-sample inference. For a pairwise estimate r computed from n_{jk} observed rows, the standard large-sample summaries use

$$Z_{jk} = \frac{1}{2} \log\left(\frac{1 + r}{1 - r}\right) \sqrt{n_{jk} - 2}$$

and

$$T_{jk} = \sqrt{n_{jk} - 2}\, \frac{|r|}{\sqrt{1 - r^2}}.$$

The reported p-value is the two-sided Student-t tail probability with $n_{jk} - 2$ degrees of freedom. When `ci = TRUE`, the package also reports an approximate Fisher-z confidence interval obtained from

$$z_{jk} = \operatorname{atanh}(r), \qquad \mathrm{SE}(z_{jk}) = \frac{1}{\sqrt{n_{jk} - 3}},$$

followed by back-transformation with `tanh()`. Confidence intervals are currently available only for dense, unweighted outputs.

Basic properties. $\operatorname{bicor}(ax + b,\ cy + d) = \operatorname{sign}(ac)\operatorname{bicor}(x, y)$. With no missing data (and with per-column hybrid/robust standardisation), the output is symmetric and PSD. As with Pearson, affine equivariance does not hold for the associated biweight midcovariance.

Value

A symmetric correlation matrix with class `bicor` (or a `dgCMatrix` if `sparse_threshold` is used), with attributes: `method = "biweight_mid_correlation"`, `description`, and `package = "matrixCorr"`. Downstream code should be prepared to handle either a dense numeric matrix or a sparse `dgCMatrix`. When `ci = TRUE`, the object also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, `conf.level`, and `ci.method`, together with an `inference` attribute containing the standard large-sample summary matrices `estimate`, `statistic`, `p_value`, `Z`, and `n_obs`. Pairwise complete-case counts are stored in `attr(x, "diagnostics")$n_complete`. Internally, all medians/MADs, Tukey weights, optional pairwise-NA handling, and OpenMP loops are implemented in the C++ helpers (`bicor_*_cpp()`), so the R wrapper mostly validates arguments and dispatches to the appropriate backend.

Invisibly returns `x`.

A `ggplot` object.

Author(s)

Thiago de Paula Oliveira

References

Langfelder, P. & Horvath, S. (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. doi:10.18637/jss.v046.i11

Examples

```
set.seed(1)
X <- matrix(rnorm(2000 * 40), 2000, 40)
R <- bicor(X, c_const = 9, max_p_outliers = 1,
                       pearson_fallback = "hybrid")
print(attr(R, "method"))
summary(R)
R_ci <- bicor(X[, 1:5], ci = TRUE)
summary(R_ci)
estimate(R_ci)
tidy(R_ci)
```

```
ci(R_ci)
confint(R_ci)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(R)
}
```

`biserial` *Biserial Correlation Between Continuous and Binary Variables*

Description

Computes biserial correlations between continuous variables in `data` and binary variables in `y`. Both pairwise vector mode and rectangular matrix/data-frame mode are supported.

Usage

```
biserial(data, y, na_method = c("error", "pairwise"), ci = FALSE, p_value = FALSE,
  conf_level = 0.95, ...)

## S3 method for class 'biserial_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'biserial_corr'
plot(
  x,
  title = "Biserial correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

## S3 method for class 'biserial_corr'
```

```
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.biserial_corr'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric vector, matrix, or data frame containing continuous variables.
y	A binary vector, matrix, or data frame. In data-frame mode, only two-level columns are retained.
na_method	Character scalar controlling missing-data handling. "error" rejects missing values. "pairwise" uses pairwise complete cases.
ci	Logical (default FALSE). If TRUE, attach approximate large-sample confidence intervals derived from a Fisher z-transformation of the biserial estimate.
p_value	Logical (default FALSE). If TRUE, attach model-based large-sample p-values, test statistics, and degrees of freedom for each biserial estimate.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
...	Additional arguments passed to print().
x	An object of class summary.biserial_corr.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".

`title`	Plot title. Default is `"Biserial correlation heatmap"`.
`low_color`	Color for the minimum correlation.
`high_color`	Color for the maximum correlation.
`mid_color`	Color for zero correlation.
`value_text_size`	Font size used in tile labels.
`ci_text_size`	Text size for confidence intervals in the heatmap.
`show_value`	Logical; if `TRUE` (default), overlay numeric values on the heatmap tiles.
`object`	An object of class `biserial_corr`.
`ci_digits`	Integer; digits for biserial confidence limits in the pairwise summary.
`p_digits`	Integer; digits for biserial p-values in the pairwise summary.

Details

The biserial correlation is the special two-category case of the polyserial model. It assumes that a binary variable Y arises by thresholding an unobserved standard-normal variable Z that is jointly normal with a continuous variable X. Writing $p = P(Y = 1)$ and $q = 1 - p$, let $z_p = \Phi^{-1}(p)$ and $\phi(z_p)$ be the standard-normal density evaluated at z_p. If $\bar{x}_1$ and $\bar{x}_0$ denote the sample means of X in the two observed groups and s_x is the sample standard deviation of X, the usual biserial estimator is

$$r_b = \frac{\bar{x}_1 - \bar{x}_0}{s_x} \frac{pq}{\phi(z_p)}.$$

This is exactly the estimator implemented in the underlying C++ kernel.

Assumptions. The biserial coefficient is appropriate when the observed binary variable is viewed as a thresholded version of an unobserved continuous latent variable that is jointly normal with the observed continuous variable. The optional p-values and confidence intervals adopt this latent-normal interpretation together with the usual large-sample approximations used for correlation coefficients. These inferential quantities are therefore model-based and should not be interpreted as distribution-free summaries.

Inference. When `p_value = TRUE`, the package reports the large-sample t-statistic

$$t = r_b \sqrt{\frac{n-2}{1-r_b^2}},$$

referenced to a Student t-distribution with $n - 2$ degrees of freedom. When `ci = TRUE`, the package forms an approximate Fisher z-interval by transforming r_b with $z = \operatorname{atanh}(r_b)$, using standard error $1/\sqrt{n-3}$, and mapping the limits back with $\tanh(\cdot)$. The CI is therefore an internal large-sample extension and is only computed when explicitly requested.

In vector mode a single biserial correlation is returned. In matrix/data-frame mode, every numeric column of `data` is paired with every binary column of `y`, producing a rectangular matrix of continuous-by-binary biserial correlations.

Unlike the point-biserial correlation, which is just Pearson correlation on a 0/1 coding of the binary variable, the biserial coefficient explicitly assumes an underlying latent normal threshold model and rescales the mean difference accordingly.

Computational complexity. If `data` has p_x continuous columns and `y` has p_y binary columns, the matrix path computes $p_x p_y$ closed-form estimates with negligible extra memory beyond the output matrix.

Value

If both `data` and `y` are vectors, a numeric scalar. Otherwise a numeric matrix of class `biserial_corr` with rows corresponding to the continuous variables in `data` and columns to the binary variables in `y`. Matrix outputs carry attributes `method`, `description`, and `package = "matrixCorr"`. When `p_value = TRUE`, the object also carries an `inference` attribute with matrices `estimate`, `statistic`, `parameter`, `p_value`, and `n_obs`. When `ci = TRUE`, it additionally carries a `ci` attribute with matrices `lwr.ci` and `upr.ci`, plus `attr(x, "conf.level")`. Scalar outputs keep the same point estimate and gain the same metadata only when inference is requested.

Author(s)

Thiago de Paula Oliveira

References

Olsson, U., Drasgow, F., & Dorans, N. J. (1982). The polyserial correlation coefficient. *Psychometrika*, 47(3), 337-347.

Fisher, R. A. (1921). On the probable error of a coefficient of correlation deduced from a small sample. *Metron*, 1, 3-32.

Examples

```
set.seed(126)
n <- 1000
Sigma <- matrix(c(
  1.00, 0.35, 0.50, 0.25,
  0.35, 1.00, 0.30, 0.55,
  0.50, 0.30, 1.00, 0.40,
  0.25, 0.55, 0.40, 1.00
), 4, 4, byrow = TRUE)

Z <- mnormt::rmnorm(n = n, mean = rep(0, 4), varcov = Sigma)
X <- data.frame(x1 = Z[, 1], x2 = Z[, 2])
Y <- data.frame(
  g1 = Z[, 3] > stats::qnorm(0.65),
  g2 = Z[, 4] > stats::qnorm(0.55)
)

bs <- biserial(X, Y, ci = TRUE, p_value = TRUE)
print(bs, digits = 3)
summary(bs)
estimate(bs)
tidy(bs)
ci(bs)
confint(bs)
plot(bs)
```

ccc *Pairwise Lin's concordance correlation coefficient*

Description

Computes all pairwise Lin's Concordance Correlation Coefficients (CCC) from the numeric columns of a matrix or data frame. CCC measures both precision (Pearson correlation) and accuracy (closeness to the 45-degree line). This function is backed by a high-performance 'C++' implementation.

Lin's CCC quantifies the concordance between a new test/measurement and a gold-standard for the same variable. Like a correlation, CCC ranges from -1 to 1 with perfect agreement at 1, and it cannot exceed the absolute value of the Pearson correlation between variables. It can be legitimately computed even with small samples (e.g., 10 observations), and results are often similar to intraclass correlation coefficients. CCC provides a single summary of agreement, but it may not capture systematic bias; a Bland-Altman plot (differences vs. means) is recommended to visualize bias, proportional trends, and heteroscedasticity (see ba).

Usage

```
ccc(
  data,
  ci = FALSE,
  conf_level = 0.95,
  na_method = c("error", "complete", "pairwise"),
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  verbose = FALSE
)

## S3 method for class 'ccc'
print(
  x,
  digits = 4,
  ci_digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ccc'
summary(
  object,
```

```
  digits = 4,
  ci_digits = 2,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.ccc'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'ccc'
plot(
  x,
  title = "Lin's Concordance Correlation Heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

data	A numeric matrix or data frame with at least two numeric columns. Non-numeric columns will be ignored.
ci	Logical; if TRUE, return lower and upper confidence bounds
conf_level	Confidence level for CI, default = 0.95
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "complete" removes rows incomplete in any retained numeric column before computing all CCC estimates. "pairwise" recomputes each method pair on its own complete finite overlap.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).

output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
verbose	Logical; if TRUE, prints how many threads are used
x	An object of class "ccc" (either a matrix or a list with CIs).
digits	Integer; decimals for CCC estimates (default 4).
ci_digits	Integer; decimals for CI bounds (default 2).
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Passed to ggplot2::theme().
object	A "ccc" or "ccc_ci" object to summarize.
title	Title for the plot.
low_color	Color for low CCC values.
high_color	Color for high CCC values.
mid_color	Color for mid CCC values.
value_text_size	Text size for CCC values in the heatmap.
ci_text_size	Text size for confidence intervals.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.

Details

Lin's CCC is defined as

$$\rho_c \;=\; \frac{2\,\mathrm{cov}(X,Y)}{\sigma_X^2+\sigma_Y^2+(\mu_X-\mu_Y)^2},$$

where μ_X, μ_Y are the means, σ_X^2, σ_Y^2 the variances, and $\mathrm{cov}(X,Y)$ the covariance. Equivalently,

$$\rho_c \;=\; r \times C_b, \qquad r \;=\; \frac{\mathrm{cov}(X,Y)}{\sigma_X\sigma_Y}, \quad C_b \;=\; \frac{2\sigma_X\sigma_Y}{\sigma_X^2+\sigma_Y^2+(\mu_X-\mu_Y)^2}.$$

Hence $|\rho_c| \leq |r| \leq 1$, $\rho_c = r$ iff $\mu_X = \mu_Y$ and $\sigma_X = \sigma_Y$, and $\rho_c = 1$ iff, in addition, $r = 1$. CCC is symmetric in (X, Y) and penalises both location and scale differences; unlike Pearson's r, it is not invariant to affine transformations that change means or variances.

When `ci = TRUE`, large-sample confidence intervals for ρ_c are returned for each pair. The implementation uses Lin's delta-method standard error and then forms limits on a Fisher-z transformed CCC scale before mapping back to $[-1, 1]$. For speed, CIs are omitted when `ci = FALSE`.

If either variable has zero variance, ρ_c is undefined and `NA` is returned for that pair (including the diagonal).

Missing-data handling follows the same contract as `pearson_corr` and `spearman_rho`. With `na_method = "error"` (default), missing and non-finite values are rejected. With `"complete"`, rows incomplete in any retained numeric column are removed once before all pairwise CCC estimates are computed. With `"pairwise"`, each method pair is computed on its own complete finite overlap, and `n_complete` diagnostics may vary by pair. Pairwise CIs require at least three complete observations for that pair.

Negative CCC estimates are fully supported. Matrix output preserves the signed estimate, while thresholded `"sparse"` and `"edge_list"` outputs retain entries according to `abs(CCC) >= threshold`.

Probability of agreement, available through `prob_agree`, answers a different question from CCC. CCC is a coefficient-based summary of concordance between paired measurements. `prob_agree()` follows Stevens and Anderson-Cook (2017) and estimates the probability that two estimated quantities or curves differ by no more than a user-specified practical tolerance.

Value

A symmetric numeric matrix with class `"ccc"` and attributes:

- `method`: The method used ("Lin's concordance")
- `description`: Description string

If `ci = FALSE`, returns matrix of class `"ccc"`. If `ci = TRUE`, returns a list with elements: `est`, `lwr.ci`, `upr.ci`.

For `summary.ccc`, a data frame with columns `item1`, `item2`, `estimate`, and (optionally) `lwr`, `upr`, plus `n_complete` when available.

Author(s)

Thiago de Paula Oliveira

References

Lin L (1989). A concordance correlation coefficient to evaluate reproducibility. Biometrics 45: 255-268.

Lin L (2000). A note on the concordance correlation coefficient. Biometrics 56: 324-325.

Bland J, Altman D (1986). Statistical methods for assessing agreement between two methods of clinical measurement. The Lancet 327: 307-310.

See Also

print.ccc, plot.ccc, ba, prob_agree, and cia. See robust_ccc for MCD based robust concordance using joint reweighted location and scatter estimates. ccc() answers the question "How well do two paired measurements agree overall, accounting for both correlation and mean/scale bias?". In contrast, cia() answers "Are two or more methods interchangeable at the individual level relative to within-method replicate disagreement?".

For repeated measurements look at ccc_rm_reml, ccc_rm_ustat or ba_rm

Examples

```
# Example with multivariate normal data
Sigma <- matrix(c(1, 0.5, 0.3,
                  0.5, 1, 0.4,
                  0.3, 0.4, 1), nrow = 3)
mu <- c(0, 0, 0)
set.seed(123)
mat_mvn <- MASS::mvrnorm(n = 100, mu = mu, Sigma = Sigma)
result_mvn <- ccc(mat_mvn, ci = TRUE)
print(result_mvn)
summary(result_mvn)
estimate(result_mvn)
tidy(result_mvn)
ci(result_mvn)
confint(result_mvn)
plot(result_mvn)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(result_mvn)
}
```

---

ccc_glmm *Poisson GLMM concordance correlation for count agreement*

Description

Computes pairwise generalized concordance correlation coefficients (CCC) for count outcomes measured repeatedly by two or more methods, observers, or devices. The current implementation fits Poisson-log generalized linear mixed models and reports total, inter-method, and intra-method agreement summaries from the fitted mean and variance components.

Usage

```
ccc_glmm(
  data,
  response,
  subject,
```

```
  method,
  replicate = NULL,
  family = "poisson",
  link = "log",
  overdispersion = c("none", "pearson"),
  include_subject_method = FALSE,
  ci = FALSE,
  conf_level = 0.95,
  max_iter = 1000,
  tol = 1e-08,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE
)
```

Arguments

`data`	A data frame containing the measurements.
`response`	Character. Name of the non-negative integer-like count column.
`subject`	Character. Name of the subject identifier column.
`method`	Character. Name of the method/observer column.
`replicate`	Optional character. Name of the replicate column. If `NULL`, one replicate per subject-method cell is assumed.
`family`	Character. Currently only `"poisson"` is implemented.
`link`	Character. Currently only `"log"` is implemented.
`overdispersion`	Character. `"none"` fixes $\phi = 1$; `"pearson"` estimates ϕ from post-fit Pearson residuals.
`include_subject_method`	Logical. If `TRUE`, include an additional subject-by-method random intercept variance component.
`ci`	Logical. If `TRUE`, compute delta-method confidence intervals for the total, inter-method, and intra-method CCC estimates.
`conf_level`	Confidence level for delta-method confidence intervals.
`max_iter`	Positive integer. Maximum optimiser iterations.
`tol`	Positive numeric convergence tolerance.
`n_threads`	Integer ≥ 1. Number of OpenMP threads to make available to compiled code.
`verbose`	Logical. If `TRUE`, emit progress messages.

Details

The fitted model for a pair of methods is

$$Y_{ijl} \mid \alpha_i, \gamma_{ij} \sim \text{Poisson}(\mu_{ijl}), \qquad \log(\mu_{ijl}) = \eta_j + \alpha_i + \gamma_{ij},$$

where i indexes subjects, j indexes methods, and l indexes replicate readings. The subject effect α_i captures between-subject heterogeneity. When `include_subject_method = TRUE`, γ_{ij} captures

subject-specific method departures; otherwise it is fixed at zero. The fixed method difference contributes to disagreement through `sigma2_method`.

The model is fitted by marginal maximum likelihood using Gauss-Hermite quadrature. The random-intercept model uses 40 quadrature points. The subject-by-method model uses tensor-product quadrature with fewer points per dimension because it integrates over a three-dimensional subject block. The reported CCC quantities are then computed from the fitted Poisson-log mean and variance components.

The main matrix value, `rho_ccc`, is the total agreement coefficient. Use it as the primary overall count-agreement summary when each individual reading is the unit of inference. It penalizes lack of between-subject signal, systematic method bias, subject-by-method disagreement, and Poisson residual variation.

`rho_ccc_inter` is the inter-method agreement coefficient for averages over replicated readings. It is useful when decisions are based on the mean of m replicate readings per subject-method cell. Replication reduces only the residual count variation term; it does not dilute systematic method disagreement or between-subject variation.

`rho_ccc_intra_method1` and `rho_ccc_intra_method2` are method-specific repeatability coefficients. Use them to diagnose whether one method is internally more repeatable than the other on the count scale. These are not direct method-comparison coefficients; they describe within-method reproducibility after accounting for the fitted mean and random effects.

`precision` isolates the share of non-systematic variation attributable to subject ranking/heterogeneity, while `accuracy` is the ratio `rho_ccc / precision`. Low accuracy with reasonable precision indicates that disagreement is driven mainly by method bias or extra method-specific variation rather than poor subject discrimination.

`overdispersion = "none"` fixes $\phi = 1$, the Poisson model. `overdispersion = "pearson"` replaces the residual term by a Pearson dispersion estimate. Use the Pearson adjustment as a sensitivity analysis when counts appear more variable than the Poisson model allows; it should generally reduce CCC when extra-Poisson variation is present.

Confidence intervals:

When `ci = TRUE`, large-sample confidence intervals are computed for `rho_ccc`, `rho_ccc_inter`, `rho_ccc_intra_method1`, and `rho_ccc_intra_method2`. The implementation uses a delta-method standard error based on the fitted GLMM parameter vector and the inverse Hessian of the marginal negative log-likelihood:

$$\mathrm{Var}\{g(\hat{\theta})\} \approx \nabla g(\hat{\theta})^{\top} \mathrm{Var}(\hat{\theta}) \nabla g(\hat{\theta}),$$

where $g(\theta)$ is the relevant CCC function. Gradients are evaluated numerically by central finite differences. The reported point estimates are not replaced by $g(\hat{\theta})$; they remain the values from the standard point-estimate path.

Intervals are formed on a Fisher-Z transformed CCC scale, $z = 0.5 \log\{(1+\rho)/(1-\rho)\}$, and then back-transformed to the CCC scale. This is the same broad strategy used elsewhere for CCC-style intervals because it is usually more stable than a raw Wald interval near the boundaries. CI limits are not forcibly truncated to $[0, 1]$.

For `overdispersion = "pearson"`, ϕ is a post-fit Pearson dispersion estimate rather than a likelihood parameter. Delta-method confidence intervals therefore treat ϕ as fixed and a warning is issued.

This function currently implements the Poisson-log count-data case. The `family` and `link` arguments are present for API stability and future extensions, but only `family = "poisson"` and

`link = "log"` are currently supported. The returned estimates are variance-component CCCs constrained to $[0, 1]$; they are not Lin's raw moment CCC and should not be expected to produce negative values.

Value

A symmetric numeric matrix of class `c("ccc_glmm", "ccc")` containing `rho_ccc`. Additional pairwise matrices are stored as attributes:

- `rho_ccc_inter`: agreement for replicated method averages.
- `rho_ccc_intra_method1`, `rho_ccc_intra_method2`: method-specific repeatability coefficients.
- `sigma2_subject`, `sigma2_method`, `sigma2_subject_method`: fitted variance/disagreement components.
- `phi`, `mu`, `precision`, `accuracy`: fitted count-scale diagnostics used in the CCC decomposition.
- `beta0`, `beta_method`, `nll`, `logLik`, `convergence_code`: fitting diagnostics.
- `n_obs`, `n_subjects`, `m_reps`: design diagnostics.
- when `ci = TRUE`, standard error and confidence-limit matrices for `rho_ccc`, `rho_ccc_inter`, `rho_ccc_intra_method1`, and `rho_ccc_intra_method2`.

References

Carrasco JL (2010). A generalized concordance correlation coefficient based on the variance components generalized linear mixed models for overdispersed count data. *Biometrics*.

Examples

```
# Example 1: overall agreement for two Poisson count methods.
# Here the methods have similar means, so rho_ccc is mainly driven by
# between-subject count heterogeneity versus Poisson residual noise.
set.seed(1)
df1 <- expand.grid(
  subject = factor(seq_len(12)),
  method = factor(c("A", "B")),
  replicate = factor(seq_len(2))
)
subject_eff <- rnorm(12, 0, 0.5)
df1$eta <- 1.1 + subject_eff[as.integer(df1$subject)]
df1$y <- rpois(nrow(df1), exp(df1$eta))

fit1 <- ccc_glmm(df1, "y", "subject", "method", replicate = "replicate",
ci = TRUE)
fit1
summary(fit1)
estimate(fit1)
tidy(fit1)
ci(fit1)
confint(fit1)
```

```
# Example 2: method bias lowers total agreement.
# This tests whether a systematic method shift is reflected in rho_ccc
# and the accuracy component.
set.seed(2)
df2 <- expand.grid(
  subject = factor(seq_len(12)),
  method = factor(c("A", "B")),
  replicate = factor(seq_len(2))
)
subject_eff <- rnorm(12, 0, 0.6)
df2$eta <- 1.0 + subject_eff[as.integer(df2$subject)] +
  ifelse(df2$method == "B", 0.6, 0)
df2$y <- rpois(nrow(df2), exp(df2$eta))
fit2 <- ccc_glmm(df2, "y", "subject", "method", replicate = "replicate")
summary(fit2)

# Example 3: subject-by-method variation.
# This tests whether individual subjects respond differently by method.
# The subject-method component is useful when disagreement is not explained
# by a single fixed method bias.
set.seed(3)
df3 <- expand.grid(
  subject = factor(seq_len(10)),
  method = factor(c("A", "B")),
  replicate = factor(seq_len(2))
)
subject_eff <- rnorm(10, 0, 0.4)
subject_method_eff <- matrix(rnorm(20, 0, 0.25), nrow = 10)
method_id <- as.integer(df3$method)
df3$eta <- 1.1 + subject_eff[as.integer(df3$subject)] +
  subject_method_eff[cbind(as.integer(df3$subject), method_id)]
df3$y <- rpois(nrow(df3), exp(df3$eta))

fit3 <- ccc_glmm(
  df3, "y", "subject", "method",
  replicate = "replicate",
  include_subject_method = TRUE
)
attr(fit3, "sigma2_subject_method")


# Example 4: four methods.
# This tests pairwise agreement across several count methods and helps
# identify which method pairs have the strongest total CCC.
set.seed(4)
df4 <- expand.grid(
  subject = factor(seq_len(10)),
  method = factor(c("A", "B", "C", "D")),
  replicate = factor(seq_len(2))
)
subject_eff <- rnorm(10, 0, 0.5)
method_bias <- c(A = 0, B = 0.1, C = 0.4, D = -0.2)
```

```
df4$eta <- 1.0 + subject_eff[as.integer(df4$subject)] +
  method_bias[as.character(df4$method)]
df4$y <- rpois(nrow(df4), exp(df4$eta))
fit4 <- ccc_glmm(df4, "y", "subject", "method", replicate = "replicate")
fit4
summary(fit4, n = 3)
```

`ccc_rm_reml` *Concordance Correlation via REML (Linear Mixed-Effects Model)*

Description

Compute Lin's Concordance Correlation Coefficient (CCC) from a linear mixed-effects model fitted by REML. The fixed-effects part can include `method` and/or `time` (optionally their interaction), with a subject-specific random intercept to capture between-subject variation. Large $n \times n$ inversions are avoided by solving small per-subject systems.

Assumption: time levels are treated as *regular, equally spaced* visits indexed by their order within subject. The AR(1) residual model is in discrete time on the visit index (not calendar time). `NA` time codes break the serial run. Gaps in the factor levels are *ignored* (adjacent observed visits are treated as lag-1).

Usage

```
ccc_rm_reml(
  data,
  response,
  subject,
  method = NULL,
  time = NULL,
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  ci_mode = c("auto", "raw", "logit"),
  verbose = FALSE,
  digits = 4,
  use_message = TRUE,
  interaction = FALSE,
  max_iter = 100,
  tol = 1e-06,
  Dmat = NULL,
  Dmat_type = c("time-avg", "typical-visit", "weighted-avg", "weighted-sq"),
  Dmat_weights = NULL,
  Dmat_rescale = TRUE,
  ar = c("none", "ar1"),
  ar_rho = NA_real_,
  slope = c("none", "subject", "method", "custom"),
```

```
  slope_var = NULL,
  slope_Z = NULL,
  drop_zero_cols = TRUE,
  vc_select = c("auto", "none"),
  vc_alpha = 0.05,
  vc_test_order = c("subj_time", "subj_method"),
  include_subj_method = NULL,
  include_subj_time = NULL,
  sb_zero_tol = 1e-10
)
```

Arguments

data	A data frame.
response	Character. Response variable name.
subject	Character. Subject ID variable name.
method	Character or NULL. Optional column name of method factor (added to fixed effects).
time	Character or NULL. Optional column name of time factor (added to fixed effects).
ci	Logical. If TRUE, return a CI container; limits are computed by a large-sample delta method for CCC (see **CIs** note below).
conf_level	Numeric in $(0, 1)$. Confidence level when ci = TRUE (default 0.95).
n_threads	Integer ≥ 1. Number of OpenMP threads to use for computation. Defaults to getOption("matrixCorr.threads", 1L).
ci_mode	Character scalar; one of c("auto","raw","logit"). Controls how confidence intervals are computed when ci = TRUE. If "raw", a Wald CI is formed on the CCC scale and truncated to [0,1]. If "logit", a Wald CI is computed on the $\mathrm{logit}(\mathrm{CCC})$ scale and back-transformed to the original scale (often more stable near 0 or 1). If "auto" (default), the method is chosen per estimate based on simple diagnostics (e.g., proximity to the [0,1] boundary / numerical stability), typically preferring "logit" near the boundaries and "raw" otherwise.
verbose	Logical. If TRUE, prints a structured summary of the fitted variance components and S_B for each fit. Default FALSE.
digits	Integer (≥ 0). Number of decimal places to use in the printed summary when verbose = TRUE. Default 4.
use_message	Logical. When verbose = TRUE, verbose summaries are emitted with **cli** messages.
interaction	Logical. Include method:time interaction? (default FALSE).
max_iter	Integer. Maximum iterations for variance-component updates (default 100).
tol	Numeric. Convergence tolerance on parameter change (default 1e-6).
Dmat	Optional $n_t \times n_t$ numeric matrix to weight/aggregate time-specific fixed biases in the S_B quadratic form. If supplied, it is used (after optional mass rescaling; see Dmat_rescale) whenever at least two *present* time levels exist; otherwise it is ignored. **If** Dmat **is** NULL, a canonical kernel D_m is *constructed* from Dmat_type and Dmat_weights (see below). Dmat should be symmetric positive semidefinite; small asymmetries are symmetrized internally.

Dmat_type — Character, one of `c("time-avg","typical-visit", "weighted-avg","weighted-sq")`. Only used when `Dmat = NULL`. It selects the aggregation target for time-specific fixed biases in S_B. Options are:

- `"time-avg"`: square of the time-averaged bias, $D_m = (1/n_t)\,1 1^\top$.
- `"typical-visit"`: average of squared per-time biases, $D_m = I_{n_t}$.
- `"weighted-avg"`: square of a weighted average, $D_m = n_t\, w\, w^\top$ with $\sum w = 1$.
- `"weighted-sq"`: weighted average of squared biases, $D_m = n_t\, \mathrm{diag}(w)$ with $\sum w = 1$.

Pick `"time-avg"` for CCC targeting the time-averaged measurement; pick `"typical-visit"` for CCC targeting a randomly sampled visit (typical occasion). Default `"time-avg"`.

Dmat_weights — Optional numeric weights w used when `Dmat_type %in% c("weighted-avg","weighted-sq")`. Must be nonnegative and finite. If `names(w)` are provided, they should match the *full* time levels in `data`; they are aligned to the *present* time subset per fit. If unnamed, the length must equal the number of present time levels. In all cases w is internally normalized to sum to 1.

Dmat_rescale — Logical. When `TRUE` (default), the supplied/built D_m is rescaled to satisfy the simple mass rule $1^\top D_m 1 = n_t$. This keeps the S_B denominator invariant and harmonizes with the κ-shrinkage used for variance terms.

ar — Character. Residual correlation structure: `"none"` (iid) or `"ar1"` for subject-level AR(1) correlation within contiguous time runs. Default `c("none")`.

ar_rho — Numeric in $(-0.999, 0.999)$ or `NA`. If `ar = "ar1"` and `ar_rho` is finite, it is treated as fixed. If `ar = "ar1"` and `ar_rho = NA`, ρ is estimated by profiling a 1-D objective (REML when available; an approximation otherwise). Default `NA_real_`.

slope — Character. Optional extra random-effect design Z. With `"subject"` a single random slope is added (one column in Z); with `"method"` one column per method level is added; with `"custom"` you provide `slope_Z` directly. Default `c("none","subject","method","custom")`.

slope_var — For `slope %in% c("subject","method")`, a character string giving the name of a column in `data` used as the slope regressor (e.g., centered time). It is looked up inside `data`; do not pass the vector itself. NAs are treated as zeros in Z.

slope_Z — For `slope = "custom"`, a numeric matrix with n rows (same order as `data`) providing the full extra random-effect design Z. **Each column of `slope_Z` has its own variance component** $\sigma^2_{Z,j}$; columns are treated as *uncorrelated* (diagonal block in G). Ignored otherwise.

drop_zero_cols — Logical. When `slope = "method"`, drop all-zero columns of Z after subsetting (useful in pairwise fits). Default `TRUE`.

vc_select — Character scalar; one of `c("auto","none")`. Controls how the subject by method $\sigma^2_{A\times M}$ ("subj_method") and subject by time $\sigma^2_{A\times T}$ ("subj_time") variance components are included. If `"auto"` (default), the function performs boundary-aware REML likelihood-ratio tests (LRTs; null on the boundary at zero with a half-χ^2_1 reference) to decide whether to retain each component, in the order given by `vc_test_order`. If `"none"`, no testing is done and inclusion is taken from `include_subj_method`/`include_subj_time` (or, if `NULL`, from the mere

presence of the corresponding factor in the design). In pairwise fits, the decision is made independently for each method pair.

`vc_alpha` Numeric scalar in $(0, 1)$; default `0.05`. Per-component significance level for the boundary-aware REML LRTs used when `vc_select = "auto"`. The tests are one-sided for variance components on the boundary and are *not* multiplicity-adjusted.

`vc_test_order` Character vector (length 2) with a permutation of `c("subj_time","subj_method")`; default `c("subj_time","subj_method")`. Specifies the order in which the two variance components are tested when `vc_select = "auto"`. The component tested first may be dropped before testing the second. If a factor is absent in the design (e.g., no time factor so "subj_time" is undefined), the corresponding test is skipped.

`include_subj_method`, `include_subj_time`
Logical scalars or `NULL`. When `vc_select = "none"`, these control whether the $\sigma^2_{A\times M}$ ("subj_method") and $\sigma^2_{A\times T}$ ("subj_time") random effects are included (`TRUE`) or excluded (`FALSE`) in the model. If `NULL` (default), inclusion defaults to the presence of the corresponding factor in the data (i.e., at least two method/time levels). When `vc_select = "auto"`, these arguments are ignored (automatic selection is used instead).

`sb_zero_tol` Non-negative numeric scalar; default `1e-10`. Numerical threshold for the fixed-effect dispersion term S_B. After computing $\widehat{S_B}$ and its delta-method variance, if $\widehat{S_B} \le$ `sb_zero_tol` or non-finite, the procedure treats S_B as fixed at zero in the delta step. It sets $d_{S_B} = 0$ and $\mathrm{Var}(\widehat{S_B}) = 0$, preventing numerical blow-ups of `SE(CCC)` when $\widehat{S_B} \to 0$ and the fixed-effects variance is ill-conditioned for the contrast. This stabilises inference in rare boundary cases; it has no effect when $\widehat{S_B}$ is comfortably above the threshold.

Details

For measurement y_{ij} on subject i under fixed levels (method, time), we fit

$$y = X\beta + Zu + \varepsilon, \qquad u \sim N(0,\, G),\ \varepsilon \sim N(0,\, R).$$

Notation: m subjects, $n = \sum_i n_i$ total rows; nm method levels; nt time levels; q_Z extra random-slope columns (if any); $r = 1 + nm + nt$ (or $1 + nm + nt + q_Z$ with slopes). Here Z is the subject-structured random-effects design and G is block-diagonal at the subject level with the following *per-subject* parameterisation. Specifically,

- one random intercept with variance σ^2_A;
- optionally, *method* deviations (one column per method level) with a common variance $\sigma^2_{A\times M}$ and zero covariances across levels (i.e., multiple of an identity);
- optionally, *time* deviations (one column per time level) with a common variance $\sigma^2_{A\times T}$ and zero covariances across levels;
- optionally, an *extra* random effect aligned with Z (random slope), where each *column* has its own variance $\sigma^2_{Z,j}$ and columns are uncorrelated.

The fixed-effects design is `~ 1 + method + time` and, if `interaction=TRUE`, `+ method:time`.

Residual correlation R (regular, equally spaced time). Write $R_i = \sigma_E^2\, C_i(\rho)$. With `ar="none"`, $C_i = I$. With `ar="ar1"`, within-subject residuals follow a *discrete* AR(1) process along the visit index after sorting by increasing time level. Ties retain input order, and any `NA` time code breaks the series so each contiguous block of non-`NA` times forms a run. The correlation between *adjacent observed visits* in a run is ρ; we do not use calendar-time gaps. Internally we work with the *precision* of the AR(1) correlation: for a run of length $L \geq 2$, the tridiagonal inverse has

$$(C^{-1})_{11} = (C^{-1})_{LL} = \frac{1}{1-\rho^2}, \quad (C^{-1})_{tt} = \frac{1+\rho^2}{1-\rho^2}\ (2 \leq t \leq L{-}1), \quad (C^{-1})_{t,t+1} = (C^{-1})_{t+1,t} = \frac{-\rho}{1-\rho^2}.$$

The working inverse is $R_i^{-1} = \sigma_E^{-2}\, C_i(\rho)^{-1}$.

Per-subject Woodbury system. For subject i with n_i rows, define the per-subject random-effects design U_i (columns: intercept, method indicators, time indicators; dimension $r = 1 + nm + nt$). The core never forms $V_i = R_i + U_i G U_i^\top$ explicitly. Instead,

$$M_i \;=\; G^{-1} \;+\; U_i^\top R_i^{-1} U_i,$$

and accumulates GLS blocks via rank-r corrections using $V_i^{-1} = R_i^{-1} - R_i^{-1} U_i M_i^{-1} U_i^\top R_i^{-1}$:

$$X^\top V^{-1} X \;=\; \sum_i \Big[X_i^\top R_i^{-1} X_i \;-\; (X_i^\top R_i^{-1} U_i)\, M_i^{-1}\, (U_i^\top R_i^{-1} X_i) \Big],$$

$$X^\top V^{-1} y \;=\; \sum_i \Big[X_i^\top R_i^{-1} y_i \;-\; (X_i^\top R_i^{-1} U_i)\, M_i^{-1}\, (U_i^\top R_i^{-1} y_i) \Big].$$

Because G^{-1} is diagonal with positive entries, each M_i is symmetric positive definite; solves/inversions use symmetric-PD routines with a small diagonal ridge and a pseudo-inverse if needed.

Random-slope Z. Besides U_i, the function can include an extra design Z_i.

- `slope="subject"`: Z has one column (the regressor in `slope_var`); Z_i is the subject-i block, with its own variance $\sigma^2_{Z,1}$.
- `slope="method"`: Z has one column per method level; row t uses the slope regressor if its method equals level ℓ, otherwise 0; all-zero columns can be dropped via `drop_zero_cols=TRUE` after subsetting. Each column has its own variance $\sigma^2_{Z,\ell}$.
- `slope="custom"`: Z is provided fully via `slope_Z`. Each column is an independent random effect with its own variance $\sigma^2_{Z,j}$; cross-covariances among columns are set to 0.

Computations simply augment $\tilde{U}_i = [U_i\ Z_i]$ and the corresponding inverse-variance block. The EM updates then include, for each column $j = 1, \ldots, q_Z$,

$$\sigma^{2\,(new)}_{Z,j} \;=\; \frac{1}{m} \sum_{i=1}^{m} \Big(b^2_{i,\text{extra},j} + (M_i^{-1})_{\text{extra},jj} \Big) \quad (\text{if } q_Z > 0).$$

Interpretation: the $\sigma^2_{Z,j}$ represent additional within-subject variability explained by the slope regressor(s) in column j and are *not* part of the CCC denominator (agreement across methods/time).

EM-style variance-component updates. With current $\hat{\beta}$, form residuals $r_i = y_i - X_i\hat{\beta}$. The BLUPs and conditional covariances are

$$b_i \;=\; M_i^{-1}\, (U_i^\top R_i^{-1} r_i), \qquad \mathrm{Var}(b_i \mid y) \;=\; M_i^{-1}.$$

Let $e_i = r_i - U_i b_i$. Expected squares then yield closed-form updates:

$$\sigma_A^{2\,(new)} \;=\; \frac{1}{m}\sum_i \Big(b_{i,0}^2 + (M_i^{-1})_{00}\Big),$$

$$\sigma_{A\times M}^{2\,(new)} \;=\; \frac{1}{m\,nm}\sum_i \sum_{\ell=1}^{nm}\Big(b_{i,\ell}^2 + (M_i^{-1})_{\ell\ell}\Big) \quad (\text{if } nm>0),$$

$$\sigma_{A\times T}^{2\,(new)} \;=\; \frac{1}{m\,nt}\sum_i \sum_{t=1}^{nt}\Big(b_{i,t}^2 + (M_i^{-1})_{tt}\Big) \quad (\text{if } nt>0),$$

$$\sigma_E^{2\,(new)} \;=\; \frac{1}{n}\sum_i \Big(e_i^\top C_i(\rho)^{-1} e_i \;+\; \mathrm{tr}\big(M_i^{-1} U_i^\top C_i(\rho)^{-1} U_i\big)\Big),$$

together with the per-column update for $\sigma_{Z,j}^2$ given above. Iterate until the ℓ_1 change across components is $<$`tol` or `max_iter` is reached.

Fixed-effect dispersion S_B: choosing the time-kernel D_m.

Let $d = L^\top \hat\beta$ stack the within-time, pairwise method differences, grouped by time as $d = (d_1^\top, \dots, d_{n_t}^\top)^\top$ with $d_t \in \mathbb{R}^P$ and $P = n_m(n_m-1)$. The symmetric positive semidefinite kernel $D_m \succeq 0$ selects which functional of the bias profile $t \mapsto d_t$ is targeted by S_B. Internally, the code rescales any supplied/built D_m to satisfy $1^\top D_m 1 = n_t$ for stability and comparability.

- `Dmat_type = "time-avg"` (square of the time-averaged bias). Let

  $$w \;=\; \frac{1}{n_t}\,\mathbf{1}_{n_t}, \qquad D_m \;\propto\; I_P \otimes (w\,w^\top),$$

  so that

  $$d^\top D_m d \;\propto\; \sum_{p=1}^{P}\left(\frac{1}{n_t}\sum_{t=1}^{n_t} d_{t,p}\right)^2.$$

  Methods have equal *time-averaged* means within subject, i.e. $\sum_{t=1}^{n_t} d_{t,p}/n_t = 0$ for all p. Appropriate when decisions depend on an average over time and opposite-signed biases are allowed to cancel.

- `Dmat_type = "typical-visit"` (average of squared per-time biases). With equal visit probability, take

  $$D_m \;\propto\; I_P \otimes \mathrm{diag}\Big(\tfrac{1}{n_t},\dots,\tfrac{1}{n_t}\Big),$$

  yielding

  $$d^\top D_m d \;\propto\; \frac{1}{n_t}\sum_{t=1}^{n_t}\sum_{p=1}^{P} d_{t,p}^2.$$

  Methods agree on a *typical* occasion drawn uniformly from the visit set. Use when each visit matters on its own; alternating signs $d_{t,p}$ do not cancel.

- `Dmat_type = "weighted-avg"` (square of a weighted time average). For user weights $a = (a_1,\dots,a_{n_t})^\top$ with $a_t \ge 0$, set

  $$w \;=\; \frac{a}{\sum_{t=1}^{n_t} a_t}, \qquad D_m \;\propto\; I_P \otimes (w\,w^\top),$$

so that

$$d^\top D_m d \;\propto\; \sum_{p=1}^{P} \left(\sum_{t=1}^{n_t} w_t \, d_{t,p} \right)^2 .$$

Methods have equal *weighted time-averaged* means, i.e. $\sum_{t=1}^{n_t} w_t \, d_{t,p} = 0$ for all p. Use when some visits (e.g., baseline/harvest) are a priori more influential; opposite-signed biases may cancel according to w.

- `Dmat_type = "weighted-sq"` (weighted average of squared per-time biases). With the same weights w, take

$$D_m \;\propto\; I_P \otimes \mathrm{diag}(w_1, \ldots, w_{n_t}),$$

giving

$$d^\top D_m d \;\propto\; \sum_{t=1}^{n_t} w_t \sum_{p=1}^{P} d_{t,p}^2 .$$

Methods agree at visits sampled with probabilities $\{w_t\}$, counting each visit's discrepancy on its own. Use when per-visit agreement is required but some visits should be emphasised more than others.

Time-averaging for CCC (regular visits). The reported CCC targets agreement of the *time-averaged* measurements per method within subject by default (`Dmat_type="time-avg"`). Averaging over T non-`NA` visits shrinks time-varying components by

$$\kappa_T^{(g)} \;=\; 1/T, \qquad \kappa_T^{(e)} \;=\; \{T + 2\sum_{k=1}^{T-1}(T-k)\rho^k\}/T^2,$$

with $\kappa_T^{(e)} = 1/T$ when residuals are i.i.d. With unbalanced T, the implementation averages the per-(subject,method) κ values across the pairs contributing to CCC and then clamps $\kappa_T^{(e)}$ to $[10^{-12}, 1]$ for numerical stability. Choosing `Dmat_type="typical-visit"` makes S_B match the interpretation of a randomly sampled occasion instead.

Concordance correlation coefficient. The CCC used is

$$\mathrm{CCC} \;=\; \frac{\sigma_A^2 + \kappa_T^{(g)}\,\sigma_{A\times T}^2}{\sigma_A^2 + \sigma_{A\times M}^2 + \kappa_T^{(g)}\,\sigma_{A\times T}^2 + S_B + \kappa_T^{(e)}\,\sigma_E^2}.$$

Special cases: with no method factor, $S_B = \sigma_{A\times M}^2 = 0$; with a single time level, $\sigma_{A\times T}^2 = 0$ (no κ-shrinkage). When $T = 1$ or $\rho = 0$, both κ-factors equal 1. The *extra* random-effect variances $\{\sigma_{Z,j}^2\}$ (if used) are *not* included.

CIs / SEs (delta method for CCC). Let

$$\theta \;=\; \left(\sigma_A^2,\, \sigma_{A\times M}^2,\, \sigma_{A\times T}^2,\, \sigma_E^2,\, S_B\right)^\top,$$

and write $\mathrm{CCC}(\theta) = N/D$ with $N = \sigma_A^2 + \kappa_T^{(g)}\sigma_{A\times T}^2$ and $D = \sigma_A^2 + \sigma_{A\times M}^2 + \kappa_T^{(g)}\sigma_{A\times T}^2 + S_B + \kappa_T^{(e)}\sigma_E^2$. The gradient components are

$$\frac{\partial\,\mathrm{CCC}}{\partial\sigma_A^2} \;=\; \frac{\sigma_{A\times M}^2 + S_B + \kappa_T^{(e)}\sigma_E^2}{D^2},$$

$$\frac{\partial\,\mathrm{CCC}}{\partial\sigma^2_{A\times M}} \;=\; -\,\frac{N}{D^2}, \qquad \frac{\partial\,\mathrm{CCC}}{\partial\sigma^2_{A\times T}} \;=\; \frac{\kappa_T^{(g)}\big(\sigma^2_{A\times M}+S_B+\kappa_T^{(e)}\sigma^2_E\big)}{D^2},$$

$$\frac{\partial\,\mathrm{CCC}}{\partial\sigma^2_{E}} \;=\; -\,\frac{\kappa_T^{(e)}\,N}{D^2}, \qquad \frac{\partial\,\mathrm{CCC}}{\partial S_B} \;=\; -\,\frac{N}{D^2}.$$

Estimating $\mathrm{Var}(\hat\theta)$. The EM updates write each variance component as an average of per-subject quantities. For subject i,

$$t_{A,i} \;=\; b_{i,0}^2 + (M_i^{-1})_{00}, \qquad t_{M,i} \;=\; \frac{1}{nm}\sum_{\ell=1}^{nm}\Big(b_{i,\ell}^2 + (M_i^{-1})_{\ell\ell}\Big),$$

$$t_{T,i} \;=\; \frac{1}{nt}\sum_{j=1}^{nt}\Big(b_{i,j}^2 + (M_i^{-1})_{jj}\Big), \qquad s_i \;=\; \frac{e_i^\top C_i(\rho)^{-1}e_i + \mathrm{tr}\big(M_i^{-1}U_i^\top C_i(\rho)^{-1}U_i\big)}{n_i},$$

where $b_i = M_i^{-1}(U_i^\top R_i^{-1} r_i)$ and $M_i = G^{-1} + U_i^\top R_i^{-1} U_i$. With m subjects, form the empirical covariance of the stacked subject vectors and scale by m to approximate the covariance of the means:

$$\widehat{\mathrm{Cov}}\left(\begin{bmatrix} t_{A,\cdot} \\ t_{M,\cdot} \\ t_{T,\cdot}\end{bmatrix}\right) \;\approx\; \frac{1}{m}\,\mathrm{Cov}_i\left(\begin{bmatrix} t_{A,i} \\ t_{M,i} \\ t_{T,i}\end{bmatrix}\right).$$

(Drop rows/columns as needed when `nm==0` or `nt==0`.)

The residual variance estimator is a weighted mean $\hat\sigma^2_E = \sum_i w_i s_i$ with $w_i = n_i/n$. Its variance is approximated by the variance of a weighted mean of independent terms,

$$\widehat{\mathrm{Var}}(\hat\sigma^2_E) \;\approx\; \Big(\sum_i w_i^2\Big)\,\widehat{\mathrm{Var}}(s_i),$$

where $\widehat{\mathrm{Var}}(s_i)$ is the sample variance across subjects. The method-dispersion term uses the quadratic-form delta already computed for S_B:

$$\widehat{\mathrm{Var}}(S_B) \;=\; \frac{2\,\mathrm{tr}\big((A_{fix}\,\mathrm{Var}(\hat\beta))^2\big) \;+\; 4\,\hat\beta^\top A_{fix}\,\mathrm{Var}(\hat\beta)\,A_{fix}\,\hat\beta}{\big[nm\,(nm-1)\,\max(nt,1)\big]^2},$$

with $A_{fix} = L\,D_m\,L^\top$.

Putting it together. Assemble $\widehat{\mathrm{Var}}(\hat\theta)$ by combining the $(\sigma^2_A, \sigma^2_{A\times M}, \sigma^2_{A\times T})$ covariance block from the subject-level empirical covariance, add the $\widehat{\mathrm{Var}}(\hat\sigma^2_E)$ and $\widehat{\mathrm{Var}}(S_B)$ terms on the diagonal, and ignore cross-covariances across these blocks (a standard large-sample simplification). Then

$$\widehat{\mathrm{se}}\{\widehat{\mathrm{CCC}}\} \;=\; \sqrt{\nabla\mathrm{CCC}(\hat\theta)^\top\,\widehat{\mathrm{Var}}(\hat\theta)\,\nabla\mathrm{CCC}(\hat\theta)}\,.$$

A two-sided $(1-\alpha)$ normal CI is

$$\widehat{\mathrm{CCC}} \;\pm\; z_{1-\alpha/2}\,\widehat{\mathrm{se}}\{\widehat{\mathrm{CCC}}\},$$

truncated to $[0,1]$ in the output for convenience. When S_B is truncated at 0 or samples are very small/imbalanced, the normal CI can be mildly anti-conservative near the boundary; a logit transform for CCC or a subject-level (cluster) bootstrap can be used for sensitivity analysis.

Choosing ρ for AR(1). When `ar="ar1"` and `ar_rho = NA`, ρ is estimated by profiling the REML log-likelihood at $(\hat\beta, \hat G, \hat\sigma^2_E)$. With very few visits per subject, ρ can be weakly identified; consider sensitivity checks over a plausible range.

Notes on stability and performance

All per-subject solves are $r \times r$ with $r = 1+nm+nt+q_Z$, so cost scales with the number of subjects and the fixed-effects dimension rather than the total number of observations. Solvers use symmetric-PD paths with a small diagonal ridge and pseudo-inverse, which helps for very small/unbalanced subsets and near-boundary estimates. For `AR(1)`, observations are ordered by time within subject; `NA` time codes break the run, and gaps between factor levels are treated as regular steps (elapsed time is not used).

Heteroscedastic slopes across Z columns are supported. Each Z column has its own variance component $\sigma^2_{Z,j}$, but cross-covariances among Z columns are set to zero (diagonal block). Column rescaling changes the implied prior on $b_{i,\text{extra}}$ but does not introduce correlations.

Threading and BLAS guards

The C++ backend uses OpenMP loops while also forcing vendor BLAS libraries to run single-threaded so that overall CPU usage stays predictable. This guard is applied to OpenBLAS, Apple's Accelerate, and Intel MKL when their runtime controls are available. You can opt out manually by setting `MATRIXCORR_DISABLE_BLAS_GUARD=1` in the environment before loading **matrixCorr**.

Model overview

Internally, the call is routed to `ccc_lmm_reml_pairwise()`, which fits one repeated-measures mixed model per pair of methods. Each model includes:

- subject random intercepts (always)
- optional subject-by-method (`sigma^2_{A \times M}`) and subject-by-time (`sigma^2_{A \times T}`) variance components
- optional random slopes specified via `slope`/`slope_var`/`slope_Z`
- residual structure `ar = "none"` (iid) or `ar = "ar1"`

D-matrix options (`Dmat_type`, `Dmat`, `Dmat_weights`) control how time averaging operates when translating variance components into CCC summaries.

Author(s)

Thiago de Paula Oliveira

References

Lin L (1989). A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45: 255-268.

Lin L (2000). A note on the concordance correlation coefficient. *Biometrics*, 56: 324-325.

Carrasco, J. L. et al. (2013). Estimation of the concordance correlation coefficient for repeated measures using SAS and R. *Computer Methods and Programs in Biomedicine*, 109(3), 293-304. doi:10.1016/j.cmpb.2012.09.002

King et al. (2007). A Class of Repeated Measures Concordance Correlation Coefficients. *Journal of Biopharmaceutical Statistics*, 17(4). doi:10.1080/10543400701329455

See Also

build_L_Dm_Z_cpp for constructing $L/D_m/Z$; ccc_rm_ustat for a U-statistic alternative; and **cccrm** for a reference approach via **nlme**.

Examples

```
# ======================================================================
# 1) Subject x METHOD variance present, no time
#     y_{i,m} = mu + b_m + u_i + w_{i,m} + e_{i,m}
#     with u_i ~ N(0, s_A^2), w_{i,m} ~ N(0, s_{AxM}^2)
# ======================================================================
set.seed(102)
n_subj <- 60
n_time <- 8

id     <- factor(rep(seq_len(n_subj), each = 2 * n_time))
time   <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))
method <- factor(rep(rep(c("A","B"),    each  = n_time), times = n_subj))

sigA  <- 0.6   # subject
sigAM <- 0.3   # subject x method
sigAT <- 0.5   # subject x time
sigE  <- 0.4   # residual
# Expected estimate S_B = 0.2^2 = 0.04
biasB <- 0.2   # fixed method bias

# random effects
u_i <- rnorm(n_subj, 0, sqrt(sigA))
u   <- u_i[as.integer(id)]

sm      <- interaction(id, method, drop = TRUE)
w_im_lv <- rnorm(nlevels(sm), 0, sqrt(sigAM))
w_im    <- w_im_lv[as.integer(sm)]

st      <- interaction(id, time, drop = TRUE)
g_it_lv <- rnorm(nlevels(st), 0, sqrt(sigAT))
g_it    <- g_it_lv[as.integer(st)]

# residuals & response
e <- rnorm(length(id), 0, sqrt(sigE))
y <- (method == "B") * biasB + u + w_im + g_it + e

dat_both <- data.frame(y, id, method, time)

# Both sigma2_subject_method and sigma2_subject_time are identifiable here
fit_both <- ccc_rm_reml(dat_both, "y", "id", method = "method", time = "time",
                        vc_select = "auto", ci = TRUE, verbose = TRUE)
summary(fit_both)
estimate(fit_both)
tidy(fit_both)
ci(fit_both)
confint(fit_both)
```

```
plot(fit_both)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(fit_both)
}

# ======================================================================
# 2) Subject x TIME variance present (sag > 0) with two methods
#     y_{i,m,t} = mu + b_m + u_i + g_{i,t} + e_{i,m,t}
#     where g_{i,t} ~ N(0, s_{AxT}^2) shared across methods at time t
# ======================================================================
set.seed(202)
n_subj <- 60; n_time <- 14
id     <- factor(rep(seq_len(n_subj), each = 2 * n_time))
method <- factor(rep(rep(c("A","B"), each = n_time), times = n_subj))
time   <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))

sigA  <- 0.7
sigAT <- 0.5
sigE  <- 0.5
biasB <- 0.25

u   <- rnorm(n_subj, 0, sqrt(sigA))[as.integer(id)]
gIT <- rnorm(n_subj * n_time, 0, sqrt(sigAT))
g   <- gIT[ (as.integer(id) - 1L) * n_time + as.integer(time) ]
y   <- (method == "B") * biasB + u + g + rnorm(length(id), 0, sqrt(sigE))
dat_sag <- data.frame(y, id, method, time)

# sigma_AT should be retained; sigma_AM may be dropped (since w_{i,m}=0)
fit_sag <- ccc_rm_reml(dat_sag, "y", "id", method = "method", time = "time",
                       vc_select = "auto", verbose = TRUE)
summary(fit_sag)
plot(fit_sag)

# ======================================================================
# 3) BOTH components present: sab > 0 and sag > 0 (2 methods x T times)
#     y_{i,m,t} = mu + b_m + u_i + w_{i,m} + g_{i,t} + e_{i,m,t}
# ======================================================================
set.seed(303)
n_subj <- 60; n_time <- 4
id     <- factor(rep(seq_len(n_subj), each = 2 * n_time))
method <- factor(rep(rep(c("A","B"), each = n_time), times = n_subj))
time   <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))

sigA  <- 0.8
sigAM <- 0.3
sigAT <- 0.4
sigE  <- 0.5
biasB <- 0.2

u   <- rnorm(n_subj, 0, sqrt(sigA))[as.integer(id)]
# (subject, method) random deviations: repeat per (i,m) across its times
```

```
wIM <- rnorm(n_subj * 2, 0, sqrt(sigAM))
w   <- wIM[ (as.integer(id) - 1L) * 2 + as.integer(method) ]
# (subject, time) random deviations: shared across methods at time t
gIT <- rnorm(n_subj * n_time, 0, sqrt(sigAT))
g   <- gIT[ (as.integer(id) - 1L) * n_time + as.integer(time) ]
y   <- (method == "B") * biasB + u + w + g + rnorm(length(id), 0, sqrt(sigE))
dat_both <- data.frame(y, id, method, time)

fit_both <- ccc_rm_reml(dat_both, "y", "id", method = "method", time = "time",
                        vc_select = "auto", verbose = TRUE, ci = TRUE)
summary(fit_both)
plot(fit_both)

# If you want to force-include both VCs (skip testing):
fit_both_forced <-
 ccc_rm_reml(dat_both, "y", "id", method = "method", time = "time",
              vc_select = "none", include_subj_method  = TRUE,
              include_subj_time  = TRUE, verbose = TRUE)
summary(fit_both_forced)
plot(fit_both_forced)

# ======================================================================
# 4) D_m choices: time-averaged (default) vs typical visit
# ======================================================================
# Time-average
ccc_rm_reml(dat_both, "y", "id", method = "method", time = "time",
             vc_select = "none", include_subj_method  = TRUE,
             include_subj_time  = TRUE, Dmat_type = "time-avg")

# Typical visit
ccc_rm_reml(dat_both, "y", "id", method = "method", time = "time",
             vc_select = "none", include_subj_method  = TRUE,
             include_subj_time  = TRUE, Dmat_type = "typical-visit")

# ======================================================================
# 5) AR(1) residual correlation with fixed rho (larger example)
# ======================================================================

set.seed(10)
n_subj   <- 40
n_time   <- 10
methods  <- c("A", "B", "C", "D")
nm       <- length(methods)
id     <- factor(rep(seq_len(n_subj), each = n_time * nm))
method <- factor(rep(rep(methods, each = n_time), times = n_subj),
                 levels = methods)
time   <- factor(rep(rep(seq_len(n_time), times = nm), times = n_subj))

beta0    <- 0
beta_t   <- 0.2
bias_met <- c(A = 0.00, B = 0.30, C = -0.15, D = 0.05)
sigA     <- 1.0
rho_true <- 0.6
```

```
sigE     <- 0.7

t_num <- as.integer(time)
t_c   <- t_num - mean(seq_len(n_time))
mu    <- beta0 + beta_t * t_c + bias_met[as.character(method)]

u_subj <- rnorm(n_subj, 0, sqrt(sigA))
u      <- u_subj[as.integer(id)]

e <- numeric(length(id))
for (s in seq_len(n_subj)) {
  for (m in methods) {
    idx <- which(id == levels(id)[s] & method == m)
    e[idx] <- stats::arima.sim(list(ar = rho_true), n = n_time, sd = sigE)
  }
}
y <- mu + u + e
dat_ar4 <- data.frame(y = y, id = id, method = method, time = time)

fit4 <- ccc_rm_reml(dat_ar4,
             response = "y", subject = "id", method = "method", time = "time",
             ar = "ar1", ar_rho = 0.6, verbose = TRUE)
fit4
summary(fit4)
plot(fit4)


# ======================================================================
# 6) Random slope variants (subject, method, custom Z)
# ======================================================================

## By SUBJECT
set.seed(2)
n_subj <- 60; n_time <- 4
id  <- factor(rep(seq_len(n_subj), each = 2 * n_time))
tim <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))
method <- factor(rep(rep(c("A","B"), each = n_time), times = n_subj))
subj <- as.integer(id)
slope_i <- rnorm(n_subj, 0, 0.15)
slope_vec <- slope_i[subj]
base <- rnorm(n_subj, 0, 1.0)[subj]
tnum <- as.integer(tim)
y <- base + 0.3*(method=="B") + slope_vec*(tnum - mean(seq_len(n_time))) +
     rnorm(length(id), 0, 0.5)
dat_s <- data.frame(y, id, method, time = tim)
dat_s$t_num <- as.integer(dat_s$time)
dat_s$t_c   <- ave(dat_s$t_num, dat_s$id, FUN = function(v) v - mean(v))
ccc_rm_reml(dat_s, "y", "id", method = "method", time = "time",
             slope = "subject", slope_var = "t_c", verbose = TRUE)

## By METHOD
set.seed(3)
n_subj <- 60; n_time <- 4
```

```
id  <- factor(rep(seq_len(n_subj), each = 2 * n_time))
tim <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))
method <- factor(rep(rep(c("A","B"), each = n_time), times = n_subj))
slope_m <- ifelse(method=="B", 0.25, 0.00)
base <- rnorm(n_subj, 0, 1.0)[as.integer(id)]
tnum <- as.integer(tim)
y <- base + 0.3*(method=="B") + slope_m*(tnum - mean(seq_len(n_time))) +
     rnorm(length(id), 0, 0.5)
dat_m <- data.frame(y, id, method, time = tim)
dat_m$t_num <- as.integer(dat_m$time)
dat_m$t_c   <- ave(dat_m$t_num, dat_m$id, FUN = function(v) v - mean(v))
ccc_rm_reml(dat_m, "y", "id", method = "method", time = "time",
             slope = "method", slope_var = "t_c", verbose = TRUE)

## SUBJECT + METHOD random slopes (custom Z)
set.seed(4)
n_subj <- 50; n_time <- 4
id  <- factor(rep(seq_len(n_subj), each = 2 * n_time))
tim <- factor(rep(rep(seq_len(n_time), times = 2), times = n_subj))
method <- factor(rep(rep(c("A","B"), each = n_time), times = n_subj))
subj <- as.integer(id)
slope_subj <- rnorm(n_subj, 0, 0.12)[subj]
slope_B    <- ifelse(method=="B", 0.18, 0.00)
tnum <- as.integer(tim)
base <- rnorm(n_subj, 0, 1.0)[subj]
y <- base + 0.3*(method=="B") +
     (slope_subj + slope_B) * (tnum - mean(seq_len(n_time))) +
     rnorm(length(id), 0, 0.5)
dat_bothRS <- data.frame(y, id, method, time = tim)
dat_bothRS$t_num <- as.integer(dat_bothRS$time)
dat_bothRS$t_c   <- ave(dat_bothRS$t_num, dat_bothRS$id, FUN = function(v) v - mean(v))
MM <- model.matrix(~ 0 + method, data = dat_bothRS)
Z_custom <- cbind(
  subj_slope = dat_bothRS$t_c,
  MM * dat_bothRS$t_c
)
ccc_rm_reml(dat_bothRS, "y", "id", method = "method", time = "time",
             slope = "custom", slope_Z = Z_custom, verbose = TRUE)
```

`ccc_rm_ustat` *Repeated-Measures Lin's Concordance Correlation Coefficient (CCC)*

Description

Computes all pairwise Lin's Concordance Correlation Coefficients (CCC) across multiple methods ($L \geq 2$) for repeated-measures data. Each subject must be measured by all methods across the same set of time points or replicates.

CCC measures both accuracy (how close measurements are to the line of equality) and precision (Pearson correlation). Confidence intervals are optionally computed using a U-statistics-based estimator with Fisher's Z transformation

Usage

```
ccc_rm_ustat(
  data,
  response,
  subject,
  method,
  time = NULL,
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE,
  Dmat = NULL,
  delta = 1
)
```

Arguments

`data`	A data frame containing the repeated-measures dataset.
`response`	Character. Name of the numeric outcome column.
`subject`	Character. Column identifying subjects. Every subject must have measurements from all methods (and times, when supplied); rows with incomplete {subject, time, method} coverage are dropped per pair.
`method`	Character. Name of the method column (factor with L $\geq$ 2 levels).
`time`	Character or NULL. Name of the time/repetition column. If NULL, one time point is assumed.
`ci`	Logical. If TRUE, returns confidence intervals (default FALSE).
`conf_level`	Confidence level for CI (default 0.95).
`n_threads`	Integer ($\geq$ 1). Number of OpenMP threads to use for computation. Defaults to `getOption("matrixCorr.threads", 1L)`.
`verbose`	Logical. If TRUE, prints diagnostic output (default FALSE).
`Dmat`	Optional numeric weight matrix (T $\times$ T) for timepoints. Defaults to identity.
`delta`	Numeric. Exponent applied to the absolute pointwise differences between two method trajectories before the time-weighted quadratic form is evaluated. Internally, the function forms $v_t = \|X_t - Y_t\|^\delta$ and then computes $v^\top D v$, where D is `Dmat`. Therefore, when `Dmat` is diagonal: • `delta = 1` (default) gives the standard quadratic repeated-measures CCC distance, $(X-Y)^\top D(X-Y)$. • `delta = 2` gives a fourth-power loss, which puts stronger emphasis on large disagreements. • `delta = 0` gives a binary disagreement indicator before aggregation, analogous to a repeated-measures kappa-type distance.

In most applications, `delta = 1` should be used because it matches the usual quadratic distance used in repeated-measures CCC.

Details

This function computes pairwise Lin's Concordance Correlation Coefficient (CCC) between methods in a repeated-measures design using a U-statistics-based nonparametric estimator proposed by Carrasco et al. (2013). It is computationally efficient and robust, particularly for large-scale or balanced longitudinal designs.

Lin's CCC is defined as

$$\rho_c = \frac{2 \cdot \text{cov}(X, Y)}{\sigma_X^2 + \sigma_Y^2 + (\mu_X - \mu_Y)^2}$$

where:

- X and Y are paired measurements from two methods.
- μ_X, μ_Y are means, and σ_X^2, σ_Y^2 are variances.

U-statistics Estimation:

For repeated measures across T time points and n subjects we assume

- all $n(n-1)$ pairs of subjects are considered to compute a U-statistic estimator for within-method and cross-method distances.
- if `delta > 0`, pairwise distances are raised to a power before applying a time-weighted kernel matrix D.
- if `delta = 0`, the method reduces to a version similar to a repeated-measures kappa.

Confidence Intervals:

Confidence intervals are constructed using a **Fisher Z-transformation** of the CCC. Specifically,

- The CCC is transformed using $Z = 0.5 \log((1 + \rho_c)/(1 - \rho_c))$.
- Standard errors are computed from the asymptotic variance of the U-statistic.
- Normal-based intervals are computed on the Z-scale and then back-transformed to the CCC scale.

Assumptions:

- The design must be **balanced**, where all subjects must have complete observations for all methods and time points.
- The method is **nonparametric** and does not require assumptions of normality or linear mixed effects.
- Weights (`Dmat`) allow differential importance of time points.

For **unbalanced** or **complex hierarchical** data (e.g., missing timepoints, covariate adjustments), consider using `ccc_rm_reml`, which uses a variance components approach via linear mixed models.

Value

If `ci = FALSE`, a symmetric matrix of class `"ccc"` (estimates only). If `ci = TRUE`, a list of class `"ccc"`, `"ccc_ci"` with elements:

- `est`: CCC estimate matrix
- `lwr.ci`: Lower bound matrix
- `upr.ci`: Upper bound matrix

Author(s)

Thiago de Paula Oliveira

References

Lin L (1989). A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45: 255-268.

Lin L (2000). A note on the concordance correlation coefficient. *Biometrics*, 56: 324-325.

Carrasco JL, Jover L (2003). Estimating the concordance correlation coefficient: a new approach. *Computational Statistics & Data Analysis*, 47(4): 519-539.

See Also

`ccc`, `ccc_rm_reml`, `plot.ccc`, `print.ccc`

Examples

```
set.seed(123)
df <- expand.grid(subject = 1:10,
                  time = 1:2,
                  method = c("A", "B", "C"))
df$y <- rnorm(nrow(df), mean = match(df$method, c("A", "B", "C")), sd = 1)

# CCC matrix (no CIs)
ccc1 <- ccc_rm_ustat(df, response = "y", subject = "subject",
                           method = "method", time = "time")
print(ccc1)
summary(ccc1)
plot(ccc1)

# With confidence intervals
ccc2 <- ccc_rm_ustat(df, response = "y", subject = "subject",
                           method = "method", time = "time", ci = TRUE)
print(ccc2)
summary(ccc2)
estimate(ccc2)
tidy(ccc2)
ci(ccc2)
confint(ccc2)
plot(ccc2)
```

```
# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(ccc2)
}

#-------------------------------------------------------------------------
# Choosing delta based on distance sensitivity
#-------------------------------------------------------------------------
# Standard quadratic RM-CCC distance: (X - Y)' D (X - Y)
ccc_rm_ustat(df, response = "y", subject = "subject",
             method = "method", time = "time", delta = 1)

# Fourth-power loss when D is diagonal: emphasises large disagreements
ccc_rm_ustat(df, response = "y", subject = "subject",
             method = "method", time = "time", delta = 2)
# Binary disagreement indicator before aggregation
ccc_rm_ustat(df, response = "y", subject = "subject",
             method = "method", time = "time", delta = 0)
```

`cia` *Coefficient of Individual Agreement*

Description

`cia()` estimates the coefficient of individual agreement. CIA assesses individual-level interchangeability by comparing between-method disagreement with within-method replicate disagreement. Unlike CCC, CIA is not intended to be driven by between-subject heterogeneity.

The estimator requires replicated readings within method. A data set with one observation per subject per method is insufficient for CIA because the within-method disagreement term cannot be estimated.

Usage

```
cia(
  data,
  response,
  subject,
  method,
  replicate,
  reference = NULL,
  scope = c("pairwise", "overall"),
  estimator = c("mom_unconstrained", "vc_constrained"),
  ci = FALSE,
  conf_level = 0.95,
  inference = c("delta", "bootstrap", "none"),
  B = 1000L,
  seed = NULL,
  n_threads = getOption("matrixCorr.threads", 1L),
```

```
  verbose = FALSE
)

## S3 method for class 'cia'
summary(object, digits = 4, ci_digits = 3, ...)

## S3 method for class 'cia'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.cia'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'cia'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

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

Arguments

data	A data frame in long format.

response	Character scalar naming the numeric measurement column.
subject	Character scalar naming the subject/unit identifier column.
method	Character scalar naming the method/device/rater column.
replicate	Character scalar naming the replicate identifier within each subject-method cell.
reference	Optional character scalar naming the reference method. When NULL (default), the no-reference CIA is estimated.
scope	One of "pairwise" or "overall". "pairwise" returns the two-method CIA for each method pair, or the reference-scaled CIA for each method-versus-reference comparison when reference is supplied. "overall" returns one CIA coefficient across all retained methods.
estimator	One of "mom_unconstrained" or "vc_constrained". "mom_unconstrained" is the literal raw method-of-moments CIA ratio and may exceed 1. "vc_constrained" is a bounded package-level variance-component variant that constrains the estimated inter-method variance component to be non-negative before converting it back to CIA.
ci	Logical; if TRUE, attach confidence intervals using the selected inference method.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
inference	One of "delta", "bootstrap", or "none". When confidence intervals are requested, pairwise CIA uses a delta-method normal interval by default and also supports subject-bootstrap percentile intervals. For overall CIA, delta-method normal intervals are available for the unconstrained moment estimator, and subject-bootstrap percentile intervals are also available.
B	Number of subject-bootstrap resamples when ci = TRUE and inference = "bootstrap".
seed	Optional positive integer seed for reproducible bootstrap resampling.
n_threads	Integer >= 1. Number of OpenMP threads passed to the C++ backend.
verbose	Logical; if TRUE, emit short progress messages.
object	A cia object.
digits	Integer; number of decimal places for estimates.
ci_digits	Integer; number of decimal places for confidence limits.
...	Additional arguments passed to downstream print helpers.
x	A cia object.
n	Optional preview row threshold.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.
width	Optional display width.
show_ci	One of "yes" or "no".
title	Optional plot title.
low_color	Color used for lower agreement values.
high_color	Color used for higher agreement values.
mid_color	Midpoint color for pairwise heatmaps.

`value_text_size`
: Text size for overlaid estimate labels.

`ci_text_size`
: Text size for CI labels.

`show_value`
: Logical; whether to overlay numeric values.

Details

Let `Y_ijk` denote replicate `k` on subject `i` measured by method `j`.

Without a reference method, CIA is defined by

$$\mathrm{CIA}^N = \frac{\sum_j E\{(Y_{ijk} - Y_{ijk'})^2\}/2}{\sum_{j<j'} E\{(Y_{ij} - Y_{ij'})^2\}/(J-1)}.$$

With a reference method `J`, CIA is defined by

$$\mathrm{CIA}^R = \frac{E\{(Y_{iJk} - Y_{iJk'})^2\}}{\sum_{j=1}^{J-1} E\{(Y_{ij} - Y_{iJ})^2\}/(J-1)}.$$

This implementation uses method-of-moments estimators built from subject-level disagreement functions.

For pairwise CIA, consider two methods `X` and `Y`. For each eligible subject `i`, define

$$G_{1i} = \mathrm{mean}\{(X_{ik} - X_{ik'})^2\},$$

$$G_{2i} = \mathrm{mean}\{(Y_{il} - Y_{il'})^2\},$$

$$G_{3i} = \mathrm{mean}\{(X_{ik} - Y_{il})^2\},$$

where the first two means are over all distinct within-method replicate pairs and the third mean is over all cross-method replicate combinations. Let $\bar{G}_1$, $\bar{G}_2$, and $\bar{G}_3$ denote the averages of these subject-level quantities across eligible subjects. Then the no-reference pairwise estimator is

$$\widehat{\mathrm{CIA}}^{\mathrm{raw}}_{XY} = \frac{(\bar{G}_1 + \bar{G}_2)/2}{\bar{G}_3},$$

and the reference pairwise estimator is

$$\widehat{\mathrm{CIA}}^{\mathrm{raw}}_{XR} = \frac{\bar{G}_1}{\bar{G}_3},$$

where `X` is the reference method.

For overall CIA, the current implementation follows the balanced replicated formulas in the cited papers. This requires `n` common subjects, `J` retained methods, and the same replicate count `K >= 2` in every retained subject-method cell. If the data are not balanced in this sense, `scope = "overall"` returns an informative error rather than using an approximation.

Write $\bar{Y}_{ij}$ for the mean of the `K` replicates on subject `i` and method `j`. Define the within-method mean square

$$A_{ij} = \frac{\sum_{k=1}^{K}(Y_{ijk} - \bar{Y}_{ij})^2}{K-1}.$$

For the no-reference overall estimator, define the subject-level within term

$$A_{i\cdot} = \frac{1}{J}\sum_{j=1}^{J} A_{ij},$$

the subject-wide mean

$$\bar{Y}_{i\cdot\cdot} = \frac{1}{J}\sum_{j=1}^{J} \bar{Y}_{ij},$$

and the subject-level denominator

$$B_{Ni} = \frac{\sum_{j=1}^{J}(\bar{Y}_{ij} - \bar{Y}_{i\cdot\cdot})^2}{J-1} + \left(1 - \frac{1}{K}\right) A_{i\cdot}.$$

With $\bar{A} = n^{-1}\sum_i A_{i\cdot}$ and $\bar{B} = n^{-1}\sum_i B_{Ni}$, the overall no-reference estimator is

$$\widehat{\mathrm{CIA}}_N^{\mathrm{raw}} = \frac{\bar{A}}{\bar{B}}.$$

For the reference overall estimator, let method `R` be the retained reference method. Define

$$A_{iR} = A_{i,R}$$

and

$$B_{Ri} = \frac{\sum_{j\neq R}(\bar{Y}_{ij} - \bar{Y}_{iR})^2}{J-1} + \left(1 - \frac{1}{K}\right)\frac{\sum_{j\neq R} A_{ij}}{J-1} + \left(1 - \frac{1}{K}\right) A_{iR}.$$

With $\bar{A}_R = n^{-1}\sum_i A_{iR}$ and $\bar{B}_R = n^{-1}\sum_i B_{Ri}$, the overall reference estimator is

$$\widehat{\mathrm{CIA}}_R^{\mathrm{raw}} = \frac{2\bar{A}_R}{\bar{B}_R}.$$

Two estimators are available. `estimator = "mom_unconstrained"` reports the raw ratio estimator directly. `estimator = "vc_constrained"` applies the non-negative variance-component boundary from the cited papers. In the no-reference setting this uses $\hat{\tau}^2_{\mathrm{raw}} = \bar{B} - \bar{A}$, and in the reference setting it uses $\hat{\tau}^2_{\mathrm{raw}} = \bar{B}_R/2 - \bar{A}_R$. The constrained estimator then sets $\hat{\tau}^2 = \max(\hat{\tau}^2_{\mathrm{raw}}, 0)$ and reports $\widehat{\mathrm{CIA}} = \hat{\sigma}^2_W/(\hat{\sigma}^2_W + \hat{\tau}^2)$ on the corresponding scale. This applies the boundary on the implied inter-method variance component rather than clamping CIA directly.

When confidence intervals are requested, pairwise CIA uses a large-sample delta-method normal interval by default and also supports subject-bootstrap percentile intervals. For overall CIA, delta-method normal intervals are available for the unconstrained moment estimator, and subject-bootstrap percentile intervals are also available.

High CIA indicates stronger individual agreement. The FDA individual bioequivalence boundary `IEC <= 2.4948` corresponds to `CIA >= 0.445`, and `CIA >= 0.8` is sometimes used as a stronger practical rule. Such thresholds are context-dependent and are not hard-coded by this function.

Missing rows in the required columns are removed before estimation and the counts are recorded in `attr(x, "diagnostics")`.

Value

For `scope = "overall"`, a one-row data frame with class `c("cia_overall", "cia", "data.frame")`. For `scope = "pairwise"` and `ci = FALSE`, a dense matrix-style object using the package's standard correlation-result infrastructure. For `scope = "pairwise"` and `ci = TRUE`, a list with elements `est`, `lwr.ci`, and `upr.ci`, classed as `c("cia", "cia_ci")`.

Author(s)

Thiago de Paula Oliveira

References

Barnhart HX, Kosinski AS, Haber M. (2007). Assessing individual agreement. *Journal of Biopharmaceutical Statistics*, 17(4), 697-719. doi:10.1080/10543400701329489

Barnhart HX, Haber M, Lokhnygina Y, Kosinski AS. (2007). Comparison of concordance correlation coefficient and coefficient of individual agreement in assessing agreement. *Journal of Biopharmaceutical Statistics*, 17(4), 721-738.

Pan Y, Gao J, Haber M, Barnhart HX. (2010). Estimation of coefficients of individual agreement (CIA's) for quantitative and binary data using SAS and R. *Computer Methods and Programs in Biomedicine*.

See Also

`ccc`, `ba`, and `prob_agree`. `cia()` answers the question "Are methods interchangeable for individual subjects once within-method replicate disagreement is taken into account?". In contrast, `ccc()` answers "How well do two paired measurements agree overall, combining correlation with location/scale agreement?".

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:30)
methods <- c("A", "B", "C")
replicates <- sprintf("r%02d", 1:20)
dat <- expand.grid(
  subject = subjects,
  method = methods,
  replicate = replicates,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
subject_effect <- stats::rnorm(length(subjects), sd = 3)
method_shift <- c(A = 0, B = 0.15, C = -0.10)
method_sd <- c(A = 0.45, B = 0.45, C = 0.30)
dat$value <- subject_effect[match(dat$subject, subjects)] +
  method_shift[dat$method] +
  stats::rnorm(nrow(dat), sd = method_sd[dat$method])

fit_overall <- cia(
  dat,
```

```
  response = "value",
  subject = "subject",
  method = "method",
  replicate = "replicate",
  scope = "overall"
)
print(fit_overall)
summary(fit_overall)
estimate(fit_overall)
tidy(fit_overall)
plot(fit_overall)

fit_pairwise <- cia(
  dat,
  response = "value",
  subject = "subject",
  method = "method",
  replicate = "replicate",
  scope = "pairwise"
)
print(fit_pairwise)
summary(fit_pairwise)
tidy(fit_pairwise)
plot(fit_pairwise)
```

`cia_rm` *Repeated-Measures Coefficient of Individual Agreement*

Description

Computes pairwise repeated-measures coefficients of individual agreement (CIA) from long-format matched repeated-measures data using the categorical condition ANOVA formulation of Haber, Gao, and Barnhart (2010).

Usage

```
cia_rm(
  data,
  response,
  subject,
  method = NULL,
  time = NULL,
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  estimator = c("vc_constrained", "mom_unconstrained"),
  inference = c("delta", "bootstrap", "none"),
  verbose = FALSE,
```

```
  digits = 4,
  use_message = TRUE,
  homogeneous = FALSE,
  B = 1000L,
  seed = NULL,
  ...
)

## S3 method for class 'cia_rm'
summary(
  object,
  digits = 4,
  ci_digits = 3,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'cia_rm'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.cia_rm'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'cia_rm'
plot(
  x,
```

```
  title = NULL,
  facet_by_pair = FALSE,
  facet_scales = c("fixed", "free_y"),
  show_ci = NULL,
  show_common = TRUE,
  point_size = 2.2,
  line_size = 0.7,
  ci_alpha = 0.16,
  ci_linewidth = 0.45,
  ...
)
```

Arguments

data	A data frame in long format.
response	Character scalar naming the numeric response column.
subject	Character scalar naming the subject identifier column.
method	Character scalar naming the method column. Required.
time	Character scalar naming the repeated condition column. Required.
ci	Logical; if TRUE, attach confidence intervals.
conf_level	Confidence level for intervals when ci = TRUE.
n_threads	Positive integer thread hint passed to the C++ backend.
estimator	One of "vc_constrained" or "mom_unconstrained". "vc_constrained" is the default bounded variance-component estimator. It constrains the estimated subject-method variance component to be non-negative before converting it back to CIA. "mom_unconstrained" is the raw method-of-moments CIA ratio and may exceed 1.
inference	One of "delta", "bootstrap", or "none". The delta method is the default when ci = TRUE.
verbose	Logical; if TRUE, emit progress messages for CI work.
digits	Integer print precision carried on the returned object.
use_message	Logical; if TRUE, use package messaging helpers when verbose = TRUE.
homogeneous	Logical; if TRUE, also compute the pooled/common CIA from the reduced model without method-time interaction.
B	Number of subject-bootstrap resamples when ci = TRUE.
seed	Optional positive integer seed for reproducible bootstrap resampling.
...	Additional arguments passed to ggplot2::theme().
object	A cia_rm object.
ci_digits	Integer; number of digits for confidence interval bounds.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.

width	Optional display width.
show_ci	Logical; if TRUE and confidence intervals are present, show them on the plot. By default this is inferred from the object class.
x	A summary.cia_rm object.
title	Optional plot title.
facet_by_pair	Logical; if TRUE, draw one panel per method pair.
facet_scales	Passed to ggplot2::facet_wrap(scales = ...) when facet_by_pair = TRUE. One of "fixed" or "free_y".
show_common	Logical; if TRUE and common estimates are available, draw dashed horizontal reference lines for the pooled/common CIA.
point_size	Numeric point size passed to ggplot2::geom_point().
line_size	Numeric line width passed to ggplot2::geom_line().
ci_alpha	Numeric alpha used for confidence ribbons in continuous plots.
ci_linewidth	Numeric line width used for confidence-interval error bars in categorical plots.

Details

cia_rm() is for matched repeated measurements under conditions such as raters, visits, laboratories, treatments, or time points. It is not a technical-replicate estimator. If the same subject has true technical replicates within each subject-method cell, use cia() instead.

Let Y_ijk denote the measurement on subject i, with method j, under condition k, and consider the model

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + (\alpha\beta)_{ij} + (\alpha\gamma)_{ik} + (\beta\gamma)_{jk} + e_{ijk}.$$

Here subject is random, method and time/condition are fixed, subject-method and subject-time are random interactions, method-time is a fixed interaction, and e_ijk is the residual.

For each method pair and condition k, the repeated-measures CIA is

$$\psi_k = \frac{2\sigma_e^2}{d_k^2 + 2\sigma_{\alpha\beta}^2 + 2\sigma_e^2},$$

where d_k is the condition-specific mean difference between the two methods, $\sigma_{\alpha\beta}^2$ is the subject-method variance component, and σ_e^2 is the residual variance component.

The function fits this estimator separately to every method pair. The homogeneity of agreement across conditions is tested by the method-time interaction:

$$F = \frac{MS_{\text{method}\times\text{time}}}{MS_{\text{error}}}.$$

The returned homogeneity_F attribute stores this test statistic for each method pair, and homogeneity_p stores the corresponding upper-tail F-test p-value using degrees of freedom df_method_time and df_error. Larger homogeneity_F values and smaller homogeneity_p values indicate stronger evidence that agreement changes across conditions, meaning that the condition-specific CIA curves should be interpreted directly rather than relying on a single pooled/common coefficient. Conversely, a small homogeneity_F and a non-small homogeneity_p indicate that the data do not show

strong evidence of method-by-condition heterogeneity, so a common estimate may be a reasonable summary if it is scientifically meaningful.

When `homogeneous = TRUE`, the function also reports a common or pooled CIA for each method pair from the reduced model that pools the method-time sum of squares into the residual term. That common estimate is meaningful only when agreement is reasonably homogeneous across conditions.

Confidence intervals can be computed by a delta-method normal approximation or by a subject-level percentile bootstrap. The delta-method interval is the default. For each method pair and condition `k`, let $\hat{\psi}_k$ be the ANOVA estimator, let g_k be its numerical gradient with respect to the subject-level moment vector, and let Σ be the empirical covariance matrix of that moment vector divided by the number of subjects. The delta-method standard error is

$$\widehat{\mathrm{se}}(\hat{\psi}_k) = \sqrt{\max\left(g_k^\top \Sigma g_k, 0\right)}.$$

The raw normal interval is

$$\hat{\psi}_k \pm z_{1-\alpha/2}\, \widehat{\mathrm{se}}(\hat{\psi}_k).$$

Under the bootstrap option, the interval is given by the empirical percentile limits from subject-resampled estimates. The argument `estimator` controls whether the reported result is the literal method-of-moments ratio or a bounded variance-component variant. Under `estimator = "vc_constrained"`, the estimated subject-method variance component is constrained to be non-negative before converting it back to CIA, and the reported interval limits are also truncated to the CIA parameter space,

$$\mathrm{lwr} = \max(0, \mathrm{lwr}_{\mathrm{raw}}), \qquad \mathrm{upr} = \min(1, \mathrm{upr}_{\mathrm{raw}}).$$

This keeps the reported interval inside the CIA parameter space `[0, 1]`. Use `estimator = "mom_unconstrained"` to inspect the literal raw method-of-moments estimator and the corresponding unbounded interval on the estimator scale.

The current implementation requires exactly one observation in every subject-method-time cell. It therefore targets the balanced repeated-measures ANOVA setting from the cited paper and returns an error otherwise.

Value

If `ci = FALSE`, the result is a list of class `c("cia_rm", "cia")`. If `ci = TRUE`, the result is a list of class `c("cia_rm", "cia_ci", "cia")`.

Main components:

- `est`: condition-specific pairwise CIA estimates.
- `common`: pairwise overall summary CIA estimates from the reduced homogeneous model.
- `condition`: the ordered condition/time labels used in the output.
- `se`: standard errors for `est` when `ci = TRUE`.
- `lwr.ci` and `upr.ci`: lower and upper confidence limits for `est` when `ci = TRUE`.
- `common.se`: standard errors for `common` when `ci = TRUE`.
- `common.lwr.ci` and `common.upr.ci`: lower and upper confidence limits for `common` when `ci = TRUE`.

When there are three or more methods, additional overall multi-method components are returned:

- `overall`: condition-specific overall CIA across all methods.
- `overall.common`: the overall summary CIA across all methods when `homogeneous = TRUE`.
- `overall.se`: standard errors for `overall` when `ci = TRUE`.
- `overall.lwr.ci` and `overall.upr.ci`: lower and upper confidence limits for `overall` when `ci = TRUE`.
- `overall.common.se`: standard errors for `overall.common` when `ci = TRUE`.
- `overall.common.lwr.ci` and `overall.common.upr.ci`: lower and upper confidence limits for `overall.common` when `ci = TRUE`.

The object carries pairwise ANOVA diagnostics as attributes, including `sigma2_error`, `sigma2_subject_method`, `repeatability`, `homogeneity_F`, `homogeneity_p`, `df_method_time`, `df_error`, `n_obs`, `n_subjects`, `n_methods`, `n_times`, `time_levels`, and `method_levels`. Here `homogeneity_F` is the method-time interaction test statistic for each method pair, and `homogeneity_p` is the corresponding p-value for the null hypothesis of homogeneous agreement across conditions.

Author(s)

Thiago de Paula Oliveira

References

Haber M, Gao J, Barnhart HX. (2010). Evaluation of agreement between measurement methods from data with matched repeated measurements via the coefficient of individual agreement. *Journal of Data Science*, 8, 457-469.

Barnhart HX, Kosinski AS, Haber M. (2007). Assessing individual agreement. *Journal of Biopharmaceutical Statistics*, 17(4), 697-719.

See Also

`ccc_rm_reml()`, `icc_rm_reml()`, and `cia()`.

Examples

```
# Example 1
set.seed(1)
dat_rater <- expand.grid(
  id = factor(sprintf("s%02d", 1:8)),
  method = factor(c("Device_A", "Device_B")),
  time = factor(c("Rater_1", "Rater_2", "Rater_3")),
  KEEP.OUT.ATTRS = FALSE
)
subj_eff <- rnorm(nlevels(dat_rater$id), sd = 1)[dat_rater$id]
method_shift <- c(Device_A = 0, Device_B = 0.25)[dat_rater$method]
rater_shift <- c(Rater_1 = 0, Rater_2 = 0.15, Rater_3 = -0.10)[dat_rater$time]
interaction_shift <- ifelse(
  dat_rater$method == "Device_B" & dat_rater$time == "Rater_3",
  0.12, 0
)
dat_rater$y <- subj_eff + method_shift + rater_shift +
```

```
  interaction_shift + rnorm(nrow(dat_rater), sd = 0.25)

fit_rater <- cia_rm(
  dat_rater,
  response = "y",
  subject = "id",
  method = "method",
  time = "time",
  homogeneous = TRUE
)
print(fit_rater)
summary(fit_rater)
estimate(fit_rater)
tidy(fit_rater)
plot(fit_rater)

# Example 2
set.seed(2)
dat_time <- expand.grid(
  id = factor(sprintf("s%02d", 1:10)),
  method = factor(c("Assay_A", "Assay_B", "Assay_C", "Assay_D")),
  time = factor(
    c("baseline", "week2", "month1", "month2", "month3"),
    levels = c("baseline", "week2", "month1", "month2", "month3")
  ),
  KEEP.OUT.ATTRS = FALSE
)
subj_eff <- rnorm(nlevels(dat_time$id), sd = 0.9)[dat_time$id]
method_shift <- c(Assay_A = 0, Assay_B = 0.20, Assay_C = -0.10, Assay_D = 0.35)[dat_time$method]
time_shift <- c(
  baseline = 0, week2 = 0.10, month1 = 0.22, month2 = 0.32, month3 = 0.42
)[dat_time$time]
interaction_shift <- ifelse(dat_time$method == "Assay_B" & dat_time$time == "month3", 0.10, 0) +
  ifelse(dat_time$method == "Assay_D" & dat_time$time == "month2", -0.08, 0)
dat_time$y <- subj_eff + method_shift + time_shift +
  interaction_shift + rnorm(nrow(dat_time), sd = 0.30)

fit_time <- cia_rm(
  dat_time,
  response = "y",
  subject = "id",
  method = "method",
  time = "time",
  ci = TRUE
)
print(fit_time)
plot(fit_time)

# Example 3
set.seed(3)
dat_days <- expand.grid(
  id = factor(sprintf("s%02d", 1:12)),
  method = factor(c("Sensor_A", "Sensor_B", "Sensor_C")),
```

```
  time = 1:15,
  KEEP.OUT.ATTRS = FALSE
)
subj_eff <- rnorm(nlevels(dat_days$id), sd = 0.8)[dat_days$id]
method_shift <- c(Sensor_A = 0, Sensor_B = 0.15, Sensor_C = -0.08)[dat_days$method]
day_trend <- 0.05 * dat_days$time
interaction_shift <- ifelse(dat_days$method == "Sensor_B", 0.01 * dat_days$time, 0) +
  ifelse(dat_days$method == "Sensor_C", -0.005 * dat_days$time, 0)
dat_days$y <- subj_eff + method_shift + day_trend +
  interaction_shift + rnorm(nrow(dat_days), sd = 0.22)

fit_days <- cia_rm(
  dat_days,
  response = "y",
  subject = "id",
  method = "method",
  time = "time",
  ci = TRUE
)
plot(fit_days, facet_by_pair = TRUE)
```

---

cohen_kappa *Pairwise Cohen's Kappa for Nominal Ratings*

Description

Computes unweighted Cohen's kappa for either a pair of nominal rating vectors or all pairwise combinations of nominal columns in a matrix or data frame.

Usage

```
cohen_kappa(
  data,
  y = NULL,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'cohen_kappa'
summary(object, digits = 4, ci_digits = 3, p_digits = 4, ...)
```

```
## S3 method for class 'summary.cohen_kappa'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'cohen_kappa'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'cohen_kappa'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

data	In matrix mode, a matrix or data frame whose rows are observational units and whose columns are raters or classifiers. Supported column types are factor, ordered factor, character, logical, integer, and numeric, all treated as nominal categories here. If the ratings are truly ordinal and disagreements should be weighted by distance, use `weighted_kappa()` instead. In two-vector mode, the first nominal rating vector.
y	Optional second nominal rating vector. When supplied, the function returns a single Cohen's kappa estimate for `data` and `y`.
na_method	Character scalar controlling missing-data handling. `"error"` rejects missing

	values. `"complete"` applies listwise deletion across retained columns before computing the matrix. In matrix mode, `"pairwise"` uses pair-specific complete observations.
`ci`	Logical; if `TRUE`, attach large-sample delta-method confidence intervals.
`p_value`	Logical; if `TRUE`, attach large-sample delta-method Wald test statistics and p-values.
`conf_level`	Confidence level used when `ci = TRUE`. Default is `0.95`.
`n_threads`	Integer ≥ 1. Number of OpenMP threads used by the matrix C++ backend. Defaults to `getOption("matrixCorr.threads", 1L)`.
`output`	Output representation for matrix mode. • `"matrix"` (default): full dense matrix. • `"sparse"`: sparse matrix from **Matrix** containing only retained entries after thresholding. • `"edge_list"`: long-form data frame representation.
`threshold`	Non-negative absolute-value filter for non-matrix outputs: retain entries with `abs(value) >= threshold`. Must be `0` when `output = "matrix"`.
`diag`	Logical; whether to include diagonal entries in sparse and edge-list outputs.
`...`	Additional theme arguments.
`object`	A scalar or matrix-style `cohen_kappa` object.
`digits`	Integer; number of decimal places for displayed values.
`ci_digits`	Integer; number of decimal places for confidence limits.
`p_digits`	Integer; number of decimal places for p-values.
`x`	A scalar or matrix-style `cohen_kappa` object.
`n`	Optional preview row threshold.
`topn`	Optional number of leading/trailing rows to show when truncated.
`max_vars`	Optional maximum number of visible columns.
`width`	Optional display width.
`show_ci`	One of `"yes"` or `"no"`.
`title`	Optional plot title.
`low_color`	Fill/color used for negative agreement.
`high_color`	Fill/color used for positive agreement.
`mid_color`	Unused placeholder for API consistency with matrix heatmaps.
`value_text_size`	Text size for the estimate label.
`ci_text_size`	Text size for the CI label.
`show_value`	Logical; whether to print the estimate and CI labels.

Details

Cohen's kappa is an agreement coefficient for two raters assigning the same units to mutually exclusive nominal categories. For contingency-table cell proportions p_{ij},

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

where $p_o = \sum_i p_{ii}$ is the observed agreement and $p_e = \sum_i p_{i+}p_{+i}$ is the chance agreement implied by the marginal category proportions.

This implementation is strictly the original unweighted nominal-scale Cohen's kappa. If the categories are ordinal and near disagreements should count as less severe than distant disagreements, use `weighted_kappa()` instead.

In matrix mode, columns are treated as raters/classifiers and rows as shared observational units. All pairwise Cohen's kappas between columns are computed. Category labels are encoded in R before dispatch to C++, using a common label mapping across all columns so that identical labels in different columns correspond to the same category code.

Missing values are encoded as `NA_integer_` before entering the C++ backend. With `na_method = "error"`, missing values are rejected before computation. With `na_method = "complete"`, listwise deletion is applied across retained columns. With `na_method = "pairwise"`, each pair uses its own complete observations. Pairwise complete counts are stored in `attr(x, "diagnostics")$n_complete`.

Confidence intervals and standard errors. The implementation uses the exact large-sample formula. Let p_{ab} be the empirical cell proportions, $q_a = \sum_b p_{ab}$ the row margins, $r_b = \sum_a p_{ab}$ the column margins, and $D = 1 - p_e$. For each cell (a, b), define the influence contribution

$$g_{ab} \;=\; \frac{\mathbf{1}(a = b)\, D + (p_o - 1)\,(r_a + q_b)}{D^2}.$$

The variance estimator used by the code is

$$\widehat{\mathrm{Var}}(\hat{\kappa}) \;=\; \frac{1}{n}\left(\sum_{a,b} p_{ab} g_{ab}^2 - \Big(\sum_{a,b} p_{ab} g_{ab}\Big)^2\right),$$

with n the number of complete paired ratings and $\widehat{\mathrm{se}}(\hat{\kappa}) = \sqrt{\widehat{\mathrm{Var}}(\hat{\kappa})}$. The confidence interval is the Wald interval

$$\hat{\kappa} \pm z_{1-\alpha/2}\, \widehat{\mathrm{se}}(\hat{\kappa}),$$

truncated to $[-1, 1]$ in the returned result.

Value

If `y` is supplied, a scalar S3 object of class `"cohen_kappa"` backed by a numeric value, with attributes `diagnostics`, and optionally `ci`, `inference`, and `conf.level`. Otherwise a symmetric matrix-style result with estimator class `cohen_kappa`.

Author(s)

Thiago de Paula Oliveira

References

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37-46. doi:10.1177/001316446002000104

See Also

`weighted_kappa()` for two-rater ordered-category agreement with distance-sensitive disagreement weights; `multirater_kappa()` for panel-level nominal agreement among three or more raters.

Examples

```
x <- factor(c("A", "A", "B", "B", "A", "B"))
y <- factor(c("A", "B", "B", "B", "A", "A"))
cohen_kappa(x, y)

raters <- data.frame(
  r1 = factor(c("low", "low", "high", "high", "mid")),
  r2 = factor(c("low", "mid", "high", "high", "mid")),
  r3 = c("low", "low", "high", "mid", "mid")
)
ck <- cohen_kappa(raters)
print(ck)
summary(ck)
estimate(ck)
tidy(ck)
plot(ck)
```

---

dcor *Pairwise Distance Correlation (dCor)*

Description

Computes conventional pairwise distance correlation for numeric columns using a high-performance C++ backend. Distance correlation is a nonnegative sample measure of dependence that can capture nonlinear and nonmonotonic relationships. By default, `dcor()` returns the conventional sample distance correlation R_n; set `squared = TRUE` to return R_n^2.

Usage

```
dcor(
  data,
  squared = FALSE,
  na_method = c("error", "pairwise", "complete"),
  p_value = FALSE,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
```

```
  diag = TRUE,
  ...
)

## S3 method for class 'dcor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'dcor'
plot(
  x,
  title = NULL,
  low_color = "white",
  high_color = "steelblue1",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'dcor'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.dcor'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
```

```
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns are dropped. Columns must be numeric.
squared	Logical; if FALSE (default), return R_n. If TRUE, return conventional squared distance correlation R_n^2.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each association on its own pairwise complete-case overlap. "complete" performs listwise deletion once across the retained numeric columns.
p_value	Logical (default FALSE). Compatibility option. If TRUE, attach the Székely-Rizzo t-test metadata computed from the signed bias-corrected statistic returned by bcdcor(), not from the conventional R_n values in the main matrix.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates: "matrix", "sparse", or "edge_list".
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
x	An object of class summary.dcor.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
title	Plot title. Default is "Distance correlation heatmap".
low_color	Colour for zero correlation. Default is "white".
high_color	Colour for strong correlation. Default is "steelblue1".
value_text_size	Font size for displaying values. Default is 4.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class dcor.

Details

For distances $a_{ij} = |x_i - x_j|$, define the conventional doubly centred distances

$$A_{ij} = a_{ij} - \bar{a}_{i\cdot} - \bar{a}_{\cdot j} + \bar{a}_{\cdot\cdot}$$

Define B_{ij} analogously for y. The sample squared distance covariance is

$$V_n^2(x, y) = \frac{1}{n^2} \sum_{i,j} A_{ij} B_{ij},$$

with $V_n^2(x, x)$ and $V_n^2(y, y)$ defined analogously. The conventional squared distance correlation is

$$R_n^2(x, y) = \frac{V_n^2(x, y)}{\sqrt{V_n^2(x, x) V_n^2(y, y)}}.$$

`dcor()` returns $R_n = \sqrt{R_n^2}$ unless `squared = TRUE`.

Under the required finite first-moment conditions, population distance correlation is zero if and only if the variables are independent. This population property should not be read as a sample-level test by itself.

Value

A symmetric correlation result where the `(i, j)` entry is conventional distance correlation R_n, or R_n^2 when `squared = TRUE`. When `p_value = TRUE`, the `inference` attribute contains `bcdcor`, `statistic`, `parameter`, and `p_value`; the t-test uses the signed U-centred statistic in `inference$bcdcor`.

Invisibly returns `x`.

A `ggplot` object representing the heatmap.

Difference between `dcor()` and `bcdcor()`

`dcor()` returns conventional nonnegative distance correlation R_n. `dcor(squared = TRUE)` returns conventional R_n^2. `bcdcor()` returns a U-centred bias-corrected squared distance-correlation statistic that may be negative in finite samples and is used by the bias-corrected dCor t-test. In finite samples, `bcdcor(x)` is not generally equal to `dcor(x)^2`.

Note

Requires $n \geq 4$. Columns with zero distance variance yield `NA` in their off-diagonal row/column entries.

Author(s)

Thiago de Paula Oliveira

References

Szekely, G. J., Rizzo, M. L., & Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *Annals of Statistics*, 35(6), 2769-2794.

Szekely, G. J., & Rizzo, M. L. (2013). The distance correlation t-test of independence. *Journal of Multivariate Analysis*, 117, 193-213.

See Also

bcdcor(), robust_dcor()

Examples

```
# Conventional distance correlation
set.seed(42)
n <- 200
x <- rnorm(n)
y <- 0.7 * x + sqrt(1 - 0.7^2) * rnorm(n)
X <- cbind(x = x, y = y)

# dcor() returns conventional distance correlation R.
# squared = TRUE returns conventional R^2.
dcor(X)
dcor(X, squared = TRUE)

D <- dcor(X)
D2 <- dcor(X, squared = TRUE)

D2["x", "y"]
D["x", "y"]^2

# Pearson can be close to zero for this symmetric nonlinear relationship,
# while distance correlation remains positive.
set.seed(123)
n <- 300
x <- rnorm(n)
y <- x^2 + rnorm(n, sd = 0.25)
X <- cbind(x = x, y = y)

cor(x, y)
dcor(X)

# Pairwise distance-correlation matrix across columns
if (requireNamespace("MASS", quietly = TRUE)) {
  set.seed(7)
  p <- 5
  n <- 150
  rho <- 0.6
  Sigma <- rho^abs(outer(seq_len(p), seq_len(p), "-"))

  X <- MASS::mvrnorm(
    n,
    mu = rep(0, p),
    Sigma = Sigma
  )
  colnames(X) <- paste0("V", seq_len(p))

  D <- dcor(X)
  D[1:3, 1:3]
}
```

```
# Backward-compatible inference through dcor()
fit <- dcor(cbind(x = x, y = y), p_value = TRUE)
fit

# The displayed matrix contains conventional dCor.
# The attached t-test is based on the signed bias-corrected statistic.
inf <- attr(fit, "inference")
inf$bcdcor
inf$p_value

# Missing-data modes
X_miss <- cbind(
  a = c(1, 2, NA, 4, 5, 6),
  b = c(2, 3, 4, 5, NA, 7),
  c = c(6, 5, 4, 3, 2, 1)
)

dcor(X_miss, na_method = "complete")
dcor(X_miss, na_method = "pairwise")

# Edge-list output for larger matrices
set.seed(11)
X <- matrix(rnorm(500), nrow = 100, ncol = 5)
colnames(X) <- paste0("V", seq_len(ncol(X)))

dcor(
  X,
  output = "edge_list",
  threshold = 0.2,
  diag = FALSE
)
```

`deprecated-matrixCorr` *Deprecated Compatibility Wrappers*

Description

Temporary wrappers for functions renamed in `matrixCorr` 1.0.0. These wrappers preserve the pre-1.0 entry points while warning that they will be removed in 2.0.0.

Usage

```
bland_altman(
  group1,
  group2,
  two = 1.96,
  mode = 1L,
```

```
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE
)

bland_altman_repeated(
  data = NULL,
  response,
  subject,
  method,
  time,
  two = 1.96,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  include_slope = FALSE,
  use_ar1 = FALSE,
  ar1_rho = NA_real_,
  max_iter = 200L,
  tol = 1e-06,
  verbose = FALSE
)

biweight_mid_corr(
  data,
  c_const = 9,
  max_p_outliers = 1,
  pearson_fallback = c("hybrid", "none", "all"),
  na_method = c("error", "pairwise", "complete"),
  mad_consistent = FALSE,
  w = NULL,
  sparse_threshold = NULL,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

distance_corr(data, na_method = c("error", "pairwise", "complete"), ...)

partial_correlation(
  data,
  method = c("oas", "ridge", "sample"),
  lambda = 0.001,
  return_cov_precision = FALSE,
  ci = FALSE,
  conf_level = 0.95,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
```

```
  diag = TRUE
)

ccc_lmm_reml(
  data,
  response,
  rind,
  method = NULL,
  time = NULL,
  interaction = FALSE,
  max_iter = 100,
  tol = 1e-06,
  Dmat = NULL,
  Dmat_type = c("time-avg", "typical-visit", "weighted-avg", "weighted-sq"),
  Dmat_weights = NULL,
  Dmat_rescale = TRUE,
  ci = FALSE,
  conf_level = 0.95,
  ci_mode = c("auto", "raw", "logit"),
  verbose = FALSE,
  digits = 4,
  use_message = TRUE,
  ar = c("none", "ar1"),
  ar_rho = NA_real_,
  slope = c("none", "subject", "method", "custom"),
  slope_var = NULL,
  slope_Z = NULL,
  drop_zero_cols = TRUE,
  vc_select = c("auto", "none"),
  vc_alpha = 0.05,
  vc_test_order = c("subj_time", "subj_method"),
  include_subj_method = NULL,
  include_subj_time = NULL,
  sb_zero_tol = 1e-10
)

ccc_pairwise_u_stat(
  data,
  response,
  method,
  subject,
  time = NULL,
  Dmat = NULL,
  delta = 1,
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE
```

```
)
```

Arguments

group1, group2	Numeric vectors of equal length.
two	Positive scalar; the multiple of the standard deviation used to define the limits of agreement.
mode	Integer; 1 uses group1 - group2, 2 uses group2 - group1.
conf_level	Confidence level.
n_threads	Integer number of OpenMP threads.
verbose	Logical; print brief progress or diagnostic output.
data	A data.frame, matrix, or repeated-measures dataset accepted by the corresponding replacement function.
response	Numeric response vector or column name, depending on the target method.
subject	Subject identifier or subject column name.
method	Method label or method column name.
time	Replicate/time index or time column name.
include_slope	Logical; whether to estimate proportional bias.
use_ar1	Logical; whether to use AR(1) within-subject correlation.
ar1_rho	AR(1) parameter.
max_iter, tol	EM control parameters.
c_const	Positive numeric Tukey biweight tuning constant.
max_p_outliers	Numeric in (0, 1]; optional cap on the maximum proportion of outliers on each side.
pearson_fallback	Character fallback policy used by bicor().
na_method	Missing-data policy forwarded to the replacement function when supported.
mad_consistent	Logical; if TRUE, uses the consistency-corrected MAD.
w	Optional vector of case weights.
sparse_threshold	Optional threshold controlling sparse output.
output	Output representation for the computed estimates.
threshold	Non-negative absolute-value filter for non-matrix outputs.
diag	Logical; whether to include diagonal entries in non-matrix outputs.
...	Additional arguments forwarded to the replacement function when supported.
lambda	Numeric regularisation strength used by pcorr().
return_cov_precision	Logical; if TRUE, also return covariance and precision matrices.
ci	Logical; if TRUE, request confidence intervals when supported by the replacement function.

rind	Character; column identifying subjects, forwarded as subject to ccc_rm_reml().
interaction	Logical; forwarded to ccc_rm_reml().
Dmat	Optional distance matrix forwarded to ccc_rm_reml().
Dmat_type	Character selector controlling how Dmat is constructed.
Dmat_weights	Optional weights used when Dmat_type requires them.
Dmat_rescale	Logical; whether to rescale Dmat.
ci_mode	Character selector for the confidence-interval scale used by ccc_rm_reml().
digits	Display precision forwarded to ccc_rm_reml().
use_message	Logical; whether the deprecated wrapper emits a lifecycle message.
ar	Character selector for the within-subject residual correlation model.
ar_rho	Numeric AR(1) parameter.
slope	Character selector for the proportional-bias slope structure.
slope_var	Optional covariance matrix for custom slopes.
slope_Z	Optional design matrix for custom slopes.
drop_zero_cols	Logical; whether zero-variance design columns are dropped.
vc_select	Character selector controlling variance-component selection.
vc_alpha	Significance level used in variance-component selection.
vc_test_order	Character vector controlling the variance-component test order.
include_subj_method	Optional logical override for the subject-by-method component.
include_subj_time	Optional logical override for the subject-by-time component.
sb_zero_tol	Numerical tolerance used when stabilising the scale-bias term.
delta	Numeric power exponent for U-statistics distances.

Details

Renamed functions:

- bland_altman() -> ba()
- bland_altman_repeated() -> ba_rm()
- biweight_mid_corr() -> bicor()
- distance_corr() -> dcor()
- partial_correlation() -> pcorr()
- ccc_lmm_reml() -> ccc_rm_reml()
- ccc_pairwise_u_stat() -> ccc_rm_ustat()

The deprecated wrappers will be removed in matrixCorr 2.0.0.

`estimate` *Access MatrixCorr Estimates, Confidence Intervals, and Tidy Results*

Description

Lightweight accessors for matrixCorr result objects. These functions do not change the stored object structure; they just provide a stable way to extract the estimate matrix, confidence intervals, or a pairwise data frame without reading attributes directly.

Usage

```
estimate(x, ...)

tidy(x, ...)

ci(x, ...)

## S3 method for class 'corr_result'
estimate(x, ...)

## S3 method for class 'summary.corr_result'
estimate(x, ...)

## S3 method for class 'summary.matrixCorr'
estimate(x, ...)

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

## S3 method for class 'corr_result'
ci(x, ...)

## Default S3 method:
ci(x, ...)

## S3 method for class 'ba'
ci(x, ...)

## S3 method for class 'ba_matrix'
ci(x, ...)

## S3 method for class 'ba_repeated'
ci(x, ...)

## S3 method for class 'ba_repeated_matrix'
ci(x, ...)
```

```
## S3 method for class 'ccc'
ci(x, ...)

## S3 method for class 'ccc_ci'
ci(x, ...)

## S3 method for class 'ccc_glmm'
ci(x, ...)

## S3 method for class 'cia'
ci(x, ...)

## S3 method for class 'cia_ci'
ci(x, ...)

## S3 method for class 'cia_rm'
ci(x, ...)

## S3 method for class 'cohen_kappa'
ci(x, ...)

## S3 method for class 'gwet_ac'
ci(x, ...)

## S3 method for class 'icc'
ci(x, ...)

## S3 method for class 'icc_overall'
ci(x, ...)

## S3 method for class 'icc_rm_reml'
ci(x, ...)

## S3 method for class 'krippendorff_alpha'
ci(x, ...)

## S3 method for class 'multirater_kappa'
ci(x, ...)

## S3 method for class 'partial_corr'
ci(x, ...)

## S3 method for class 'prob_agree'
ci(x, ...)

## S3 method for class 'rmcorr'
ci(x, ...)
```

```
## S3 method for class 'rmcorr_matrix'
ci(x, ...)

## S3 method for class 'weighted_kappa'
ci(x, ...)

## S3 method for class 'partial_corr'
estimate(x, ...)

## S3 method for class 'ba'
estimate(x, ...)

## S3 method for class 'ba_matrix'
estimate(x, ...)

## S3 method for class 'ba_repeated'
estimate(x, ...)

## S3 method for class 'ba_repeated_matrix'
estimate(x, ...)

## S3 method for class 'ccc_glmm'
estimate(x, ...)

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

## Default S3 method:
estimate(x, ...)

## S3 method for class 'corr_result'
tidy(x, diag = FALSE, triangle = c("upper", "lower", "full"), ...)

## S3 method for class 'summary.corr_result'
tidy(x, ...)

## S3 method for class 'summary.matrixCorr'
tidy(x, ...)

## S3 method for class 'partial_corr'
tidy(x, diag = FALSE, triangle = c("upper", "lower", "full"), ...)

## S3 method for class 'ba'
tidy(x, ...)

## S3 method for class 'ba_matrix'
tidy(x, ...)
```

```
## S3 method for class 'ba_repeated'
tidy(x, ...)

## S3 method for class 'ba_repeated_matrix'
tidy(x, ...)

## Default S3 method:
tidy(x, diag = FALSE, triangle = c("upper", "lower", "full"), ...)

## S3 method for class 'corr_result'
confint(object, parm, level = NULL, ...)

## S3 method for class 'summary.corr_result'
confint(object, parm, level = NULL, ...)

## S3 method for class 'summary.matrixCorr'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ba'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ba_repeated'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ba_matrix'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ba_repeated_matrix'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ccc'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ccc_ci'
confint(object, parm, level = NULL, ...)

## S3 method for class 'ccc_glmm'
confint(object, parm, level = NULL, ...)

## S3 method for class 'chatterjee_xi_scalar'
confint(object, parm, level = NULL, ...)

## S3 method for class 'cia'
confint(object, parm, level = NULL, ...)

## S3 method for class 'cia_ci'
confint(object, parm, level = NULL, ...)
```

```
## S3 method for class 'cia_rm'
confint(object, parm, level = NULL, ...)

## S3 method for class 'cohen_kappa'
confint(object, parm, level = NULL, ...)

## S3 method for class 'gwet_ac'
confint(object, parm, level = NULL, ...)

## S3 method for class 'icc'
confint(object, parm, level = NULL, ...)

## S3 method for class 'icc_overall'
confint(object, parm, level = NULL, ...)

## S3 method for class 'icc_rm_reml'
confint(object, parm, level = NULL, ...)

## S3 method for class 'krippendorff_alpha'
confint(object, parm, level = NULL, ...)

## S3 method for class 'multirater_kappa'
confint(object, parm, level = NULL, ...)

## S3 method for class 'prob_agree'
confint(object, parm, level = NULL, ...)

## S3 method for class 'partial_corr'
confint(object, parm, level = NULL, ...)

## S3 method for class 'rmcorr'
confint(object, parm, level = NULL, ...)

## S3 method for class 'rmcorr_matrix'
confint(object, parm, level = NULL, ...)

## S3 method for class 'weighted_kappa'
confint(object, parm, level = NULL, ...)
```

Arguments

x, object	A matrixCorr result, summary object, or scalar result.
...	Additional arguments passed to methods.
diag	Logical; include diagonal entries for matrix-style results. Default is FALSE.
triangle	For matrix-style correlation results, which triangle to return from tidy(): "upper" (default), "lower", or "full". The default avoids duplicate pairs for symmetric matrices; directed matrices use all off-diagonal entries unless triangle is supplied explicitly.

parm	Ignored; included for compatibility with stats::confint().
level	Confidence level requested by stats::confint(). MatrixCorr objects store pre-computed intervals; if level differs from the stored level, recompute the original object with the desired level.

Value

estimate() returns the primary estimate in its natural shape: a matrix-like result returns a matrix, an edge-list result returns a data frame, and a scalar result returns a numeric value.

tidy() returns a data frame with columns such as item1, item2, estimate, lwr, upr, diagnostics, and inferential quantities when available.

confint() returns a data frame containing confidence limits when they are available.

ci() returns the stored confidence-interval payload. It is mainly a structured alternative to reading attr(x, "ci") directly.

Author(s)

Thiago de Paula Oliveira

Examples

```
X <- cbind(a = 1:10, b = 1:10 + rnorm(10), c = rnorm(10))
fit <- pearson_corr(X, ci = TRUE)

estimate(fit)
tidy(fit)
ci(fit)
confint(fit)
```

---

gwet_ac *Gwet's AC1 and AC2 Agreement Coefficients*

Description

Estimates Gwet's chance-corrected agreement coefficient for either unweighted nominal agreement (AC1) or weighted agreement (AC2) in two-rater pairwise form or panel-level multi-rater form.

Usage

```
gwet_ac(
  data,
  y = NULL,
  weights = c("unweighted", "linear", "quadratic", "ordinal", "radical", "ratio",
    "circular", "bipolar"),
  levels = NULL,
  input = c("pairwise", "ratings", "counts"),
```

```
  na_method = c("error", "pairwise", "complete", "available"),
  min_raters = 2L,
  by_category = FALSE,
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  se_method = c("asymptotic", "jackknife", "none"),
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  verbose = FALSE,
  ...
)

## S3 method for class 'gwet_ac'
summary(object, digits = 4, ci_digits = 3, p_digits = 4, ...)

## S3 method for class 'summary.gwet_ac'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'gwet_ac'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  show_by_category = FALSE,
  ...
)

## S3 method for class 'gwet_ac'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
```

```
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  type = c("agreement_map", "estimate", "item_agreement", "category_proportion",
    "by_category"),
  bins = 30L,
  ...
)
```

Arguments

data	Input data. If y is supplied, data is the first rater vector. Otherwise: for input = "pairwise", rows are units and columns are raters; for input = "ratings", rows are items and columns are raters; for input = "counts", rows are items and columns are categories containing rater counts.
y	Optional second rater vector for scalar two-rater agreement.
weights	Weight specification. "unweighted" yields AC1. Weighted schemes yield AC2. Supported built-in schemes are "linear", "quadratic", "ordinal", "radical", "ratio", "circular", and "bipolar". A custom numeric agreement-weight matrix may also be supplied. Character schemes must be supplied as a single value.
levels	Optional explicit category labels. This is mainly useful when unused categories should still be retained in the calculation, and is the recommended way to define ordering for ordinal AC2 weights.
input	One of "pairwise", "ratings", or "counts".
na_method	Missing-data rule. For scalar two-rater and pairwise matrix modes, "error" rejects missing values, "complete" applies listwise deletion in matrix mode, and "pairwise" uses complete pairs within each pair of raters. For input = "ratings", "available" retains items with at least min_raters observed ratings. For input = "counts", missing values are not allowed.
min_raters	Minimum number of observed ratings required for an item to be retained in panel mode. Must be at least 2.
by_category	Logical; if TRUE, attach one-vs-all category-wise AC1 estimates for panel-level fits. This is available only for the unweighted coefficient.
ci	Logical; if TRUE, attach confidence intervals.
p_value	Logical; if TRUE, attach test statistics and p-values.
conf_level	Confidence level for intervals. Default is 0.95.
se_method	Standard-error method. "asymptotic" uses the analytic large-sample formula. "jackknife" is retained as an alternative for panel-level fits. "none" disables inferential output.
n_threads	Integer ≥ 1. Number of OpenMP threads used by the C++ backend.
output	One of "matrix", "sparse", or "edge_list" for pairwise matrix mode.
threshold	Numeric threshold used only for thresholded pairwise matrix output.

diag	Logical; include diagonal entries in thresholded pairwise matrix output.
verbose	Logical; if TRUE, emit short progress messages.
...	Unused.
object	A gwet_ac object.
digits	Integer; number of decimal places for displayed values.
ci_digits	Integer; number of decimal places for confidence limits.
p_digits	Integer; number of decimal places for p-values.
x	A gwet_ac object.
n	Optional preview row threshold.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.
width	Optional display width.
show_ci	One of "yes" or "no".
show_by_category	Logical; whether to print attached category-wise panel results when available.
title	Optional plot title.
low_color	Fill/color used for negative agreement.
high_color	Fill/color used for positive agreement.
mid_color	Unused placeholder for API consistency with matrix heatmaps.
value_text_size	Text size for estimate labels in matrix plots.
ci_text_size	Text size for CI labels in matrix plots.
show_value	Logical; whether to print estimate and CI labels.
type	Plot type for panel-level fits: "agreement_map", "estimate", "item_agreement", "category_proportion", or "by_category".
bins	Integer number of bins retained for compatibility with the panel-level item-agreement plot.

Details

For two raters, let w_{ab} denote the agreement weight for cell (a, b), p_{ab} the cell proportion, and $\pi_k = (p_{k+} + p_{+k})/2$. Then

$$P_o = \sum_{a,b} w_{ab} p_{ab},$$

$$P_e = \frac{\sum_{a,b} w_{ab}}{q(q-1)} \sum_{k=1}^{q} \pi_k (1 - \pi_k),$$

and

$$AC = \frac{P_o - P_e}{1 - P_e}.$$

With identity weights $w_{ab} = 1(a = b)$ this is Gwet's AC1, the nominal-agreement coefficient. With non-identity agreement weights this is Gwet's AC2, which extends the same chance-correction

idea to ordered or partially creditable disagreement by letting near disagreements receive larger agreement weights than distant disagreements.

Category ordering for weighted AC2. Built-in weighted schemes use the category order to define near and distant disagreements. If `levels` is supplied, that order is used. Otherwise, factor inputs use their factor levels, numeric/integer/logical inputs use sorted observed values, character inputs use first-observed order, and counts inputs use column order (or `levels`, when supplied). For ordinal analyses, supplying `levels` is recommended whenever first-observed character order is not the intended scale order.

For panel-level counts, if r_{ik} is the number of raters assigning item i to category k, $r_i = \sum_k r_{ik}$, and w_{kl} is the agreement-weight matrix, then the item-level agreement is

$$A_i = \frac{1}{r_i(r_i-1)} \sum_k r_{ik} \left(\sum_l w_{kl} r_{il} - 1 \right).$$

The observed agreement is the mean of A_i over retained items. The chance term uses the average item-wise category proportions $\pi_k = m^{-1} \sum_i r_{ik}/r_i$, where m is the number of retained items.

Confidence intervals and inference.

The primary inferential path is the analytic large-sample method. For two raters, define

$$s_{ab} = w_{ab} - \frac{2(1-AC) \sum_{u,v} w_{uv}}{q(q-1)} \left(1 - \frac{\pi_a + \pi_b}{2} \right).$$

The backend variance estimator is

$$\widehat{\mathrm{Var}}(\widehat{AC}) = \frac{1}{n(1-P_e)^2} \left[\sum_{a,b} p_{ab} s_{ab}^2 - \{P_o - 2(1-AC)P_e\}^2 \right],$$

where n is the number of complete paired ratings.

For panel-level counts, write

$$AC_i = \frac{A_i - P_e}{1 - P_e},$$

and define the item-specific chance term

$$P_{e,i} = \frac{\sum_{k,l} w_{kl}}{q(q-1)} \sum_{k=1}^{q} \frac{r_{ik}}{r_i} (1 - \pi_k).$$

The asymptotic linearised contribution used by the backend is

$$\xi_i = AC_i - \frac{2(1-AC)}{1-P_e} (P_{e,i} - P_e),$$

giving

$$\widehat{\mathrm{Var}}(\widehat{AC}) = \frac{1}{m(m-1)} \sum_{i=1}^{m} (\xi_i - AC)^2,$$

where m is the number of retained items.

The reported standard error is $\widehat{\mathrm{se}}(\widehat{AC}) = \sqrt{\widehat{\mathrm{Var}}(\widehat{AC})}$, and the confidence interval is the t interval

$$\widehat{AC} \pm t_{1-\alpha/2,\nu}\, \widehat{\mathrm{se}}(\widehat{AC}),$$

truncated to $[-1, 1]$. Here $\nu = n-1$ for the two-rater path and $\nu = m-1$ for panel-level asymptotic inference. The reported test statistic is the corresponding t ratio for $H_0 : AC = 0$. For panel-level fits, `se_method = "jackknife"` remains available as a second option, using the leave-one-item-out variance

$$\widehat{\mathrm{Var}}_{\mathrm{JK}}(\widehat{AC}) = \frac{m-1}{m} \sum_{i=1}^{m} \left(\widehat{AC}_{(-i)} - \overline{AC}_{(\cdot)} \right)^2 .$$

Value

If `y` is supplied, a scalar numeric object of class `c("gwet_ac", "numeric")`. If `input = "pairwise"`, a `corr_result`. If `input` is `"ratings"` or `"counts"`, a one-row data frame with class `c("gwet_ac", "agreement_result", "data.frame")`.

Author(s)

Thiago de Paula Oliveira

References

Gwet, K. L. (2008). Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61, 29-48. doi:10.1348/000711006X126600

See Also

`cohen_kappa()` for two-rater nominal kappa; `multirater_kappa()` for panel-level nominal kappa; `weighted_kappa()` for weighted Cohen-type agreement.

Examples

```
x <- c("A", "A", "B", "B", "A", "C")
y <- c("A", "B", "B", "B", "A", "C")
gwet_ac(x, y)
gwet_ac(x, y, weights = "quadratic", levels = c("A", "B", "C"))

raters <- data.frame(
  r1 = c("A", "A", "B", "C", "A", "B"),
  r2 = c("A", "B", "B", "C", "A", "B"),
  r3 = c("A", "A", "B", "B", "A", "C"),
  stringsAsFactors = FALSE
)

fit_pw <- gwet_ac(raters)
fit_panel <- gwet_ac(raters, input = "ratings")
print(fit_pw)
print(fit_panel)
estimate(fit_pw)
```

```
tidy(fit_pw)
tidy(fit_panel)
```

`hsic` *Pairwise Hilbert-Schmidt Independence Criterion*

Description

Computes pairwise kernel dependence measures for the numeric columns of a matrix or data frame using the Hilbert-Schmidt Independence Criterion (HSIC). By default, `hsic()` returns a normalised kernel independence correlation in `[0, 1]`, analogous to a distance-correlation matrix.

Usage

```
hsic(
  data,
  kernel = c("gaussian", "linear", "laplace", "polynomial"),
  bandwidth = c("median", "silverman", "scott"),
  normalise = TRUE,
  estimator = c("biased", "unbiased"),
  na_method = c("error", "pairwise", "complete"),
  p_value = FALSE,
  B = 499L,
  seed = NULL,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

## S3 method for class 'hsic'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'hsic'
summary(
  object,
  n = NULL,
```

```
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.hsic'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'hsic'
plot(
  x,
  title = NULL,
  low_color = "white",
  high_color = "steelblue1",
  value_text_size = 4,
  show_value = TRUE,
  ...
)
```

Arguments

data	A numeric matrix or data frame with at least two numeric columns. Non-numeric columns are dropped.
kernel	Kernel used to build univariate Gram matrices. Supported options are "gaussian" (RBF), "linear", "laplace", and "polynomial".
bandwidth	Bandwidth rule for Gaussian and Laplace kernels. "median" uses the upper median of finite off-diagonal pairwise distances divided by $\sqrt{2}$, with fallback 0.001 for degenerate columns. "silverman" uses Silverman's univariate rule, and "scott" uses Scott's univariate rule. Linear and polynomial kernels ignore this argument.
normalise	Logical. If TRUE, return the normalised kernel independence correlation. If FALSE, return raw HSIC.
estimator	HSIC estimator. "biased" uses the stable V-statistic $n^{-2}\mathrm{tr}(KHLH)$. "unbiased" uses the U-statistic estimator with zero diagonal Gram matrices and requires at least four complete observations per pair.
na_method	Missing-data handling. "error" rejects missing, NaN, and infinite values; "complete"

	performs one listwise finite-row deletion; `"pairwise"` recomputes each pair on its pairwise complete rows.
`p_value`	Logical. If `TRUE`, attach permutation p-values for the HSIC independence test.
`B`	Positive integer. Number of permutations used when `p_value = TRUE`.
`seed`	Optional positive integer seed for reproducible permutation inference. The seed is passed to the C++ permutation engine and does not mutate the user's global R RNG state.
`n_threads`	Integer ≥ 1. Number of OpenMP threads. Defaults to `getOption("matrixCorr.threads", 1L)`.
`output`	Output representation: `"matrix"`, `"sparse"`, or `"edge_list"`.
`threshold`	Non-negative absolute-value filter for non-matrix outputs. Must be `0` when `output = "matrix"`.
`diag`	Logical; whether to include diagonal entries in sparse and edge-list outputs.
`x`	An object of class `hsic`.
`digits`	Integer; number of decimal places to print.
`n`	Optional row threshold for compact preview output.
`topn`	Optional number of leading/trailing rows to show when truncated.
`max_vars`	Optional maximum number of visible columns.
`width`	Optional display width.
`show_ci`	One of `"yes"` or `"no"`; accepted for compatibility with shared matrix-printing methods.
`...`	Additional arguments passed to print or plot methods.
`object`	An object of class `hsic`.
`title`	Optional plot title.
`low_color`	Colour for low HSIC values.
`high_color`	Colour for high HSIC values.
`value_text_size`	Font size for tile labels.
`show_value`	Logical; whether to overlay numeric values on the heatmap.

Details

For paired observations (x_i, y_i), HSIC measures the squared Hilbert-Schmidt norm of the RKHS cross-covariance operator. With centred Gram matrices $K_c = HKH$ and $L_c = HLH$, the biased empirical estimator is

$$\widehat{\mathrm{HSIC}}_b = \frac{1}{n^2}\mathrm{tr}(KHLH) = \frac{1}{n^2}\sum_{ij}(K_c)_{ij}(L_c)_{ij}.$$

With `normalise = TRUE`, the returned matrix contains the kernel independence correlation

$$\widehat{\mathrm{kCor}}(X,Y) = \frac{\widehat{\mathrm{HSIC}}(X,Y)}{\sqrt{\widehat{\mathrm{HSIC}}(X,X)\widehat{\mathrm{HSIC}}(Y,Y)}}.$$

The raw HSIC matrix is stored in `attr(x, "hsic_raw")`. When `estimator = "unbiased"`, raw HSIC can be slightly negative in finite samples; these signed raw values are retained in `hsic_raw`, while the displayed normalised coefficient is clipped only for the user-facing matrix.

Kernel formulas are: Gaussian $\exp(-|x_i - x_j|^2/(2\sigma^2))$, Laplace $\exp(-|x_i - x_j|/\sigma)$, linear $x_i x_j$, and polynomial $(x_i x_j + 1)^2$. For `bandwidth = "median"`, σ is the upper median of $\{|x_i - x_j| : i < j\}$ divided by $\sqrt{2}$; if this is non-finite or non-positive, $\sigma = 0.001$.

For `p_value = TRUE`, the test statistic is the raw HSIC estimate. The null distribution is generated by permuting one variable within each pair, and the p-value is $(1 + \#\{T_b \geq T_{\mathrm{obs}}\})/(B + 1)$. This is an $O(Bn^2)$ pairwise procedure and can be expensive for large matrices or large `B`.

The polynomial kernel currently uses the fixed form $(x_i x_j + 1)^2$. More polynomial controls can be added later without changing the main HSIC object contract.

Value

A matrix-like correlation result. With `normalise = TRUE`, entries are normalised kernel independence correlations and the diagonal is `1` when the variable has positive kernel self-dependence. With `normalise = FALSE`, entries are raw HSIC estimates. Attributes record the kernel, bandwidth rule, estimator, raw HSIC matrix, diagnostics, and optional permutation inference.

Author(s)

Thiago de Paula Oliveira

References

Gretton, A., Bousquet, O., Smola, A., & Schoelkopf, B. (2005). Measuring statistical dependence with Hilbert-Schmidt norms. *Algorithmic Learning Theory*.

Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schoelkopf, B., & Smola, A. J. (2007). A kernel statistical test of independence. *Advances in Neural Information Processing Systems*.

Gretton, A., Bousquet, O., Smola, A., & Schoelkopf, B. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6, 2075-2129.

Pfister, N., Buhlmann, P., Scholkopf, B., & Peters, J. (2018). Kernel-based tests for joint independence. *Journal of the Royal Statistical Society: Series B*, 80(1), 5-31.

Examples

```
set.seed(1)
x <- rnorm(200)
y <- rnorm(200)
H0 <- hsic(cbind(x = x, y = y))
H0

set.seed(2)
x <- rnorm(200)
y <- x^2 + rnorm(200, sd = 0.1)
H1 <- hsic(cbind(x = x, y = y))
H1

set.seed(3)
```

```
X <- cbind(a = rnorm(80), b = rnorm(80), c = rnorm(80))
H_perm <- hsic(X, p_value = TRUE, B = 19, seed = 1)
summary(H_perm)
estimate(H_perm)
tidy(H_perm)
```

---

icc *Intraclass Correlation for Wide Data*

Description

Computes intraclass correlation coefficients for the numeric columns of a matrix or data frame using the classical ANOVA mean-square formulas. The output can be either a pairwise matrix across columns or an overall all-column coefficient table.

Usage

```
icc(
  data,
  model = c("oneway", "twoway_random", "twoway_mixed"),
  type = c("consistency", "agreement"),
  unit = c("single", "average"),
  scope = c("pairwise", "overall"),
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE,
  ...
)

## S3 method for class 'icc'
print(
  x,
  digits = 4,
  ci_digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'icc'
summary(
```

```
  object,
  digits = 4,
  ci_digits = 2,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.icc'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'icc_overall'
print(
  x,
  digits = 4,
  ci_digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'icc_overall'
summary(
  object,
  digits = 4,
  ci_digits = 2,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

```
## S3 method for class 'summary.icc_overall'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or data frame with at least two numeric columns.
model	Character scalar selecting the classical ICC model. "oneway" uses the one-way random-effects formulation, "twoway_random" uses the two-way random-effects formulation, and "twoway_mixed" uses the two-way mixed-effects formulation.
type	Character scalar selecting the reliability target. "consistency" does not penalize systematic column mean differences in the same way as absolute agreement, whereas "agreement" targets absolute agreement and therefore incorporates those differences.
unit	Character scalar selecting whether the coefficient refers to a single measurement ("single") or to the mean of multiple measurements ("average"). For scope = "pairwise", the average-measure coefficient always uses k = 2. For scope = "overall", it uses the full number of analysed columns.
scope	Character scalar selecting the analysis target. "pairwise" returns the symmetric matrix of two-column ICCs. "overall" returns the six-row overall ANOVA table for the full set of columns. Choose "pairwise" when the question is about specific column pairs; choose "overall" when the question is about the full set of columns analysed jointly.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values before estimation. "pairwise" uses pair-specific complete cases for scope = "pairwise" and complete rows across all analysed columns for scope = "overall". "complete" performs listwise deletion once across all analysed columns before ICC estimation.
ci	Logical; if TRUE, return confidence intervals.
conf_level	Confidence level for the interval output. Ignored when ci = FALSE.
n_threads	Integer number of OpenMP threads.
verbose	Logical; if TRUE, report how many threads are requested.
...	Passed to the underlying print helper.
x	An intraclass-correlation object returned by icc(), or a summary object returned by summary() for that fit.

digits	Integer; number of digits to print.
ci_digits	Integer; number of digits for CI bounds.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
object	An intraclass-correlation object returned by icc().

Details

Each column is treated as a measurement channel, method, or rater and each row is treated as a subject.

The function supports two distinct analysis targets.

- scope = "pairwise" answers: "how reliable is each specific column pair?" Each estimate is based on exactly two columns and the output is a symmetric matrix.
- scope = "overall" answers: "how reliable is the full set of columns when analysed jointly?" The output is the standard six-form overall ANOVA table (ICC1, ICC2, ICC3, ICC1k, ICC2k, ICC3k).

These two scopes do not target the same quantity. The overall coefficients are computed from the full multi-column ANOVA decomposition and are not obtained by averaging or otherwise aggregating the pairwise matrix.

The three main choice arguments determine the classical ICC form.

- model controls the rater structure:
  - "oneway" uses the one-way random-effects formulation.
  - "twoway_random" uses the two-way random-effects formulation.
  - "twoway_mixed" uses the two-way mixed-effects formulation.
- type controls whether systematic column mean differences are penalized:
  - "consistency" targets consistency across columns.
  - "agreement" targets absolute agreement across columns.
- unit controls whether reliability refers to one measurement or to the average of multiple measurements:
  - "single" returns the single-measure coefficient.
  - "average" returns the average-measure coefficient.

The supported mappings are:

- model = "oneway", type = "consistency", unit = "single" gives ICC1.
- model = "oneway", type = "consistency", unit = "average" gives ICC1k.
- model = "twoway_random", type = "agreement", unit = "single" gives ICC2.
- model = "twoway_random", type = "agreement", unit = "average" gives ICC2k.

- model = "twoway_random", type = "consistency", unit = "single" gives ICC3.
- model = "twoway_random", type = "consistency", unit = "average" gives ICC3k.
- model = "twoway_mixed", type = "agreement", unit = "single" gives the mixed-effects absolute-agreement analogue with the same classical point formula as ICC2.
- model = "twoway_mixed", type = "agreement", unit = "average" gives the corresponding average-measure analogue.
- model = "twoway_mixed", type = "consistency", unit = "single" gives ICC3.
- model = "twoway_mixed", type = "consistency", unit = "average" gives ICC3k.

The combination model = "oneway", type = "agreement" is not defined here and returns an error.

For scope = "pairwise", the point estimates are computed in C++ directly from the two-column ANOVA mean squares for each complete pair. For unit = "average", the implementation uses k = 2 because each estimate is based on exactly two columns.

For scope = "overall", the point estimates are computed jointly from the full wide matrix using the classical ANOVA decomposition over all columns. Here the average-measure coefficients use k = ncol(data) after any row filtering required by na_method.

Missing-data handling depends on scope:

- with na_method = "error", missing values are rejected before estimation;
- with na_method = "pairwise" and scope = "pairwise", each pair uses its own complete-case overlap;
- with na_method = "pairwise" and scope = "overall", rows are restricted to complete cases across all columns because the overall ANOVA requires a common wide matrix.

When ci = TRUE, confidence intervals are obtained from the classical F-based ANOVA formulas corresponding to the selected coefficient. For scope = "pairwise", non-estimable off-diagonal pairs return NA. For scope = "overall", the coefficient table includes interval columns for all six standard rows.

Value

For scope = "pairwise", if ci = FALSE, a symmetric matrix of class icc. If ci = TRUE, a list with elements est, lwr.ci, and upr.ci and class c("icc", "icc_ci").

For scope = "overall", a list of class c("icc_overall", "icc") with a coefficient table, ANOVA table, and mean-square metadata. The coefficient table always includes the standard six overall coefficients. Confidence interval columns are attached when ci = TRUE.

All outputs carry attributes describing the selected model, type, unit, and method.

References

Shrout PE, Fleiss JL (1979). Intraclass correlations: uses in assessing rater reliability. Psychological Bulletin, 86(2), 420-428.

McGraw KO, Wong SP (1996). Forming inferences about some intraclass correlation coefficients. Psychological Methods, 1(1), 30-46.

See Also

ccc(), ccc_rm_reml(), ba(), ba_rm(), rmcorr()

Examples

```
set.seed(123)
n <- 40
subj <- rnorm(n, sd = 1)
dat <- data.frame(
  m1 = subj + rnorm(n, sd = 0.3),
  m2 = 0.2 + subj + rnorm(n, sd = 0.4),
  m3 = -0.1 + subj + rnorm(n, sd = 0.5)
)

fit_icc <- icc(dat,
  model = "twoway_random",
  type = "agreement",
  unit = "single",
  scope = "pairwise"
)
print(fit_icc)
summary(fit_icc)
estimate(fit_icc)
tidy(fit_icc)

fit_icc_overall <- icc(dat, scope = "overall", ci = TRUE)
print(fit_icc_overall)
summary(fit_icc_overall)
confint(fit_icc_overall)
```

---

icc_rm_reml *Repeated-Measures Intraclass Correlation via REML*

Description

Computes pairwise repeated-measures intraclass correlation coefficients from long-format data using the same REML and Woodbury-identity backend used by the repeated-measures agreement models.

Usage

```
icc_rm_reml(
  data,
  response,
  subject,
  method = NULL,
  time = NULL,
  type = c("consistency", "agreement"),
```

```
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  ci_mode = c("auto", "raw", "logit"),
  verbose = FALSE,
  digits = 4,
  use_message = TRUE,
  interaction = FALSE,
  max_iter = 100,
  tol = 1e-06,
  Dmat = NULL,
  Dmat_type = c("time-avg", "typical-visit", "weighted-avg", "weighted-sq"),
  Dmat_weights = NULL,
  Dmat_rescale = TRUE,
  ar = c("none", "ar1"),
  ar_rho = NA_real_,
  slope = c("none", "subject", "method", "custom"),
  slope_var = NULL,
  slope_Z = NULL,
  drop_zero_cols = TRUE,
  vc_select = c("auto", "none"),
  vc_alpha = 0.05,
  vc_test_order = c("subj_time", "subj_method"),
  include_subj_method = NULL,
  include_subj_time = NULL,
  sb_zero_tol = 1e-10
)

## S3 method for class 'icc_rm_reml'
print(
  x,
  digits = 4,
  ci_digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'icc_rm_reml'
summary(
  object,
  digits = 4,
  ci_digits = 2,
  n = NULL,
  topn = NULL,
```

```
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.icc_rm_reml'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A data frame.
response	Character. Response variable name.
subject	Character. Subject ID variable name.
method	Character or NULL. Optional column name of method factor (added to fixed effects).
time	Character or NULL. Optional column name of time factor (added to fixed effects).
type	Character scalar; one of c("consistency","agreement"). For "consistency", systematic fixed method differences are not penalized through S_B. For "agreement", the fixed-effect dispersion term S_B is added to the denominator.
ci	Logical. If TRUE, return a CI container for the repeated ICC.
conf_level	Numeric in $(0, 1)$. Confidence level when ci = TRUE (default 0.95).
n_threads	Integer ≥ 1. Number of OpenMP threads to use for computation. Defaults to getOption("matrixCorr.threads", 1L).
ci_mode	Character scalar; one of c("auto","raw","logit"). Controls how confidence intervals are computed when ci = TRUE. If "raw", a Wald CI is formed on the ICC scale and truncated to [0,1]. If "logit", a Wald CI is computed on the $\mathrm{logit}(\mathrm{ICC})$ scale and back-transformed to the original scale. If "auto" (default), the backend chooses per estimate between the raw-scale and logit-scale interval.
verbose	Logical. If TRUE, prints a structured summary of the fitted variance components and S_B for each fit. Default FALSE.
digits	Integer (≥ 0). Number of decimal places to use in the printed summary when verbose = TRUE. Default 4.
use_message	Logical. When verbose = TRUE, verbose summaries are emitted with **cli** messages.
interaction	Logical. Include method:time interaction? (default FALSE).

`max_iter` Integer. Maximum iterations for variance-component updates (default `100`).

`tol` Numeric. Convergence tolerance on parameter change (default `1e-6`).

`Dmat` Optional $n_t \times n_t$ numeric matrix to weight/aggregate time-specific fixed biases in the S_B quadratic form. If supplied, it is used (after optional mass rescaling; see `Dmat_rescale`) whenever at least two *present* time levels exist; otherwise it is ignored. **If `Dmat` is `NULL`**, a canonical kernel D_m is *constructed* from `Dmat_type` and `Dmat_weights` (see below). `Dmat` should be symmetric positive semidefinite; small asymmetries are symmetrized internally.

`Dmat_type` Character, one of `c("time-avg","typical-visit", "weighted-avg","weighted-sq")`. Only used when `Dmat = NULL`. It selects the aggregation target for time-specific fixed biases in S_B. Options are:

- `"time-avg"`: square of the time-averaged bias, $D_m = (1/n_t)\,1 1^\top$.
- `"typical-visit"`: average of squared per-time biases, $D_m = I_{n_t}$.
- `"weighted-avg"`: square of a weighted average, $D_m = n_t\, w\, w^\top$ with $\sum w = 1$.
- `"weighted-sq"`: weighted average of squared biases, $D_m = n_t\, \mathrm{diag}(w)$ with $\sum w = 1$.

Pick `"time-avg"` for ICC targeting the time-averaged measurement; pick `"typical-visit"` for ICC targeting a randomly sampled visit (typical occasion). Default `"time-avg"`.

`Dmat_weights` Optional numeric weights w used when `Dmat_type %in% c("weighted-avg","weighted-sq")`. Must be nonnegative and finite. If `names(w)` are provided, they should match the *full* time levels in `data`; they are aligned to the *present* time subset per fit. If unnamed, the length must equal the number of present time levels. In all cases w is internally normalized to sum to 1.

`Dmat_rescale` Logical. When `TRUE` (default), the supplied/built D_m is rescaled to satisfy the simple mass rule $1^\top D_m 1 = n_t$. This keeps the S_B denominator invariant and harmonizes with the time-averaging factors used for the variance terms.

`ar` Character. Residual correlation structure: `"none"` (iid) or `"ar1"` for subject-level AR(1) correlation within contiguous time runs. Default `c("none")`.

`ar_rho` Numeric in $(-0.999, 0.999)$ or `NA`. If `ar = "ar1"` and `ar_rho` is finite, it is treated as fixed. If `ar = "ar1"` and `ar_rho = NA`, ρ is estimated by profiling a 1-D objective (REML when available; an approximation otherwise). Default `NA_real_`.

`slope` Character. Optional extra random-effect design Z. With `"subject"` a single random slope is added (one column in Z); with `"method"` one column per method level is added; with `"custom"` you provide `slope_Z` directly. Default `c("none","subject","method","custom")`.

`slope_var` For `slope %in% c("subject","method")`, a character string giving the name of a column in `data` used as the slope regressor (e.g., centered time). It is looked up inside `data`; do not pass the vector itself. NAs are treated as zeros in Z.

`slope_Z` For `slope = "custom"`, a numeric matrix with n rows (same order as `data`) providing the full extra random-effect design Z. **Each column of `slope_Z` has its own variance component** $\sigma^2_{Z,j}$; columns are treated as *uncorrelated* (diagonal block in G). Ignored otherwise.

`drop_zero_cols` Logical. When `slope = "method"`, drop all-zero columns of Z after subsetting (useful in pairwise fits). Default `TRUE`.

`vc_select` Character scalar; one of `c("auto","none")`. Controls how the subject by method $\sigma^2_{A\times M}$ ("subj_method") and subject by time $\sigma^2_{A\times T}$ ("subj_time") variance components are included. If `"auto"` (default), the function performs boundary-aware REML likelihood-ratio tests (LRTs; null on the boundary at zero with a half-χ^2_1 reference) to decide whether to retain each component, in the order given by `vc_test_order`. If `"none"`, no testing is done and inclusion is taken from `include_subj_method`/`include_subj_time` (or, if `NULL`, from the mere presence of the corresponding factor in the design). In pairwise fits, the decision is made independently for each method pair.

`vc_alpha` Numeric scalar in $(0, 1)$; default `0.05`. Per-component significance level for the boundary-aware REML LRTs used when `vc_select = "auto"`. The tests are one-sided for variance components on the boundary and are not multiplicity-adjusted.

`vc_test_order` Character vector (length 2) with a permutation of `c("subj_time","subj_method")`; default `c("subj_time","subj_method")`. Specifies the order in which the two variance components are tested when `vc_select = "auto"`. The component tested first may be dropped before testing the second. If a factor is absent in the design (e.g., no time factor so "subj_time" is undefined), the corresponding test is skipped.

`include_subj_method`, `include_subj_time`
Logical scalars or `NULL`. When `vc_select = "none"`, these control whether the $\sigma^2_{A\times M}$ ("subj_method") and $\sigma^2_{A\times T}$ ("subj_time") random effects are included (`TRUE`) or excluded (`FALSE`) in the model. If `NULL` (default), inclusion defaults to the presence of the corresponding factor in the data (i.e., at least two method/time levels). When `vc_select = "auto"`, these arguments are ignored (automatic selection is used instead).

`sb_zero_tol` Non-negative numeric scalar; default `1e-10`. Numerical threshold for the fixed-effect dispersion term S_B. After computing $\widehat{S_B}$ and its delta-method variance, if $\widehat{S_B} \le$ `sb_zero_tol` or non-finite, the procedure treats S_B as fixed at zero in the delta step.

`x` For `print()`, an object returned by `icc_rm_reml()` or `summary.icc_rm_reml()`.

`ci_digits` Integer; number of digits for confidence interval bounds in printed method summaries.

`n` Optional row threshold for compact preview output.

`topn` Optional number of leading/trailing rows to show when truncated.

`max_vars` Optional maximum number of visible columns; `NULL` derives this from console width.

`width` Optional display width; defaults to `getOption("width")`.

`show_ci` One of `"yes"` or `"no"`.

`...` Passed to the underlying display helpers.

`object` For `summary()`, an object returned by `icc_rm_reml()`.

Details

The repeated-measures model is fit separately for each method pair using the same REML and Woodbury-identity backend used by the repeated-measures concordance estimator.

Kernel D_m and the fixed-bias term S_B. Let $d = L^\top \hat{\beta}$ stack the within-time, pairwise method differences, grouped by time. The symmetric positive semidefinite kernel $D_m \succeq 0$ selects which functional of the bias profile is targeted by S_B. Internally, the code rescales any supplied or constructed D_m to satisfy $1^\top D_m 1 = n_t$ for stability and comparability.

- `Dmat_type = "time-avg"` targets the square of the time-averaged bias.
- `Dmat_type = "typical-visit"` targets the average of squared per-time biases.
- `Dmat_type = "weighted-avg"` targets the square of a weighted time average.
- `Dmat_type = "weighted-sq"` targets the weighted average of squared per-time biases.

As in the repeated-measures concordance implementation, S_B is the fitted fixed-effect dispersion term induced by D_m. It enters the denominator only for `type = "agreement"`.

Time-averaging and shrinkage factors. The fitted variance components reported in the summary are σ_S^2, $\sigma_{S\times M}^2$, $\sigma_{S\times T}^2$, and σ_E^2, stored respectively as `sigma2_subject`, `sigma2_subject_method`, `sigma2_subject_time`, and `sigma2_error`.

The repeated-measures ICC always uses σ_S^2 in the numerator only. The denominator uses the same time-averaging logic already used by the repeated-measures concordance backend, through two pair-specific factors $\bar{\kappa}_g$ and $\bar{\kappa}_e$.

If `time` is absent, the implementation sets

$$\bar{\kappa}_g = 0, \qquad \bar{\kappa}_e = 1.$$

If `time` is present and the target is a single visit (`Dmat_type %in% c("typical-visit", "weighted-sq")`), the implementation leaves the time-varying terms unshrunk:

$$\bar{\kappa}_g = 1, \qquad \bar{\kappa}_e = 1.$$

If `time` is present and the target is time-averaged (`Dmat_type %in% c("time-avg", "weighted-avg")`), then for each observed subject-method unit with T distinct observed visits:

- with equal weights,

$$\kappa_g = \frac{1}{T}, \qquad \kappa_e = \frac{T + 2\sum_{h=1}^{T-1}(T-h)\rho^h}{T^2},$$

  with $\kappa_e = \kappa_g$ when residuals are iid;

- with normalized visit weights $w_1, \ldots, w_T$,

$$\kappa_g = \sum_{t=1}^{T} w_t^2, \qquad \kappa_e = \sum_{t=1}^{T}\sum_{s=1}^{T} w_t w_s \rho^{|t-s|},$$

  with $\kappa_e = \kappa_g$ when residuals are iid.

With unbalanced T, the implementation averages the per-unit κ values across the observations contributing to the pair and then clamps both $\bar{\kappa}_g$ and $\bar{\kappa}_e$ to $[10^{-12},\ 1]$ for numerical stability.

Repeated-measures ICC. Let $\sigma^2_{S\times M,\mathrm{eff}}$ denote the effective subject-by-method variance term, equal to $\sigma^2_{S\times M}$ when the subject-by-method random effect is included and 0 otherwise. Let $\sigma^2_{S\times T,\mathrm{eff}}$ denote the effective subject-by-time variance term, equal to $\sigma^2_{S\times T}$ when the subject-by-time random effect is included and 0 otherwise.

For `type = "consistency"`, the reported ICC is

$$\mathrm{ICC}_{\mathrm{consistency}} \;=\; \frac{\sigma^2_S}{\sigma^2_S + \sigma^2_{S\times M,\mathrm{eff}} + \bar{\kappa}_g\,\sigma^2_{S\times T,\mathrm{eff}} + \bar{\kappa}_e\,\sigma^2_E}.$$

For `type = "agreement"`, the denominator additionally includes S_B:

$$\mathrm{ICC}_{\mathrm{agreement}} \;=\; \frac{\sigma^2_S}{\sigma^2_S + \sigma^2_{S\times M,\mathrm{eff}} + \bar{\kappa}_g\,\sigma^2_{S\times T,\mathrm{eff}} + \bar{\kappa}_e\,\sigma^2_E + S_B}.$$

This differs from repeated-measures concordance because ICC uses only σ^2_S in the numerator, whereas the concordance numerator also includes the time-averaged subject-time term. Extra random-effect variances $\{\sigma^2_{Z,j}\}$ from `slope` / `slope_Z` are estimated by the shared backend but are not included in the ICC denominator.

CIs / SEs (delta method for ICC). Let $I_A = 1$ for `type = "agreement"` and $I_A = 0$ for `type = "consistency"`, and define

$$\theta \;=\; \left(\sigma^2_S,\ \sigma^2_{S\times M,\mathrm{eff}},\ \sigma^2_{S\times T,\mathrm{eff}},\ \sigma^2_E,\ S_B\right)^{\top}.$$

Write $\mathrm{ICC}(\theta) = N/D$ with

$$N = \sigma^2_S, \qquad D = \sigma^2_S + \sigma^2_{S\times M,\mathrm{eff}} + \bar{\kappa}_g\,\sigma^2_{S\times T,\mathrm{eff}} + \bar{\kappa}_e\,\sigma^2_E + I_A S_B.$$

The gradient used in the delta method is

$$\frac{\partial\,\mathrm{ICC}}{\partial\sigma^2_S} \;=\; \frac{\sigma^2_{S\times M,\mathrm{eff}} + \bar{\kappa}_g\,\sigma^2_{S\times T,\mathrm{eff}} + \bar{\kappa}_e\,\sigma^2_E + I_A S_B}{D^2},$$

$$\frac{\partial\,\mathrm{ICC}}{\partial\sigma^2_{S\times M,\mathrm{eff}}} \;=\; -\frac{N}{D^2}, \qquad \frac{\partial\,\mathrm{ICC}}{\partial\sigma^2_{S\times T,\mathrm{eff}}} \;=\; -\frac{\bar{\kappa}_g\,N}{D^2},$$

$$\frac{\partial\,\mathrm{ICC}}{\partial\sigma^2_E} \;=\; -\frac{\bar{\kappa}_e\,N}{D^2}, \qquad \frac{\partial\,\mathrm{ICC}}{\partial S_B} \;=\; -\frac{I_A\,N}{D^2}.$$

The covariance matrix $\widehat{\mathrm{Var}}(\hat{\theta})$ is assembled from the same REML fit:

- the $(\sigma^2_S, \sigma^2_{S\times M,\mathrm{eff}}, \sigma^2_{S\times T,\mathrm{eff}})$ block comes from the empirical subject-level covariance of the per-subject REML component updates;
- $\widehat{\mathrm{Var}}(\hat{\sigma}^2_E)$ is approximated as the variance of the weighted mean of subject-level residual quadratic forms;
- $\widehat{\mathrm{Var}}(S_B)$ uses the fixed-effect quadratic-form delta method already computed in the backend.

Cross-covariances across these blocks are ignored as a large-sample simplification, so

$$\widehat{\mathrm{se}}\{\widehat{\mathrm{ICC}}\} \;=\; \sqrt{\nabla \mathrm{ICC}(\hat{\theta})^{\top}\, \widehat{\mathrm{Var}}(\hat{\theta})\, \nabla \mathrm{ICC}(\hat{\theta})}\,.$$

If `ci_mode = "raw"`, a Wald interval is formed on the ICC scale,

$$\widehat{\mathrm{ICC}} \;\pm\; z_{1-\alpha/2}\, \widehat{\mathrm{se}}\{\widehat{\mathrm{ICC}}\},$$

and truncated to $[0, 1]$. If `ci_mode = "logit"`, the backend applies the same Wald construction after the transform $\phi = \mathrm{logit}(\mathrm{ICC})$, with

$$\widehat{\mathrm{se}}(\hat{\phi}) \;\approx\; \frac{\widehat{\mathrm{se}}\{\widehat{\mathrm{ICC}}\}}{\widehat{\mathrm{ICC}}\,(1 - \widehat{\mathrm{ICC}})},$$

and then back-transforms

$$\mathrm{logit}^{-1}\Big(\hat{\phi} \pm z_{1-\alpha/2}\, \widehat{\mathrm{se}}(\hat{\phi})\Big).$$

If `ci_mode = "auto"`, the backend selects between the raw-scale and logit-scale interval per estimate, typically preferring the logit form near the boundaries.

Choosing ρ for AR(1). When `ar="ar1"` and `ar_rho = NA`, ρ is estimated by profiling the REML log-likelihood at $(\hat{\beta}, \hat{G}, \hat{\sigma}^2_E)$. With very few visits per subject, ρ can be weakly identified; consider sensitivity checks over a plausible range.

Value

A repeated-measures pairwise ICC object. Without confidence intervals the result is a symmetric matrix of class `c("icc_rm_reml", "icc", "matrix")`. With confidence intervals it is a list with `est`, `lwr.ci`, and `upr.ci` and class `c("icc_rm_reml", "icc_ci", "icc")`. Both carry the fitted variance-component matrices as attributes.

Notes on stability and performance

All per-subject solves are $r \times r$ with $r = 1 + n_m + n_t + q_Z$, so cost scales with the number of subjects and the fixed-effects dimension rather than the total number of observations. Solvers use symmetric positive-definite paths with a small diagonal ridge and pseudo-inverse fallback, which helps for very small or unbalanced subsets and near-boundary estimates. For `AR(1)`, observations are ordered by time within subject; `NA` time codes break the run, and gaps between factor levels are treated as regular steps.

Heteroscedastic slopes across Z columns are supported. Each Z column has its own variance component $\sigma^2_{Z,j}$, but cross-covariances among Z columns are set to zero.

Threading and BLAS guards

The C++ backend uses OpenMP loops while also forcing vendor BLAS libraries to run single-threaded so that overall CPU usage stays predictable. This guard is applied to OpenBLAS, Apple's Accelerate, and Intel MKL when their runtime controls are available. You can opt out manually by setting `MATRIXCORR_DISABLE_BLAS_GUARD=1` in the environment before loading the package.

References

Shrout PE, Fleiss JL (1979). Intraclass correlations: uses in assessing rater reliability. Psychological Bulletin, 86(2), 420-428.

McGraw KO, Wong SP (1996). Forming inferences about some intraclass correlation coefficients. Psychological Methods, 1(1), 30-46.

See Also

icc(), ccc(), ccc_rm_reml(), ba(), ba_rm(), rmcorr()

Examples

```
set.seed(321)
n_id <- 20
n_time <- 3
id <- factor(rep(seq_len(n_id), each = 2 * n_time))
method <- factor(rep(rep(c("A", "B"), each = n_time), times = n_id))
time <- factor(rep(seq_len(n_time), times = 2 * n_id))

subj <- rnorm(n_id, sd = 1)[as.integer(id)]
subj_method <- rnorm(n_id * 2, sd = 0.25)
sm <- subj_method[(as.integer(id) - 1L) * 2L + as.integer(method)]
y <- subj + sm + 0.3 * (method == "B") + rnorm(length(id), sd = 0.35)

dat_rm <- data.frame(y = y, id = id, method = method, time = time)

fit_icc_rm <- icc_rm_reml(
  dat_rm,
  response = "y",
  subject = "id",
  method = "method",
  time = "time",
  type = "consistency",
  ci = TRUE
)
print(fit_icc_rm)
summary(fit_icc_rm)
confint(fit_icc_rm)
tidy(fit_icc_rm)
```

---

kendall_tau *Pairwise (or Two-Vector) Kendall's Tau Rank Correlation*

Description

Computes pairwise Kendall's tau correlations for numeric data using a high-performance 'C++' backend. Optional confidence intervals are available for matrix and data-frame input.

Usage

```
kendall_tau(
  data,
  y = NULL,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  ci_method = c("fieller", "if_el", "brown_benedetti"),
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'kendall_matrix'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'kendall_matrix'
plot(
  x,
  title = "Kendall's Tau correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

## S3 method for class 'kendall_matrix'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
```

```
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.kendall_matrix'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	For matrix/data frame mode, a numeric matrix or a data frame with at least two numeric columns. All non-numeric columns are excluded. For two-vector mode, a numeric vector x.
y	Optional numeric vector y of the same length as data when data is a vector. If supplied, the function computes the Kendall correlation *between* data *and* y using a low-overhead scalar path and returns a single number.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each correlation on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
ci	Logical (default FALSE). If TRUE, attach pairwise confidence intervals for the off-diagonal Kendall correlations in matrix/data-frame mode.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
ci_method	Confidence-interval engine used when ci = TRUE. Supported Kendall methods are "fieller" (default), "brown_benedetti", and "if_el".
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.

threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
x	An object of class summary.kendall_matrix.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
ci_digits	Integer; digits for Kendall confidence limits in the pairwise summary.
show_ci	One of "yes" or "no".
title	Plot title. Default is "Kendall's Tau correlation heatmap".
low_color	Color for the minimum tau value. Default is "indianred1".
high_color	Color for the maximum tau value. Default is "steelblue1".
mid_color	Color for zero correlation. Default is "white".
value_text_size	Font size for displaying correlation values. Default is 4.
ci_text_size	Text size for confidence intervals in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class kendall_matrix.

Details

Kendall's tau is a rank-based measure of association between two variables. For a dataset with n observations on variables X and Y, let $n_0 = n(n-1)/2$ be the number of unordered pairs, C the number of concordant pairs, and D the number of discordant pairs. Let $T_x = \sum_g t_g(t_g-1)/2$ and $T_y = \sum_h u_h(u_h-1)/2$ be the numbers of tied pairs within X and within Y, respectively, where t_g and u_h are tie-group sizes in X and Y.

The tie-robust Kendall's tau-b is:

$$\tau_b = \frac{C - D}{\sqrt{(n_0 - T_x)\,(n_0 - T_y)}}.$$

When there are no ties ($T_x = T_y = 0$), this reduces to tau-a:

$$\tau_a = \frac{C - D}{n(n-1)/2}.$$

The function automatically handles ties. In degenerate cases where a variable is constant ($n_0 = T_x$ or $n_0 = T_y$), the tau-b denominator is zero and the correlation is undefined (returned as NA off the diagonal).

When na_method = "pairwise", each (i, j) estimate is recomputed on the pairwise complete-case overlap of columns i and j. Confidence intervals use the observed pairwise-complete Kendall estimate and the same pairwise complete-case overlap.

With ci_method = "fieller", the interval is built on the Fisher-style transformed scale $z = \operatorname{atanh}(\hat{\tau})$ using Fieller's asymptotic standard error

$$\mathrm{SE}(z) = \sqrt{\frac{0.437}{n-4}},$$

where n is the pairwise complete-case sample size. The interval is then mapped back with tanh() and clipped to $[-1, 1]$ for numerical safety. This is the default Kendall CI and is intended to be the fast, production-oriented choice.

With ci_method = "brown_benedetti", the interval is computed on the Kendall tau scale using the Brown-Benedetti large-sample variance for Kendall's tau-b. This path is tie-aware, remains on the original Kendall scale, and is intended as a conventional asymptotic alternative when a direct tau-scale interval is preferred.

With ci_method = "if_el", the interval is computed in 'C++' using an influence-function empirical-likelihood construction built from the linearised Kendall estimating equation. The lower and upper limits are found by solving the empirical-likelihood ratio equation against the χ^2_1-cutoff implied by conf_level. This method is slower than "fieller" and is intended for specialised inference.

Performance:

- In the **two-vector mode** (y supplied), the C++ backend uses a raw-double path with minimal overhead.
- In the **matrix/data-frame mode**, the no-missing estimate-only path uses the Knight (1966) $O(n \log n)$ algorithm. Pairwise-complete inference paths recompute each pair on its complete-case overlap; the "brown_benedetti" interval adds tie-aware large-sample variance calculations and "if_el" adds extra per-pair likelihood solving.

Value

- If y is NULL and data is a matrix/data frame: a symmetric numeric matrix where entry (i, j) is the Kendall's tau correlation between the i-th and j-th numeric columns. When ci = TRUE, the object also carries a ci attribute with elements est, lwr.ci, upr.ci, conf.level, and ci.method. Pairwise complete-case sample sizes are stored in attr(x, "diagnostics")$n_complete.
- If y is provided (two-vector mode): a single numeric scalar, the Kendall's tau correlation between data and y.

Invisibly returns the kendall_matrix object.

A ggplot object representing the heatmap.

Note

Missing values are rejected when na_method = "error". Columns with fewer than two usable observations are excluded. Confidence intervals are not available in the two-vector interface.

Author(s)

Thiago de Paula Oliveira

References

Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, 30(1/2), 81-93.

Knight, W. R. (1966). A Computer Method for Calculating Kendall's Tau with Ungrouped Data. *Journal of the American Statistical Association*, 61(314), 436-439.

Fieller, E. C., Hartley, H. O., & Pearson, E. S. (1957). Tests for rank correlation coefficients. I. *Biometrika*, 44(3/4), 470-481.

Brown, M. B., & Benedetti, J. K. (1977). Sampling behavior of tests for correlation in two-way contingency tables. *Journal of the American Statistical Association*, 72(358), 309-315.

Huang, Z., & Qin, G. (2023). Influence function-based confidence intervals for the Kendall rank correlation coefficient. *Computational Statistics*, 38(2), 1041-1055.

Croux, C., & Dehon, C. (2010). Influence functions of the Spearman and Kendall correlation measures. *Statistical Methods & Applications*, 19, 497-515.

See Also

`print.kendall_matrix`, `plot.kendall_matrix`

Examples

```
# Basic usage with a matrix
mat <- cbind(a = rnorm(100), b = rnorm(100), c = rnorm(100))
kt <- kendall_tau(mat)
print(kt)
summary(kt)
plot(kt)

# Confidence intervals
kt_ci <- kendall_tau(mat[, 1:3], ci = TRUE)
print(kt_ci, show_ci = "yes")
summary(kt_ci)
estimate(kt_ci)
tidy(kt_ci)
ci(kt_ci)
confint(kt_ci)

# Two-vector mode (scalar path)
x <- rnorm(1000); y <- 0.5 * x + rnorm(1000)
kendall_tau(x, y)

# Including ties
tied_df <- data.frame(
  v1 = rep(1:5, each = 20),
  v2 = rep(5:1, each = 20),
  v3 = rnorm(100)
)
```

```
kt_tied <- kendall_tau(tied_df, ci = TRUE, ci_method = "fieller")
print(kt_tied, show_ci = "yes")

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(kt)
}
```

---

krippendorff_alpha *Krippendorff's Alpha Reliability Coefficient*

Description

Estimates Krippendorff's alpha as a panel-level reliability/agreement coefficient among two or more coders, raters, observers, judges, or instruments. Missing ratings are supported through `na_method`, and nominal, ordinal, interval, and ratio disagreement functions are available.

Usage

```
krippendorff_alpha(
  data,
  levels = NULL,
  input = c("ratings", "counts"),
  level = c("nominal", "ordinal", "interval", "ratio"),
  method = c("customary", "analytical"),
  na_method = c("error", "complete", "available"),
  min_raters = 2L,
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  se_method = c("auto", "bootstrap", "jackknife", "none"),
  n_boot = 1000L,
  seed = NULL,
  n_threads = getOption("matrixCorr.threads", 1L),
  return_matrices = FALSE,
  verbose = FALSE,
  ...
)

## S3 method for class 'krippendorff_alpha'
print(x, digits = 4, ...)

## S3 method for class 'krippendorff_alpha'
summary(object, digits = 4, ...)

## S3 method for class 'summary.krippendorff_alpha'
print(
```

```
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'krippendorff_alpha'
plot(
  x,
 type = c("estimate", "item_disagreement", "category_proportion", "coincidence"),
  ...
)
```

Arguments

data	Input data. For `input = "ratings"`, rows are units/items and columns are coders/raters. For `input = "counts"`, rows are units/items and columns are categories containing per-item category counts.
levels	Optional category labels in analysis order. For ordinal character data, explicit `levels` are required unless the data are ordered factors. For interval and ratio alpha, `levels` must be coercible to finite numeric values.
input	One of `"ratings"` or `"counts"`.
level	Measurement level: `"nominal"`, `"ordinal"`, `"interval"`, or `"ratio"`.
method	Estimator. `"customary"` uses the coincidence-matrix estimator of Krippendorff/Hayes. `"analytical"` uses the improved estimator and jackknife inference direction described by Hughes.
na_method	Missing-data rule. `"error"` rejects missing ratings, `"complete"` removes any item with one or more missing ratings, and `"available"` retains items with at least `min_raters` observed ratings. For `input = "counts"`, missing counts are not allowed and rows with total counts below `min_raters` are dropped.
min_raters	Minimum number of observed ratings required for a retained item. Must be at least `2`.
ci	Logical; if `TRUE`, attach confidence intervals.
p_value	Logical; if `TRUE`, attach test statistics and p-values. This is available only for `method = "analytical"` with jackknife inference.
conf_level	Confidence level for intervals. Default is `0.95`.
se_method	Standard-error method. `"auto"` resolves to `"bootstrap"` for customary alpha when `ci = TRUE`, and to `"jackknife"` for analytical alpha when `ci = TRUE` or `p_value = TRUE`. `"bootstrap"` is allowed only for customary alpha. `"jackknife"` is the primary inference path for analytical alpha. `"none"` disables uncertainty output.
n_boot	Number of bootstrap resamples used for customary alpha when `ci = TRUE`.

seed	Optional positive integer seed used for reproducible bootstrap resampling.
n_threads	Integer ≥ 1. Number of OpenMP threads used by the C++ backend.
return_matrices	Logical; if TRUE, attach the retained count matrix, observed coincidence matrix, expected coincidence matrix, and disagreement matrix in attr(x, "matrices").
verbose	Logical; if TRUE, emit short progress messages.
...	Unused.
x	A krippendorff_alpha object.
digits	Integer; number of decimal places for rounded numeric columns.
object	A krippendorff_alpha object.
n	Optional preview row threshold.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.
width	Optional display width.
show_ci	One of "yes" or "no".
type	Plot type: "estimate", "item_disagreement", "category_proportion", or "coincidence".

Details

Krippendorff's alpha is a panel-level reliability coefficient, not a pairwise correlation matrix. The customary coincidence-matrix estimator is

$$\alpha = 1 - \frac{D_o}{D_e},$$

where D_o is the observed disagreement and D_e is the expected disagreement under chance assignment.

For retained item u with m_u observed ratings and category counts n_{uc}, the observed coincidence matrix updates by

$$O_{ck} \leftarrow O_{ck} + \frac{n_{uc}(n_{uk} - 1(c = k))}{m_u - 1}.$$

Let $n_c = \sum_k O_{ck}$ and $n = \sum_c n_c$. Then

$$D_o = \frac{1}{n} \sum_{c,k} O_{ck}\, \delta^2_{ck},$$

$$D_e = \frac{1}{n(n-1)} \sum_{c,k} n_c n_k\, \delta^2_{ck},$$

where δ^2_{ck} is the level-specific disagreement matrix.

The disagreement functions implemented here are:

- nominal: $\delta^2_{ck} = 1(c \neq k)$

- ordinal: Krippendorff's cumulative-margin disagreement based on the pooled category margins
- interval: $\delta^2_{ck} = (v_c - v_k)^2$
- ratio: $\delta^2_{ck} = (v_c - v_k)^2/(v_c + v_k)^2$, with $\delta^2_{ck} = 0$ when both values are zero

Analytical estimator and inference. Hughes (2024) recommends a different point estimator based on within-item and pooled pairwise disagreement. Let a be the number of retained items, m_i the number of ratings in item i, and $N = \sum_i m_i$. Define

$$\bar{n} = \frac{N - \sum_i m_i^2/N}{a - 1}.$$

The within-item mean square is

$$MSE = \frac{1}{N} \sum_{i=1}^{a} \frac{\sum_{j<k} \delta^2(x_{ij}, x_{ik})}{m_i - 1},$$

the pooled total disagreement is

$$SST = \frac{1}{N} \sum_{r<s} \delta^2(z_r, z_s),$$

where $z_1, \ldots, z_N$ are the pooled observed ratings, and

$$SSA = SST - (N - a)\, MSE, \qquad MSA = \frac{SSA}{a - 1}.$$

With $\theta = MSA/MSE$ and $\eta = \log(\theta)$, the analytical alpha estimate is

$$\alpha = \frac{\theta - 1}{\theta + \bar{n} - 1}.$$

When analytical inference is requested, the implementation uses the delete-one-item jackknife on η. If $\eta_{(-i)}$ is the leave-one-item-out transform and a is the number of retained items, the jackknife pseudo-values are

$$p_i = a\,\hat{\eta} - (a - 1)\,\hat{\eta}_{(-i)},$$

with standard error

$$\widehat{\mathrm{se}}(\hat{\eta}) = \sqrt{\mathrm{Var}(p_1, \ldots, p_a)/a}.$$

The confidence interval is formed on the η-scale with a t_{a-1} critical value and back-transformed to alpha using the full data $\bar{n}$. The analytical p-value tests $H_0 : \alpha = 0$, equivalently $H_0 : \eta = 0$, via the t statistic

$$t = \hat{\eta}/\widehat{\mathrm{se}}(\hat{\eta}).$$

For customary alpha, percentile bootstrap intervals are available by resampling retained items with replacement and recomputing the customary estimate.

Value

A one-row data frame with class `c("krippendorff_alpha", "agreement_result", "data.frame")`. The object stores method metadata, diagnostics, and optional matrices in attributes.

Author(s)

Thiago de Paula Oliveira

References

Krippendorff, K. (2011/2013). Computing Krippendorff's alpha-reliability.

Hayes, A. F. and Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1), 77-89.

Hughes, J. (2024). Toward improved inference for Krippendorff's Alpha agreement coefficient. *Journal of Statistical Planning and Inference*, 233, 106170.

See Also

`multirater_kappa()` for nominal multi-rater kappa; `gwet_ac()` for AC1/AC2 agreement coefficients.

Examples

```
raters <- data.frame(
  r1 = c("A", "A", "B", "B", "C", "A"),
  r2 = c("A", "B", "B", "B", "C", "A"),
  r3 = c("A", "A", "B", "C", "C", "A"),
  stringsAsFactors = FALSE
)

fit <- krippendorff_alpha(raters, level = "nominal", na_method = "available")
print(fit)
summary(fit)
estimate(fit)
tidy(fit)
plot(fit)
```

---

multirater_kappa *Multi-Rater Kappa for Nominal Ratings*

Description

Estimates panel-level chance-corrected agreement among multiple raters assigning items to nominal categories.

Usage

```
multirater_kappa(
  data,
  levels = NULL,
  input = c("ratings", "counts"),
  method = c("fleiss", "randolph"),
```

```
  exact = FALSE,
  na_method = c("error", "complete", "available"),
  min_raters = 2L,
  by_category = FALSE,
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  se_method = c("asymptotic", "jackknife", "none"),
  n_threads = getOption("matrixCorr.threads", 1L),
  verbose = FALSE,
  ...
)

## S3 method for class 'multirater_kappa'
print(x, digits = 4, show_by_category = FALSE, ...)

## S3 method for class 'multirater_kappa'
summary(object, digits = 4, ...)

## S3 method for class 'summary.multirater_kappa'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'multirater_kappa'
plot(
  x,
  type = c("agreement_map", "estimate", "item_agreement", "category_proportion",
    "by_category"),
  bins = 30L,
  ...
)
```

Arguments

data	Input ratings or counts data. For input = "ratings", rows are items/subjects and columns are raters/classifiers. For input = "counts", rows are items/subjects, columns are categories, and values are counts of raters assigning that category to that item.
levels	Optional explicit category labels. For nominal multi-rater kappa, category order is not used in the estimator itself, but explicit levels are useful when unobserved

	categories should be retained in the output. If factor columns are mixed with non-factor columns, `levels` must be supplied to avoid ambiguous implicit ordering.
`input`	One of `"ratings"` or `"counts"`.
`method`	One of `"fleiss"` (fixed-marginal multi-rater kappa) or `"randolph"` (free-marginal multi-rater kappa).
`exact`	Logical; if `TRUE`, use the exact Fleiss fixed-marginal variant. This is available only for `input = "ratings"` with complete item-by-rater data and `method = "fleiss"`.
`na_method`	Missing-data rule for `input = "ratings"`. `"error"` rejects missing ratings. `"complete"` removes any item with one or more missing ratings. `"available"` retains items with at least `min_raters` observed ratings and allows the number of raters to vary by item. For `input = "counts"`, missing counts are not allowed.
`min_raters`	Minimum number of observed ratings required for an item to be retained when `na_method = "available"` or when `input = "counts"` rows are filtered by total raters. Must be at least `2`.
`by_category`	Logical; if `TRUE`, attach category-wise kappas based on target-vs-other binary collapses of the category count matrix.
`ci`	Logical; if `TRUE`, attach confidence intervals using the selected `se_method`.
`p_value`	Logical; if `TRUE`, attach z statistics and two-sided p-values using the selected `se_method`.
`conf_level`	Confidence level for intervals. Default is `0.95`.
`se_method`	Standard-error method. `"asymptotic"` uses the closed-form large-sample variance for the standard non-exact Fleiss kappa when a common number of raters is available. `"jackknife"` uses leave-one-item-out jackknife inference. `"none"` disables inferential quantities and may not be combined with `ci = TRUE` or `p_value = TRUE`. When inferential quantities are requested and `se_method` is left at its default, `multirater_kappa()` automatically falls back to `"jackknife"` if asymptotic inference is unavailable for the chosen estimator/data combination. `se_method` and `conf_level` are validated only when `ci` or `p_value` requests inferential output.
`n_threads`	Integer ≥ 1. Number of OpenMP threads used by the C++ backend.
`verbose`	Logical; if `TRUE`, emit short progress messages.
`...`	Unused.
`x`	A `multirater_kappa` object.
`digits`	Integer; number of decimal places for displayed values.
`show_by_category`	Logical; whether to print the attached category-wise kappa table when available.
`object`	A `multirater_kappa` object.
`n`	Optional preview row threshold.
`topn`	Optional number of leading/trailing rows when truncated.
`max_vars`	Optional maximum number of visible columns.

width	Optional display width.
show_ci	One of "yes" or "no".
type	Plot type: "agreement_map", "estimate", "item_agreement", "category_proportion", or "by_category".
bins	Integer number of bins retained for compatibility; currently unused by the default item-agreement profile plot.

Details

multirater_kappa() returns one panel-level agreement coefficient, not a pairwise matrix. Fleiss' kappa is the default fixed-marginal estimator:

$$\kappa = \frac{\bar{P} - P_e}{1 - P_e},$$

where $\bar{P}$ is the mean item-level agreement and $P_e = \sum_j p_j^2$ uses the pooled marginal category proportions.

Randolph's free-marginal alternative replaces the expected agreement with $P_e = 1/K$. This function is for nominal categories and does not use category ordering. Use weighted_kappa() for two-rater ordered-category agreement and cohen_kappa() for two-rater nominal agreement.

When exact = TRUE, the exact Fleiss fixed-marginal estimate requires the original item-by-rater rating matrix. In **matrixCorr**, closed-form asymptotic inference is only available for the standard non-exact Fleiss estimator with a common number of raters per item. In other settings, use se_method = "jackknife" when inferential quantities are needed.

With input = "ratings" and na_method = "available", the implementation supports unbalanced item-specific numbers of raters as a generalisation beyond the strict equal-rater Fleiss setting. The returned diagnostics record the observed per-item rater counts.

Confidence intervals and standard errors. Two inference paths are implemented.

If se_method = "jackknife", the method computes leave-one-item-out estimates $\hat{\kappa}_{(-1)}, \ldots, \hat{\kappa}_{(-m)}$ over the m retained items and forms

$$\bar{\kappa}_{(.)} = \frac{1}{m} \sum_{i=1}^{m} \hat{\kappa}_{(-i)},$$

$$\widehat{\mathrm{Var}}_{\mathrm{JK}}(\hat{\kappa}) \;=\; \frac{m-1}{m} \sum_{i=1}^{m} \left(\hat{\kappa}_{(-i)} - \bar{\kappa}_{(.)}\right)^2.$$

The standard error is $\widehat{\mathrm{se}}(\hat{\kappa}) = \sqrt{\widehat{\mathrm{Var}}_{\mathrm{JK}}(\hat{\kappa})}$ and the CI is

$$\hat{\kappa} \pm z_{1-\alpha/2}\, \widehat{\mathrm{se}}(\hat{\kappa}),$$

truncated to $[-1, 1]$.

If se_method = "asymptotic", the closed-form variance is available only for the standard non-exact Fleiss estimator with a common number of raters per item. Let m be the number of items, n the common number of raters, p_j the pooled category proportions, and define

$$S = \sum_j p_j(1 - p_j), \qquad C = \sum_j p_j(1 - p_j)(1 - 2p_j).$$

Then the variance estimator is

$$\widehat{\mathrm{Var}}(\hat{\kappa}) \;=\; \frac{2\,(S^2 - C)}{S^2\, m\, n\, (n-1)}.$$

The standard error and CI are then obtained from the same Wald form above and truncated to $[-1, 1]$. When asymptotic inference is unavailable and inferential output is requested with the default setting, `multirater_kappa()` automatically falls back to the jackknife path.

Value

A one-row data frame with class `c("multirater_kappa", "agreement_result", "data.frame")`. The object carries estimator metadata and diagnostics in attributes, and may also attach category-wise binary-collapsed results in `attr(x, "by_category")`.

Author(s)

Thiago de Paula Oliveira

References

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76, 378-382. doi:10.1037/h0031619

Randolph, J. J. (2005). Free-Marginal Multirater Kappa: An Alternative to Fleiss' Fixed-Marginal Multirater Kappa.

See Also

`cohen_kappa()` for unweighted two-rater nominal agreement; `weighted_kappa()` for two-rater ordered-category agreement.

Examples

```
raters <- data.frame(
  r1 = c("A", "A", "B", "C", "A", "B"),
  r2 = c("A", "B", "B", "C", "A", "B"),
  r3 = c("A", "A", "B", "B", "A", "C"),
  stringsAsFactors = FALSE
)

fit <- multirater_kappa(raters)
print(fit)
summary(fit)
estimate(fit)
tidy(fit)
# The default plot is an item-by-category agreement map.
# Rows are items, ordered from stronger to weaker item-level agreement.
# Columns are categories. Each tile shows how many raters assigned that
# category to that item, and darker fill means a larger share of raters
# chose that category for that item.
plot(fit)
```

pbcor *Percentage bend correlation*

Description

Computes all pairwise percentage bend correlations for the numeric columns of a matrix or data frame. Percentage bend correlation limits the influence of extreme marginal observations by bending standardised deviations into the interval $[-1, 1]$, yielding a Pearson-like measure that is robust to outliers and heavy tails.

Usage

```
pbcor(
  data,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  beta = 0.2,
  n_boot = 500L,
  seed = NULL,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

## S3 method for class 'pbcor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'pbcor'
plot(
  x,
  title = "Percentage bend correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
```

```
  show_value = TRUE,
  ...
)

## S3 method for class 'pbcor'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.pbcor'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or data frame containing numeric columns.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each estimate on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample. With "pairwise", each correlation is computed on the overlapping complete rows for the column pair.
ci	Logical (default FALSE). If TRUE, attach percentile bootstrap confidence intervals for each pairwise estimate.
p_value	Logical (default FALSE). If TRUE, attach the method-specific large-sample test statistic and two-sided p-value for each pairwise estimate.
conf_level	Confidence level used when ci = TRUE. Default 0.95.
n_threads	Integer ≥ 1. Number of OpenMP threads used for the point-estimate matrix computation. Defaults to getOption("matrixCorr.threads", 1L).

`beta` Bending constant in `[0, 0.5)` that sets the cutoff used to bend standardised deviations toward the interval $[-1, 1]$. Larger values cause more observations to be bent and increase resistance to marginal outliers. Default `0.2`. See Details.

`n_boot` Integer ≥ 1. Number of bootstrap resamples used when `ci = TRUE`. Default `500`.

`seed` Optional positive integer used to seed the bootstrap resampling when `ci = TRUE`. If `NULL`, the current random-number stream is used.

`output` Output representation for the computed estimates.

- `"matrix"` (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code.
- `"sparse"`: sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding.
- `"edge_list"`: long-form data frame with columns `row`, `col`, `value`; convenient for filtering, joins, and network-style workflows.

`threshold` Non-negative absolute-value filter for non-matrix outputs: keep entries with `abs(value) >= threshold`. Use `threshold > 0` when you want only stronger associations (typically with `output = "sparse"` or `"edge_list"`). Keep `threshold = 0` to retain all values. Must be `0` when `output = "matrix"`.

`diag` Logical; whether to include diagonal entries in `"sparse"` and `"edge_list"` outputs.

`x` An object of class `summary.pbcor`.

`digits` Integer; number of digits to print.

`n` Optional row threshold for compact preview output.

`topn` Optional number of leading/trailing rows to show when truncated.

`max_vars` Optional maximum number of visible columns; `NULL` derives this from console width.

`width` Optional display width; defaults to `getOption("width")`.

`show_ci` One of `"yes"` or `"no"`.

`...` Additional arguments passed to the underlying print or plot helper.

`title` Character; plot title.

`low_color`, `high_color`, `mid_color`
Colors used in the heatmap.

`value_text_size`
Numeric text size for overlaid cell values.

`show_value` Logical; if `TRUE` (default), overlay numeric values on the heatmap tiles.

`object` An object of class `pbcor`.

`ci_digits` Integer; digits used for confidence limits in pairwise summaries.

`p_digits` Integer; digits used for p-values in pairwise summaries.

Details

Let $X \in \mathbb{R}^{n \times p}$ be a numeric matrix with rows as observations and columns as variables. For a column $x = (x_i)_{i=1}^n$, let $m_x = \mathrm{med}(x)$ and define $\omega_\beta(x)$ as the $\lfloor (1-\beta)n \rfloor$-th order statistic of $|x_i - m_x|$. Larger values of `beta` reduce $\omega_\beta(x)$, so more observations are bent to the bounds -1 and 1.

The one-step percentage-bend location is

$$\hat{\theta}_{pb}(x) = \frac{\sum_{i:|\psi_i| \le 1} x_i + \omega_\beta(x)(i_2 - i_1)}{n - i_1 - i_2}, \qquad \psi_i = \frac{x_i - m_x}{\omega_\beta(x)},$$

where $i_1 = \sum_{i=1}^n \mathbf{1}(\psi_i < -1)$ and $i_2 = \sum_{i=1}^n \mathbf{1}(\psi_i > 1)$. The bent scores are

$$a_i = \max\left\{-1,\ \min\left(1, \frac{x_i - \hat{\theta}_{pb}(x)}{\omega_\beta(x)}\right)\right\},$$

and likewise b_i for a second column y. The percentage bend correlation for the pair (x, y) is

$$r_{pb}(x, y) = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2}\sqrt{\sum_{i=1}^n b_i^2}}.$$

In the complete-data path, the bent score vectors are computed once per column and collected into a matrix $A = [a_{\cdot 1}, \ldots, a_{\cdot p}]$, after which the correlation matrix is formed from their cross-products:

$$R_{pb} = D_A^{-1/2} A^\top A D_A^{-1/2}, \qquad D_A = \mathrm{diag}(a_{\cdot 1}^\top a_{\cdot 1}, \ldots, a_{\cdot p}^\top a_{\cdot p}).$$

If a column yields an undefined bent-score denominator, the corresponding row and column are returned as `NA`. With `na_method = "pairwise"`, each pair is recomputed on its complete-case overlap. As with pairwise Pearson correlation, this pairwise path can break positive semidefiniteness.

When `p_value = TRUE`, the method-specific test statistic for a pairwise estimate r_{pb} based on n_{ij} complete observations is

$$T_{ij} = r_{pb,ij}\sqrt{\frac{n_{ij} - 2}{1 - r_{pb,ij}^2}},$$

and the reported p-value is the two-sided Student-t tail probability with $n_{ij} - 2$ degrees of freedom. When `ci = TRUE`, the interval is a percentile bootstrap interval based on n_{boot} resamples drawn from the pairwise complete cases. If $\tilde{r}_{pb,(1)} \le \cdots \le \tilde{r}_{pb,(B)}$ denotes the sorted bootstrap sample of finite estimates with B retained resamples, the reported limits are

$$\tilde{r}_{pb,(\ell)} \quad \text{and} \quad \tilde{r}_{pb,(u)},$$

where $\ell = \lfloor (\alpha/2)B + 0.5 \rfloor$ and $u = \lfloor (1 - \alpha/2)B + 0.5 \rfloor$ for $\alpha = 1 - \mathrm{conf_level}$. Resamples that yield undefined estimates are discarded before the percentile limits are formed.

Computational complexity. In the complete-data path, forming the bent scores requires sorting within each column and the cross-product step costs $O(np^2)$ with $O(p^2)$ output storage. When `ci = TRUE`, the bootstrap cost is incurred separately for each column pair.

Value

A symmetric correlation matrix with class pbcor and attributes method = "percentage_bend_correlation", description, and package = "matrixCorr". When ci = TRUE, the returned object also carries a ci attribute with elements est, lwr.ci, upr.ci, conf.level, and ci.method, plus attr(x, "conf.level"). When p_value = TRUE, it also carries an inference attribute with elements estimate, statistic, parameter, p_value, n_obs, and alternative. When either inferential option is requested, the object also carries diagnostics$n_complete.

Author(s)

Thiago de Paula Oliveira

References

Wilcox, R. R. (1994). The percentage bend correlation coefficient. Psychometrika, 59(4), 601-616. doi:10.1007/BF02294395

See Also

wincor(), skipped_corr(), bicor()

Examples

```
set.seed(10)
X <- matrix(rnorm(150 * 4), ncol = 4)
X[sample(length(X), 8)] <- X[sample(length(X), 8)] + 10

R <- pbcor(X)
print(R, digits = 2)
summary(R)
estimate(R)
tidy(R)
plot(R)

## Bootstrap confidence intervals
R_ci <- pbcor(X, ci = TRUE, n_boot = 49, seed = 10)
ci(R_ci)
confint(R_ci)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(R)
}
```

pcorr *Partial correlation matrix (sample / ridge / OAS / graphical lasso)*

Description

Computes Gaussian partial correlations for the numeric columns of a matrix or data frame using a high-performance 'C++' backend. Covariance estimation is available via the classical sample estimator, ridge regularisation, OAS shrinkage, or graphical lasso. Optional p-values and Fisher-z confidence intervals are available for the classical sample estimator in the ordinary low-dimensional setting.

Usage

```
pcorr(
  data,
  method = c("sample", "oas", "ridge", "glasso"),
  na_method = c("error", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  return_cov_precision = FALSE,
  return_details = FALSE,
  return_p_value = FALSE,
  lambda = 0.001,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

## S3 method for class 'partial_corr'
print(
  x,
  digits = 3,
  show_method = TRUE,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'partial_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
```

```
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.partial_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'partial_corr'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  mask_diag = TRUE,
  reorder = FALSE,
  ...
)
```

Arguments

data	A numeric matrix or data frame with at least two numeric columns. Non-numeric columns are ignored.
method	Character; one of "sample", "oas", "ridge", or "glasso". Default "sample".
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample. Pairwise deletion is not supported for partial correlations.
ci	Logical (default FALSE). If TRUE, attach Fisher-z confidence intervals for the off-diagonal partial correlations. This option is available only for the classical method = "sample" estimator in the ordinary low-dimensional setting.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
return_cov_precision	Logical; if TRUE, also return the covariance (cov) and precision (precision) matrices used to form the partial correlations. Requires return_details = TRUE. Default to FALSE.

`return_details` Logical; if `TRUE`, return the rich `"partial_corr"` list containing `pcor`, diagnostics, method metadata, and any requested covariance, precision, p-value, or CI components. If `FALSE` (default), `output = "matrix"` returns the standard matrixCorr estimate matrix, consistent with other correlation methods.

`return_p_value` Logical; if `TRUE`, compute two-sided p-values for testing whether each sample partial correlation is zero. With the default matrix result, p-values are stored in the inference metadata and exposed by `tidy()`. With `return_details = TRUE`, they are also returned as the `p_value` matrix. This option is available only for `method = "sample"` and requires $n > p$. Default to `FALSE`.

`lambda` Numeric ≥ 0; regularisation strength. For `method = "ridge"`, this is the penalty added to the covariance diagonal. For `method = "glasso"`, this is the off-diagonal precision-matrix ℓ_1 penalty. Ignored otherwise. Default `1e-3`.

`output` Output representation for the computed estimates.

- `"matrix"` (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code.
- `"sparse"`: sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding.
- `"edge_list"`: long-form data frame with columns `row`, `col`, `value`; convenient for filtering, joins, and network-style workflows.

`threshold` Non-negative absolute-value filter for non-matrix outputs: keep entries with `abs(value) >= threshold`. Use `threshold > 0` when you want only stronger associations (typically with `output = "sparse"` or `"edge_list"`). Keep `threshold = 0` to retain all values. Must be `0` when `output = "matrix"`.

`diag` Logical; whether to include diagonal entries in `"sparse"` and `"edge_list"` outputs.

`x` An object of class `partial_corr`.

`digits` Integer; number of decimal places for display (default 3).

`show_method` Logical; print a one-line header with `method` (and `lambda`/`rho` if available). Default `TRUE`.

`n` Optional row threshold for compact preview output.

`topn` Optional number of leading/trailing rows to show when truncated.

`max_vars` Optional maximum number of visible columns; `NULL` derives this from console width.

`width` Optional display width; defaults to `getOption("width")`.

`show_ci` One of `"yes"` or `"no"`.

`...` Additional arguments passed to `ggplot2::theme()` or other **ggplot2** layers.

`object` An object of class `partial_corr`.

`title` Plot title. By default, constructed from the estimator in `x$method`.

`low_color` Colour for low (negative) values. Default `"indianred1"`.

`high_color` Colour for high (positive) values. Default `"steelblue1"`.

`mid_color` Colour for zero. Default `"white"`.

value_text_size
: Font size for cell labels. Default 4.

show_value
: Logical; if TRUE (default), overlay numeric values on the heatmap tiles.

mask_diag
: Logical; if TRUE, the diagonal is masked (set to NA) and not labelled. Default TRUE.

reorder
: Logical; if TRUE, variables are reordered by hierarchical clustering of $1 - |pcor|$. Default FALSE.

Details

Statistical overview. Given an $n \times p$ data matrix X (rows are observations, columns are variables), the routine estimates a *partial correlation* matrix via the precision (inverse covariance) matrix. Let μ be the vector of column means and

$$S = (X - \mathbf{1}\mu)^\top (X - \mathbf{1}\mu)$$

be the centred cross-product matrix computed without forming a centred copy of X. Two conventional covariance scalings are formed:

$$\hat{\Sigma}_{\mathrm{MLE}} = S/n, \qquad \hat{\Sigma}_{\mathrm{unb}} = S/(n-1).$$

- *Sample:* $\Sigma = \hat{\Sigma}_{\mathrm{unb}}$.
- *Ridge:* $\Sigma = \hat{\Sigma}_{\mathrm{unb}} + \lambda I_p$ with user-supplied $\lambda \geq 0$ (diagonal inflation).
- *OAS (Oracle Approximating Shrinkage):* shrink $\hat{\Sigma}_{\mathrm{MLE}}$ towards a scaled identity target $\mu_I I_p$, where $\mu_I = \mathrm{tr}(\hat{\Sigma}_{\mathrm{MLE}})/p$. The data-driven weight $\rho \in [0, 1]$ is

$$\rho = \min\left\{1, \max\left(0,\ \frac{(1 - \frac{2}{p})\,\mathrm{tr}(\hat{\Sigma}^2_{\mathrm{MLE}}) + \mathrm{tr}(\hat{\Sigma}_{\mathrm{MLE}})^2}{(n + 1 - \frac{2}{p})\left[\mathrm{tr}(\hat{\Sigma}^2_{\mathrm{MLE}}) - \frac{\mathrm{tr}(\hat{\Sigma}_{\mathrm{MLE}})^2}{p}\right]}\right)\right\},$$

  and

$$\Sigma = (1 - \rho)\,\hat{\Sigma}_{\mathrm{MLE}} + \rho\,\mu_I I_p.$$

- *Graphical lasso:* estimate a sparse precision matrix Θ by maximising

$$\log\det(\Theta) - \mathrm{tr}(\hat{\Sigma}_{\mathrm{MLE}}\Theta) - \lambda \sum_{i \neq j} |\theta_{ij}|,$$

  with $\lambda \geq 0$. The returned covariance matrix is $\Sigma = \Theta^{-1}$.

The method then ensures positive definiteness of Σ (adding a very small diagonal *jitter* only if necessary) and computes the precision matrix $\Theta = \Sigma^{-1}$. Partial correlations are obtained by standardising the off-diagonals of Θ:

$$\mathrm{pcor}_{ij} \;=\; -\frac{\theta_{ij}}{\sqrt{\theta_{ii}\,\theta_{jj}}}, \qquad \mathrm{pcor}_{ii} = 1.$$

If return_p_value = TRUE, the function also reports the classical two-sided test p-values for the sample partial correlations, using

$$t_{ij} = \mathrm{pcor}_{ij}\sqrt{\frac{n-p}{1 - \mathrm{pcor}^2_{ij}}}$$

with $n - p$ degrees of freedom. These p-values are returned only for `method = "sample"`, where they match the standard full-model partial correlation test.

When `ci = TRUE`, the function reports Fisher-z confidence intervals for the sample partial correlations. For a partial correlation $r_{xy \cdot Z}$ conditioning on c variables, the transformed statistic is $z = \operatorname{atanh}(r_{xy \cdot Z})$ with standard error

$$\mathrm{SE}(z) = \frac{1}{\sqrt{n - 3 - c}},$$

where n is the effective complete-case sample size used for the estimate. The two-sided normal-theory interval is formed on the transformed scale using `conf_level` and then mapped back with `tanh()`. In the full matrix path implemented here, each off-diagonal entry conditions on all remaining variables, so $c = p - 2$ and the classical CI requires $n > p + 1$. This inference is only supported for `method = "sample"` without positive-definiteness repair; in unsupported or numerically singular settings, CI bounds are returned as `NA` with an informative **cli** warning or the request is rejected.

Interpretation. For Gaussian data, pcor_{ij} equals the correlation between residuals from regressing variable i and variable j on all the remaining variables; equivalently, it encodes conditional dependence in a Gaussian graphical model, where $\mathrm{pcor}_{ij} = 0$ if variables i and j are conditionally independent given the others. Partial correlations are invariant to separate rescalings of each variable; in particular, multiplying Σ by any positive scalar leaves the partial correlations unchanged.

Why shrinkage/regularisation? When $p \geq n$, the sample covariance is singular and inversion is ill-posed. Ridge and OAS both yield well-conditioned Σ. Ridge adds a fixed λ on the diagonal, whereas OAS shrinks adaptively towards $\mu_I I_p$ with a weight chosen to minimise (approximately) the Frobenius risk under a Gaussian model, often improving mean-square accuracy in high dimension.

Why glasso? Glasso is useful when the goal is not just to stabilise a covariance estimate, but to recover a manageable network of direct relationships rather than a dense matrix of overall associations. In Gaussian models, zeros in the precision matrix correspond to conditional independences, so glasso can suppress indirect associations that are explained by the other variables and return a smaller, more interpretable conditional-dependence graph. This is especially practical in high-dimensional settings, where the sample covariance may be unstable or singular. Glasso yields a positive-definite precision estimate and supports edge selection, graph recovery, and downstream network analysis.

Computational notes. The implementation forms S using 'BLAS' `syrk` when available and constructs partial correlations by traversing only the upper triangle with 'OpenMP' parallelism. Positive definiteness is verified via a Cholesky factorisation; if it fails, a tiny diagonal jitter is increased geometrically up to a small cap, at which point the routine signals an error.

Value

With the default `return_details = FALSE`, a standard matrixCorr correlation result: a dense `corr_matrix`, sparse matrix, or `corr_edge_list`, depending on `output`. With `return_details = TRUE`, an object of class `"partial_corr"` (a list) with elements:

- `pcor`: $p \times p$ partial correlation matrix.
- `cov` (if requested): covariance matrix used.
- `precision` (if requested): precision matrix Θ.

- p_value (if requested): matrix of two-sided p-values for the sample partial correlations.
- ci (if requested): a list with elements est, lwr.ci, upr.ci, conf.level, and ci.method.
- diagnostics: metadata used for inference, including the effective complete-case sample size and number of conditioned variables.
- method: the estimator used ("oas", "ridge", "sample", or "glasso").
- lambda: ridge or graphical-lasso penalty (or NA_real_).
- rho: OAS shrinkage weight in $[0, 1]$ (or NA_real_).
- jitter: diagonal jitter added (if any) to ensure positive definiteness.

Invisibly returns x.

A compact summary object of class summary.partial_corr.

A ggplot object.

References

Chen, Y., Wiesel, A., & Hero, A. O. III (2011). Robust Shrinkage Estimation of High-dimensional Covariance Matrices. IEEE Transactions on Signal Processing.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. Biostatistics.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis, 88(2), 365-411.

Schafer, J., & Strimmer, K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical Applications in Genetics and Molecular Biology, 4(1), Article 32.

Examples

```
## Structured MVN with known partial correlations
set.seed(42)
p <- 12; n <- 1000

## Build a tri-diagonal precision (Omega) so the true partial correlations
## are sparse
phi <- 0.35
Omega <- diag(p)
for (j in 1:(p - 1)) {
  Omega[j, j + 1] <- Omega[j + 1, j] <- -phi
}
## Strict diagonal dominance
diag(Omega) <- 1 + 2 * abs(phi) + 0.05
Sigma <- solve(Omega)

## Upper Cholesky
L <- chol(Sigma)
Z <- matrix(rnorm(n * p), n, p)
X <- Z %*% L
colnames(X) <- sprintf("V%02d", seq_len(p))
```

```
pc <- pcorr(X)
summary(pc)
estimate(pc)
tidy(pc)

## Fisher-z confidence intervals for sample partial correlations
pc_ci <- pcorr(X[, 1:5], ci = TRUE)
ci(pc_ci)
confint(pc_ci)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(pc)
}

## True partial correlation from Omega
pcor_true <- -Omega / sqrt(diag(Omega) %o% diag(Omega))
diag(pcor_true) <- 1

## Quick visual check (first 5x5 block)
round(estimate(pc)[1:5, 1:5], 2)
round(pcor_true[1:5, 1:5], 2)

## Plot method
plot(pc)

## Graphical-lasso example
set.seed(100)
p <- 20; n <- 250
Theta_g <- diag(p)
Theta_g[cbind(1:5, 2:6)] <- -0.25
Theta_g[cbind(2:6, 1:5)] <- -0.25
Theta_g[cbind(8:11, 9:12)] <- -0.20
Theta_g[cbind(9:12, 8:11)] <- -0.20
diag(Theta_g) <- rowSums(abs(Theta_g)) + 0.2

Sigma_g <- solve(Theta_g)
X_g <- matrix(rnorm(n * p), n, p) %*% chol(Sigma_g)
colnames(X_g) <- paste0("Node", seq_len(p))

gfit_1 <- pcorr(X_g, method = "glasso", lambda = 0.02,
                return_cov_precision = TRUE, return_details = TRUE)
gfit_2 <- pcorr(X_g, method = "glasso", lambda = 0.08,
                return_cov_precision = TRUE, return_details = TRUE)

## Larger lambda gives a sparser conditional-dependence graph
edge_count <- function(M, tol = 1e-8) {
  sum(abs(M[upper.tri(M, diag = FALSE)]) > tol)
}

c(edges_lambda_002 = edge_count(gfit_1$precision),
  edges_lambda_008 = edge_count(gfit_2$precision))
```

```
## Inspect strongest estimated conditional associations
pcor_g <- gfit_1$pcor
idx <- which(upper.tri(pcor_g), arr.ind = TRUE)
ord <- order(abs(pcor_g[idx]), decreasing = TRUE)
head(data.frame(
  i = rownames(pcor_g)[idx[ord, 1]],
  j = colnames(pcor_g)[idx[ord, 2]],
  pcor = round(pcor_g[idx][ord], 2)
))

## High-dimensional case p >> n
set.seed(7)
n <- 60; p <- 120

ar_block <- function(m, rho = 0.6) rho^abs(outer(seq_len(m), seq_len(m), "-"))

## Two AR(1) blocks on the diagonal
if (requireNamespace("Matrix", quietly = TRUE)) {
  Sigma_hd <- as.matrix(Matrix::bdiag(ar_block(60, 0.6), ar_block(60, 0.6)))
} else {
  Sigma_hd <- rbind(
    cbind(ar_block(60, 0.6), matrix(0, 60, 60)),
    cbind(matrix(0, 60, 60), ar_block(60, 0.6))
  )
}

L <- chol(Sigma_hd)
X_hd <- matrix(rnorm(n * p), n, p) %*% L
colnames(X_hd) <- paste0("G", seq_len(p))

pc_oas   <-
 pcorr(X_hd, method = "oas",   return_cov_precision = TRUE,
       return_details = TRUE)
pc_ridge <-
 pcorr(X_hd, method = "ridge", lambda = 1e-2,
                     return_cov_precision = TRUE, return_details = TRUE)
pc_samp  <-
 pcorr(X_hd, method = "sample", return_cov_precision = TRUE,
       return_details = TRUE)
pc_glasso <-
 pcorr(X_hd, method = "glasso", lambda = 5e-3,
                     return_cov_precision = TRUE, return_details = TRUE)

## Show how much diagonal regularisation was used
c(oas_jitter = pc_oas$jitter,
  ridge_lambda = pc_ridge$lambda,
  sample_jitter = pc_samp$jitter,
  glasso_lambda = pc_glasso$lambda)

## Compare conditioning of the estimated covariance matrices
c(kappa_oas = kappa(pc_oas$cov),
  kappa_ridge = kappa(pc_ridge$cov),
  kappa_sample = kappa(pc_samp$cov))
```

```
## Simple conditional-dependence graph from partial correlations
pcor <- pc_oas$pcor
vals <- abs(pcor[upper.tri(pcor, diag = FALSE)])
thresh <- quantile(vals, 0.98)  # top 2%
edges  <- which(abs(pcor) > thresh & upper.tri(pcor), arr.ind = TRUE)
head(data.frame(i = colnames(pcor)[edges[,1]],
                j = colnames(pcor)[edges[,2]],
                pcor = round(pcor[edges], 2)))
```

---

pearson_corr *Pairwise Pearson correlation*

Description

Computes pairwise Pearson correlations for the numeric columns of a matrix or data frame using a high-performance 'C++' backend. Optional Fisher-z confidence intervals are available.

Usage

```
pearson_corr(
  data,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'pearson_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'pearson_corr'
plot(
```

```
  x,
  title = "Pearson correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

## S3 method for class 'pearson_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.pearson_corr'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be excluded. Each column must have at least two non-missing values.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each correlation on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
ci	Logical (default FALSE). If TRUE, attach pairwise Fisher-z confidence intervals for the off-diagonal Pearson correlations.

conf_level	Confidence level used when ci = TRUE. Default is 0.95.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
x	An object of class summary.pearson_corr.
digits	Integer; number of decimal places to print in the concordance
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
ci_digits	Integer; digits for Pearson confidence limits in the pairwise summary.
show_ci	One of "yes" or "no".
title	Plot title. Default is "Pearson correlation heatmap".
low_color	Color for the minimum correlation. Default is "indianred1".
high_color	Color for the maximum correlation. Default is "steelblue1".
mid_color	Color for zero correlation. Default is "white".
value_text_size	Font size for displaying correlation values. Default is 4.
ci_text_size	Text size for confidence intervals in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class pearson_corr.

Details

Let $X \in \mathbb{R}^{n \times p}$ be a numeric matrix with rows as observations and columns as variables, and let $\mathbf{1} \in \mathbb{R}^n$ denote the all-ones vector. Define the column means $\mu = (1/n)\,\mathbf{1}^\top X$ and the centred cross-product matrix

$$S \;=\; (X - \mathbf{1}\mu)^\top (X - \mathbf{1}\mu) \;=\; X^\top \Big(I_n - \tfrac{1}{n} \mathbf{1}\mathbf{1}^\top \Big) X \;=\; X^\top X \;-\; n\,\mu\,\mu^\top.$$

The (unbiased) sample covariance is

$$\widehat{\Sigma} \;=\; \tfrac{1}{n-1}\, S,$$

and the sample standard deviations are $s_i = \sqrt{\widehat{\Sigma}_{ii}}$. The Pearson correlation matrix is obtained by standardising $\widehat{\Sigma}$, and it is given by

$$R \;=\; D^{-1/2}\,\widehat{\Sigma}\,D^{-1/2}, \qquad D \;=\; \mathrm{diag}(\widehat{\Sigma}_{11},\ldots,\widehat{\Sigma}_{pp}),$$

equivalently, entrywise $R_{ij} = \widehat{\Sigma}_{ij}/(s_i s_j)$ for $i \neq j$ and $R_{ii} = 1$. With $1/(n-1)$ scaling, $\widehat{\Sigma}$ is unbiased for the covariance; the induced correlations are biased in finite samples.

The implementation forms $X^\top X$ via a BLAS symmetric rank-k update (SYRK) on the upper triangle, then applies the rank-1 correction $-\,n\,\mu\,\mu^\top$ to obtain S without explicitly materialising $X - \mathbf{1}\mu$. After scaling by $1/(n-1)$, triangular normalisation by $D^{-1/2}$ yields R, which is then symmetrised to remove round-off asymmetry. Tiny negative values on the covariance diagonal due to floating-point rounding are truncated to zero before taking square roots.

If a variable has zero variance ($s_i = 0$), the corresponding row and column of R are set to `NA`. When `na_method = "pairwise"`, each (i, j) correlation is recomputed on the pairwise complete-case overlap of columns i and j.

When `ci = TRUE`, Fisher-z confidence intervals are computed from the observed pairwise Pearson correlation r_{ij} and the pairwise complete-case sample size n_{ij}:

$$z_{ij} = \mathrm{atanh}(r_{ij}), \qquad \mathrm{SE}(z_{ij}) = \frac{1}{\sqrt{n_{ij} - 3}}.$$

With $z_{1-\alpha/2} = \Phi^{-1}(1 - \alpha/2)$, the confidence limits are

$$\tanh\big(z_{ij} - z_{1-\alpha/2}\,\mathrm{SE}(z_{ij})\big) \;\text{ and }\; \tanh\big(z_{ij} + z_{1-\alpha/2}\,\mathrm{SE}(z_{ij})\big).$$

Confidence intervals are reported only when $n_{ij} > 3$.

Computational complexity. The dominant cost is $O(np^2)$ flops with $O(p^2)$ memory.

Value

A symmetric numeric matrix where the `(i, j)`-th element is the Pearson correlation between the `i`-th and `j`-th numeric columns of the input. When `ci = TRUE`, the object also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, and `conf.level`. When pairwise-complete evaluation is used, pairwise sample sizes are stored in `attr(x, "diagnostics")$n_complete`.

Invisibly returns the `pearson_corr` object.

A `ggplot` object representing the heatmap.

Note

`na_method = "complete"` is useful when a common analysis sample is required across all matrix entries. For covariance- or cross-product-based correlations, it also avoids the non-positive-semidefinite matrices that can arise from pairwise deletion.

Author(s)

Thiago de Paula Oliveira

References

Pearson, K. (1895). "Notes on regression and inheritance in the case of two parents". Proceedings of the Royal Society of London, 58, 240-242.

See Also

print.pearson_corr, plot.pearson_corr

Examples

```
## MVN with AR(1) correlation
set.seed(123)
p <- 6; n <- 300; rho <- 0.5
# true correlation
Sigma <- rho^abs(outer(seq_len(p), seq_len(p), "-"))
L <- chol(Sigma)
# MVN(n, 0, Sigma)
X <- matrix(rnorm(n * p), n, p) %*% L
colnames(X) <- paste0("V", seq_len(p))

pr <- pearson_corr(X)
print(pr, digits = 2)
summary(pr)
estimate(pr)
tidy(pr)
plot(pr)

## Confidence intervals
pr_ci <- pearson_corr(X[, 1:3], ci = TRUE)
ci(pr_ci)
confint(pr_ci)

## Compare the sample estimate to the truth
Rhat <- cor(X)
# estimated
round(Rhat[1:4, 1:4], 2)
# true
round(Sigma[1:4, 1:4], 2)
off <- upper.tri(Sigma, diag = FALSE)
# MAE on off-diagonals
mean(abs(Rhat[off] - Sigma[off]))

## Larger n reduces sampling error
n2 <- 2000
X2 <- matrix(rnorm(n2 * p), n2, p) %*% L
Rhat2 <- cor(X2)
off <- upper.tri(Sigma, diag = FALSE)
## mean absolute error (MAE) of the off-diagonal correlations
mean(abs(Rhat2[off] - Sigma[off]))

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
```

```
  view_corr_shiny(pr)
}
```

`plot.corr_edge_list` *S3 Plot for Edge-List Correlation Results*

Description

S3 Plot for Edge-List Correlation Results

Usage

```
## S3 method for class 'corr_edge_list'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

`x`	An edge-list correlation result.
`title`	Optional plot title.
`low_color`	Fill color for -1.
`high_color`	Fill color for +1.
`mid_color`	Fill color for 0.
`value_text_size`	Text size for optional overlaid values.
`ci_text_size`	Text size for optional confidence-interval labels.
`show_value`	Logical; overlay values if `TRUE`.
`...`	Additional theme arguments.

`plot.corr_matrix` *S3 Plot for Dense Correlation Results*

Description

S3 Plot for Dense Correlation Results

Usage

```
## S3 method for class 'corr_matrix'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

`x`	A dense correlation result (`corr_matrix`).
`title`	Optional plot title.
`low_color`	Fill color for -1.
`high_color`	Fill color for +1.
`mid_color`	Fill color for 0.
`value_text_size`	Text size for optional overlaid values.
`ci_text_size`	Text size for optional confidence-interval labels.
`show_value`	Logical; overlay values if `TRUE`.
`...`	Additional theme arguments.

Value

A `ggplot` heatmap.

plot.corr_sparse *S3 Plot for Sparse Correlation Results*

Description

S3 Plot for Sparse Correlation Results

Usage

```
## S3 method for class 'corr_sparse'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

x	A sparse correlation result.
title	Optional plot title.
low_color	Fill color for -1.
high_color	Fill color for +1.
mid_color	Fill color for 0.
value_text_size	Text size for optional overlaid values.
ci_text_size	Text size for optional confidence-interval labels.
show_value	Logical; overlay values if TRUE.
...	Additional theme arguments.

plot.prob_agree *Plot probability-of-agreement results*

Description

Plot probability-of-agreement results

Usage

```
## S3 method for class 'prob_agree'
plot(
  x,
  threshold = NULL,
  title = "Probability of agreement",
  style = c("auto", "curve", "facet", "heatmap"),
  show_ci = NULL,
  ...
)
```

Arguments

x	An object returned by prob_agree().
threshold	Optional probability threshold shown in curve plots.
title	Optional plot title.
style	Plot style: "auto" and "curve" draw pairwise probability curves; "facet" draws one curve panel per comparison; "heatmap" shows probability by predictor value and comparison.
show_ci	Logical; for curve plots, controls whether confidence ribbons are shown. By default ribbons are shown for one comparison and suppressed for multiple comparisons.
...	Additional arguments passed to ggplot2::theme().

Value

A ggplot object.

---

polychoric *Pairwise Polychoric Correlation*

Description

Computes the polychoric correlation for either a pair of ordinal variables or all pairwise combinations of ordinal columns in a matrix/data frame.

Usage

```
polychoric(
  data,
  y = NULL,
  na_method = c("error", "pairwise"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
```

```
  correct = 0.5,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'polychoric_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'polychoric_corr'
plot(
  x,
  title = "Polychoric correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'polychoric_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.polychoric_corr'
print(
  x,
  digits = NULL,
```

```
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	An ordinal vector, matrix, or data frame. Supported columns are factors, ordered factors, logical values, or integer-like numerics. In matrix/data-frame mode, only supported ordinal columns are retained.
y	Optional second ordinal vector. When supplied, the function returns a single polychoric correlation estimate.
na_method	Character scalar controlling missing-data handling. "error" rejects missing values. "pairwise" uses pairwise complete cases.
ci	Logical (default FALSE). If TRUE, attach model-based large-sample Wald confidence intervals derived from the observed information matrix of the latent-variable likelihood.
p_value	Logical (default FALSE). If TRUE, attach model-based large-sample Wald p-values and test statistics for each estimated latent correlation.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
correct	Non-negative continuity correction added to zero-count cells. Default is 0.5.
output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to print().
x	An object of class summary.polychoric_corr.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.

`width`	Optional display width; defaults to `getOption("width")`.
`show_ci`	One of `"yes"` or `"no"`.
`title`	Plot title. Default is `"Polychoric correlation heatmap"`.
`low_color`	Color for the minimum correlation.
`high_color`	Color for the maximum correlation.
`mid_color`	Color for zero correlation.
`value_text_size`	Font size used in tile labels.
`show_value`	Logical; if `TRUE` (default), overlay numeric values on the heatmap tiles.
`object`	An object of class `polychoric_corr`.
`ci_digits`	Integer; digits for confidence limits in the pairwise summary.
`p_digits`	Integer; digits for p-values in the pairwise summary.

Details

The polychoric correlation generalises the tetrachoric model to ordered categorical variables with more than two levels. It assumes latent standard-normal variables Z_1, Z_2 with correlation ρ, and cut-points $-\infty = \alpha_0 < \alpha_1 < \cdots < \alpha_R = \infty$ and $-\infty = \beta_0 < \beta_1 < \cdots < \beta_C = \infty$ such that

$$X = r \iff \alpha_{r-1} < Z_1 \le \alpha_r, \qquad Y = c \iff \beta_{c-1} < Z_2 \le \beta_c.$$

For an observed $R \times C$ contingency table with counts n_{rc}, the thresholds are estimated from the marginal cumulative proportions:

$$\alpha_r = \Phi^{-1}\Big(\sum_{k \le r} P(X = k)\Big), \qquad \beta_c = \Phi^{-1}\Big(\sum_{k \le c} P(Y = k)\Big).$$

Holding those thresholds fixed, the log-likelihood for the latent correlation is

$$\ell(\rho) = \sum_{r=1}^{R} \sum_{c=1}^{C} n_{rc} \log \Pr\big(\alpha_{r-1} < Z_1 \le \alpha_r,\ \beta_{c-1} < Z_2 \le \beta_c \mid \rho\big),$$

and the estimator returned is the maximiser over $\rho \in (-1, 1)$. The C++ implementation performs a dense one-dimensional search followed by Brent refinement.

The argument `correct` adds a non-negative continuity correction to empty cells before marginal threshold estimation and likelihood evaluation. This avoids numerical failures for sparse tables with structurally zero cells. When `correct = 0` and zero cells are present, the corresponding fit can be boundary-driven rather than a regular interior maximum-likelihood problem. The returned object stores sparse-fit diagnostics and the thresholds used for estimation so those cases can be inspected explicitly.

Assumptions. The coefficient is appropriate when both observed ordinal variables are viewed as discretisations of jointly normal latent variables. The optional p-values and confidence intervals adopt this latent-normal interpretation and use the same likelihood that defines the polychoric estimate. These inferential quantities are therefore model-based and should not be interpreted as distribution-free summaries.

Inference. When `ci = TRUE` or `p_value = TRUE`, the function refits the pairwise polychoric model by maximum likelihood and obtains the observed information matrix numerically in C++. The reported confidence interval is a Wald interval $\hat{\rho} \pm z_{1-\alpha/2}\,\mathrm{SE}(\hat{\rho})$, and the reported p-value is from the large-sample Wald z-test for $H_0 : \rho = 0$. These inferential quantities are only computed when explicitly requested.

In matrix/data-frame mode, all pairwise polychoric correlations are computed between supported ordinal columns. Diagonal entries are 1 for non-degenerate columns and `NA` when a column has fewer than two observed levels.

Computational complexity. For p ordinal variables, the matrix path evaluates $p(p-1)/2$ bivariate likelihoods. Each pair optimises a single scalar parameter ρ, so the main cost is repeated evaluation of bivariate normal rectangle probabilities.

Value

If `y` is supplied, a numeric scalar with attributes `diagnostics` and `thresholds`. Otherwise a symmetric matrix of class `polychoric_corr` with attributes `method`, `description`, `package = "matrixCorr"`, `diagnostics`, `thresholds`, and `correct`. When `p_value = TRUE`, the returned object also carries an `inference` attribute with elements `estimate`, `statistic`, `parameter`, `p_value`, and `n_obs`. When `ci = TRUE`, it also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, `conf.level`, and `ci.method`, plus `attr(x, "conf.level")`. Scalar outputs keep the same point estimate and gain the same metadata only when inference is requested. In matrix mode, `output = "edge_list"` returns a data frame with columns `row`, `col`, `value`; `output = "sparse"` returns a symmetric sparse matrix.

Author(s)

Thiago de Paula Oliveira

References

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, 44(4), 443-460.

Examples

```
set.seed(124)
n <- 1200
Sigma <- matrix(c(
  1.00, 0.60, 0.40,
  0.60, 1.00, 0.50,
  0.40, 0.50, 1.00
), 3, 3, byrow = TRUE)

Z <- mnormt::rmnorm(n = n, mean = rep(0, 3), varcov = Sigma)
Y <- data.frame(
  y1 = ordered(cut(
    Z[, 1],
    breaks = c(-Inf, -0.7, 0.4, Inf),
    labels = c("low", "mid", "high")
  )),
```

```
  y2 = ordered(cut(
    Z[, 2],
    breaks = c(-Inf, -1.0, -0.1, 0.8, Inf),
    labels = c("1", "2", "3", "4")
  )),
  y3 = ordered(cut(
    Z[, 3],
    breaks = c(-Inf, -0.4, 0.2, 1.1, Inf),
    labels = c("A", "B", "C", "D")
  ))
)

pc <- polychoric(Y)
print(pc, digits = 3)
summary(pc)
estimate(pc)
tidy(pc)
pc_ci <- polychoric(Y, ci = TRUE)
ci(pc_ci)
confint(pc_ci)
plot(pc)
polychoric(Y, output = "edge_list", threshold = 0.3, diag = FALSE)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(pc)
}

# latent Pearson correlations used to generate the ordinal variables
round(stats::cor(Z), 2)
```

`polyserial` *Polyserial Correlation Between Continuous and Ordinal Variables*

Description

Computes polyserial correlations between continuous variables in `data` and ordinal variables in `y`. Both pairwise vector mode and rectangular matrix/data-frame mode are supported.

Usage

```
polyserial(data, y, na_method = c("error", "pairwise"), ci = FALSE, p_value = FALSE,
  conf_level = 0.95, ...)

## S3 method for class 'polyserial_corr'
print(
  x,
  digits = 4,
```

```
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'polyserial_corr'
plot(
  x,
  title = "Polyserial correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'polyserial_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.polyserial_corr'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric vector, matrix, or data frame containing continuous variables.

y	An ordinal vector, matrix, or data frame containing ordinal variables. Supported columns are factors, ordered factors, logical values, or integer-like numerics.
na_method	Character scalar controlling missing-data handling. "error" rejects missing values. "pairwise" uses pairwise complete cases.
ci	Logical (default FALSE). If TRUE, attach model-based large-sample Wald confidence intervals derived from the observed information matrix of the latent-variable likelihood.
p_value	Logical (default FALSE). If TRUE, attach model-based large-sample Wald p-values and test statistics for each estimated latent correlation.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
...	Additional arguments passed to print().
x	An object of class summary.polyserial_corr.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
title	Plot title. Default is "Polyserial correlation heatmap".
low_color	Color for the minimum correlation.
high_color	Color for the maximum correlation.
mid_color	Color for zero correlation.
value_text_size	Font size used in tile labels.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class polyserial_corr.
ci_digits	Integer; digits for confidence limits in the pairwise summary.
p_digits	Integer; digits for p-values in the pairwise summary.

Details

The polyserial correlation assumes a latent bivariate normal model between a continuous variable and an unobserved continuous propensity underlying an ordinal variable. Let $(X, Z)^\top \sim N_2(0, \Sigma)$ with $\mathrm{corr}(X, Z) = \rho$, and suppose the observed ordinal response Y is formed by cut-points $-\infty = \beta_0 < \beta_1 < \cdots < \beta_K = \infty$:

$$Y = k \iff \beta_{k-1} < Z \leq \beta_k.$$

After standardising the observed continuous variable X, the thresholds are estimated from the marginal proportions of Y. Conditional on an observed x_i, the category probability is

$$\Pr(Y_i = k \mid X_i = x_i, \rho) = \Phi\left(\frac{\beta_k - \rho x_i}{\sqrt{1-\rho^2}}\right) - \Phi\left(\frac{\beta_{k-1} - \rho x_i}{\sqrt{1-\rho^2}}\right).$$

The returned estimate maximises the log-likelihood

$$\ell(\rho) = \sum_{i=1}^{n} \log \Pr(Y_i = y_i \mid X_i = x_i, \rho)$$

over $\rho \in (-1, 1)$ via a one-dimensional Brent search in C++.

Assumptions. The coefficient is appropriate when the ordinal variable is viewed as the discretised version of a latent normal variable that is jointly normal with the observed continuous variable. The optional p-values and confidence intervals adopt this latent-normal interpretation and use the same likelihood that defines the polyserial estimate. These inferential quantities are therefore model-based and should not be interpreted as distribution-free summaries.

Inference. When `ci = TRUE` or `p_value = TRUE`, the function refits the pairwise polyserial model by maximum likelihood and obtains the observed information matrix numerically in C++. The reported confidence interval is a Wald interval $\hat{\rho} \pm z_{1-\alpha/2}\,\mathrm{SE}(\hat{\rho})$, and the reported p-value is from the large-sample Wald z-test for $H_0 : \rho = 0$. These inferential quantities are only computed when explicitly requested.

In vector mode a single estimate is returned. In matrix/data-frame mode, every numeric column of `data` is paired with every ordinal column of `y`, producing a rectangular matrix of continuous-by-ordinal polyserial correlations.

Computational complexity. If `data` has p_x continuous columns and `y` has p_y ordinal columns, the matrix path computes $p_x p_y$ separate one-parameter likelihood optimisations.

Value

If both `data` and `y` are vectors, a numeric scalar. Otherwise a numeric matrix of class `polyserial_corr` with rows corresponding to the continuous variables in `data` and columns to the ordinal variables in `y`. Matrix outputs carry attributes `method`, `description`, and `package = "matrixCorr"`. When `p_value = TRUE`, the returned object also carries an `inference` attribute with elements `estimate`, `statistic`, `parameter`, `p_value`, and `n_obs`. When `ci = TRUE`, it also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, `conf.level`, and `ci.method`, plus `attr(x, "conf.level")`. Scalar outputs keep the same point estimate and gain the same metadata only when inference is requested.

Author(s)

Thiago de Paula Oliveira

References

Olsson, U., Drasgow, F., & Dorans, N. J. (1982). The polyserial correlation coefficient. *Psychometrika*, 47(3), 337-347.

Examples

```
set.seed(125)
n <- 1000
Sigma <- matrix(c(
  1.00, 0.30, 0.55, 0.20,
  0.30, 1.00, 0.25, 0.50,
  0.55, 0.25, 1.00, 0.40,
```

```
  0.20, 0.50, 0.40, 1.00
), 4, 4, byrow = TRUE)

Z <- mnormt::rmnorm(n = n, mean = rep(0, 4), varcov = Sigma)
X <- data.frame(x1 = Z[, 1], x2 = Z[, 2])
Y <- data.frame(
  y1 = ordered(cut(
    Z[, 3],
    breaks = c(-Inf, -0.5, 0.7, Inf),
    labels = c("low", "mid", "high")
  )),
  y2 = ordered(cut(
    Z[, 4],
    breaks = c(-Inf, -1.0, 0.0, 1.0, Inf),
    labels = c("1", "2", "3", "4")
  ))
)

ps <- polyserial(X, Y)
print(ps, digits = 3)
summary(ps)
estimate(ps)
tidy(ps)
ps_ci <- polyserial(X, Y, ci = TRUE)
ci(ps_ci)
confint(ps_ci)
plot(ps)
```

`print.ccc_ci` *S3 print for legacy class* `ccc_ci`

Description

For compatibility with objects that still carry class `"ccc_ci"`.

Usage

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

Arguments

`x`	A `matrixCorr_ccc` or `matrixCorr_ccc_ci` object.
`...`	Passed to underlying printers.

print.corr_edge_list *Print Edge-List Correlation Results*

Description

Print Edge-List Correlation Results

Usage

```
## S3 method for class 'corr_edge_list'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

x	An edge-list correlation result.
digits	Number of digits for numeric values.
n	Optional preview row threshold.
topn	Optional number of head/tail rows when preview is truncated.
max_vars	Optional maximum number of visible columns in preview.
width	Optional output width.
show_ci	One of "yes" or "no".
...	Unused.

print.matrixCorr_ccc *Print method for matrixCorr CCC objects*

Description

Print method for matrixCorr CCC objects

Usage

```
## S3 method for class 'matrixCorr_ccc'
print(
  x,
  digits = 4,
  ci_digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

x	A matrixCorr_ccc or matrixCorr_ccc_ci object.
digits	Number of digits for CCC estimates.
ci_digits	Number of digits for CI bounds.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Passed to underlying printers.

---

print.matrixCorr_ccc_ci

Print method for matrixCorr CCC objects with CIs

Description

Print method for matrixCorr CCC objects with CIs

Usage

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

Arguments

x	A matrixCorr_ccc or matrixCorr_ccc_ci object.
...	Passed to underlying printers.

print.rmcorr *Methods for Pairwise rmcorr Objects*

Description

Print, summarize, and plot methods for pairwise repeated-measures correlation objects of class "rmcorr" and "summary.rmcorr".

Usage

```
## S3 method for class 'rmcorr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'rmcorr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.rmcorr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'rmcorr'
plot(
```

```
  x,
  title = NULL,
  point_alpha = 0.8,
  line_width = 0.8,
  show_legend = FALSE,
  show_value = TRUE,
  ...
)
```

Arguments

x	An object of class "rmcorr" or "summary.rmcorr".
digits	Number of significant digits to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Additional arguments passed to downstream methods. For plot.rmcorr(), these are passed to ggplot2::theme().
object	An object of class "rmcorr".
title	Optional plot title for plot.rmcorr(). Defaults to a title containing the estimated repeated-measures correlation.
point_alpha	Alpha transparency for scatterplot points.
line_width	Line width for subject-specific fitted lines.
show_legend	Logical; if TRUE, shows the subject legend in the pairwise scatterplot.
show_value	Logical; included for a consistent plotting interface. Pairwise repeated-measures plots do not overlay numeric cell values, so this argument currently has no effect.

---

print.rmcorr_matrix *Methods for rmcorr Matrix Objects*

Description

Print, summarize, and plot methods for repeated-measures correlation matrix objects of class "rmcorr_matrix" and "summary.rmcorr_matrix".

Usage

```
## S3 method for class 'rmcorr_matrix'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'rmcorr_matrix'
plot(
  x,
  title = "Repeated-measures correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'rmcorr_matrix'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.rmcorr_matrix'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

x	An object of class "rmcorr_matrix" or "summary.rmcorr_matrix".
digits	Number of significant digits to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Additional arguments passed to downstream methods.
title	Plot title for plot.rmcorr_matrix().
low_color, high_color, mid_color	Colours used for negative, positive, and midpoint values in the heatmap.
value_text_size	Size of the overlaid numeric value labels in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on heatmap tiles when the plot type supports them.
object	An object of class "rmcorr_matrix".

---

print.summary.corr_result *Print Standardized Correlation Summaries*

Description

Print Standardized Correlation Summaries

Usage

```
## S3 method for class 'summary.corr_result'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

x	A summary.corr_result object.
digits	Number of digits for numeric values.
n	Optional preview row threshold.
topn	Optional number of head/tail rows when preview is truncated.
max_vars	Optional maximum number of visible columns in preview.
width	Optional output width.
show_ci	One of "yes" or "no".
...	Unused.

Value

Invisibly returns x.

---

print.summary.latent_corr

Summary Method for Latent Correlation Matrices

Description

Prints compact summary statistics returned by summary.tetrachoric_corr(), summary.polychoric_corr(), summary.polyserial_corr(), and summary.biserial_corr().

Usage

```
## S3 method for class 'summary.latent_corr'
print(x, digits = 4, ...)
```

Arguments

x	An object of class summary.latent_corr.
digits	Integer; number of decimal places to print.
...	Unused.

Value

Invisibly returns x.

print.summary.matrixCorr

Summary Method for Correlation Matrices

Description

Prints compact summary statistics returned by matrix-style `summary()` methods in **matrixCorr**.

Usage

```
## S3 method for class 'summary.matrixCorr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

`x`	An object of class `summary.matrixCorr`.
`digits`	Integer; number of decimal places to print.
`n`	Optional row threshold for compact preview output.
`topn`	Optional number of leading/trailing rows to show when truncated.
`max_vars`	Optional maximum number of visible columns; `NULL` derives this from console width.
`width`	Optional display width; defaults to `getOption("width")`.
`show_ci`	One of `"yes"` or `"no"`.
`...`	Unused.

Value

Invisibly returns `x`.

prob_agree *Probability of Agreement for Reliability Curves*

Description

prob_agree() implements the probability of agreement following Stevens and Anderson-Cook (2017). It fits binomial reliability curves by population or generation and estimates, for each pair, the probability that the fitted reliabilities differ by no more than a user-specified practically negligible amount.

This is not a correlation coefficient and should not be interpreted as linear association. It is a tolerance-based agreement measure. It is computed from the sampling distribution of the estimated difference between two reliability curves.

Usage

```
prob_agree(
  data,
  response,
  predictor,
  group,
  delta = NULL,
  limits = NULL,
  link = c("logit", "probit"),
  newdata = NULL,
  grid_size = 100L,
  ci = TRUE,
  conf_level = 0.95,
  max_iter = 50L,
  tol = 1e-08,
  verbose = FALSE
)
```

Arguments

data	A data frame containing the response, predictor, and group variables.
response	Character scalar naming the binary pass/fail response column.
predictor	Character scalar naming the numeric age, time, or operating condition column.
group	Character scalar naming the population/generation column. All pairwise comparisons among the observed non-missing levels are evaluated.
delta	Positive scalar or vector tolerance for symmetric limits -delta and delta. A length-two vector with a negative lower bound and a positive upper bound, for example c(-0.03, 0.05), is treated as an asymmetric tolerance interval. No default is provided because the tolerance is part of the scientific estimand.
limits	Numeric vector c(lower, upper) for asymmetric tolerance limits. Mutually exclusive with delta.

link	Link used for the reliability curve: "logit" or "probit".
newdata	Optional data frame containing predictor values where the probability of agreement is evaluated. If NULL, a regular grid over the observed predictor range is used.
grid_size	Integer number of evaluation points when newdata = NULL.
ci	Logical; if TRUE, attach pointwise confidence limits.
conf_level	Confidence level for intervals.
max_iter	Maximum number of IRLS iterations for each fitted curve.
tol	Convergence tolerance for IRLS coefficient updates.
verbose	Logical; if TRUE, emits a short cli message.

Details

For each pair of fitted reliability curves $\hat{\pi}_1(a)$ and $\hat{\pi}_2(a)$, Stevens and Anderson-Cook define

$$\theta(a) = P\{|\tilde{\pi}_1(a) - \tilde{\pi}_2(a)| \leq \delta(a) \mid a\}.$$

The models are fit with a binomial GLM using either a logit or probit link, with $g(\pi_i) = \beta_0 + \beta_1 a_i$. The C++ backend evaluates the two-population large-sample normal approximation described in the paper and Supplementary Material A; when more than two groups are supplied, prob_agree() applies that calculation to all pairwise group comparisons.

Missing rows in response, predictor, or group are removed before model fitting. The response must be binary, coded as 0/1, FALSE/TRUE, or a two-level factor.

Value

A data frame with class c("prob_agree_curve", "prob_agree", "data.frame") and columns group1, group2, the predictor, prob_agree, and optional lwr.ci, upr.ci.

Author(s)

Thiago de Paula Oliveira

References

Stevens, N. T. and Anderson-Cook, C. M. (2017). Comparing the Reliability of Related Populations With the Probability of Agreement. *Technometrics*, 59(3), 371-380. doi:10.1080/00401706.2016.1214180.

Examples

```
# Stevens and Anderson-Cook's probability of agreement evaluates whether
# two related populations have reliability curves that are similar enough
# to be treated as practically homogeneous. At each age, the agreement
# hypothesis is that the difference between the two fitted reliabilities
# lies inside the user-specified practical tolerance interval.
set.seed(1)
n <- 160
dat <- data.frame(
```

```
  age = c(runif(n, 0, 60), runif(n, 0, 45)),
  generation = rep(c("Gen1", "Gen2"), each = n)
)
eta <- ifelse(
  dat$generation == "Gen1",
  4.3 - 0.045 * dat$age,
  4.0 - 0.040 * dat$age
)
dat$pass <- rbinom(nrow(dat), size = 1, prob = plogis(eta))

fit_pa <- prob_agree(
  dat,
  response = "pass",
  predictor = "age",
  group = "generation",
  delta = 0.05,
  link = "logit",
  ci = TRUE
)
print(fit_pa)
summary(fit_pa)
estimate(fit_pa)
tidy(fit_pa)
confint(fit_pa)
plot(fit_pa)

# Four generations are compared as all pairwise two-generation contrasts.
set.seed(2)
n4 <- 120
dat4 <- data.frame(
  age = rep(runif(n4, 0, 55), times = 4),
  generation = rep(paste0("Gen", 1:4), each = n4)
)
shifts <- c(4.2, 4.0, 3.8, 3.6)
slopes <- c(-0.040, -0.042, -0.044, -0.046)
gen_id <- match(dat4$generation, paste0("Gen", 1:4))
eta4 <- shifts[gen_id] + slopes[gen_id] * dat4$age
dat4$pass <- rbinom(nrow(dat4), size = 1, prob = plogis(eta4))

fit4 <- prob_agree(
  dat4,
  response = "pass",
  predictor = "age",
  group = "generation",
  limits = c(-0.03, 0.05),
  link = "logit",
  ci = FALSE
)
print(fit4)
plot(fit4)
```

rmcorr *Repeated-Measures Correlation (rmcorr)*

Description

Computes repeated-measures correlation for two or more continuous responses observed repeatedly within subjects. Supply a `data.frame` plus column names, or pass the response matrix/data frame and subject vector directly. The repeated observations are indexed only by `subject`, thus no explicit time variable is modeled, and the method targets a common within-subject linear association after removing subject-specific means.

Usage

```
rmcorr(
  data = NULL,
  response,
  subject,
  na_method = c("error", "pairwise", "complete"),
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  keep_data = FALSE,
  verbose = FALSE,
  estimator = c("ancova", "weighted"),
  ci_method = c("auto", "fisher_z", "bootstrap", "none"),
  n_boot = 999L,
  seed = NULL,
  ...
)

rmcorr_weighted(data = NULL, response, subject, ...)
```

Arguments

`data`	Optional `data.frame`/matrix containing the repeated-measures dataset.
`response`	Either: • a character vector of at least two column names in `data`, or • a numeric matrix/data frame with at least two columns. If exactly two responses are supplied, the function returns a pairwise repeated-measures correlation object of class `"rmcorr"`. If three or more responses are supplied, a symmetric matrix of class `"rmcorr_matrix"` is returned.
`subject`	Subject identifier (factor/character/integer/numeric) or a single character string naming the subject column in `data`.
`na_method`	Character scalar controlling missing-data handling. `"error"` rejects missing values in the selected response columns or `subject`. `"pairwise"` uses pairwise complete cases and drops subjects with fewer than two complete pairs for the

	specific contrast. `"complete"` removes rows with missing or non-finite values in `subject` and all selected response variables before fitting.
`conf_level`	Confidence level used for Wald confidence intervals on the repeated-measures correlation (default `0.95`).
`n_threads`	Integer ≥ 1. Number of OpenMP threads used by the C++ backend in matrix mode. Defaults to `getOption("matrixCorr.threads", 1L)`.
`keep_data`	Logical (default `FALSE`). If `TRUE`, returned objects retain a compact copy of the numeric response matrix plus encoded subject identifiers. For pairwise fits this enables `plot()` to reconstruct subject-level fitted lines lazily; for matrix outputs it also allows `view_rmcorr_shiny()` to rebuild pairwise scatterplots from the returned object alone.
`verbose`	Logical; if `TRUE`, prints a short note about the thread count used in matrix mode.
`estimator`	Character scalar selecting the repeated-measures correlation estimator. `"ancova"` (default) is the classical subject-centred repeated-measures correlation. `"weighted"` is the weighted repeated-measures correlation coefficient of Kondo et al. (2025), designed for incomplete or unbalanced repeated-measurement schedules.
`ci_method`	Confidence-interval engine. `"auto"` (default) resolves to `"fisher_z"` for `estimator = "ancova"` and to subject-level `"bootstrap"` for `estimator = "weighted"` when `n_boot > 0`; otherwise `"none"`. `"none"` returns `NA` bounds.
`n_boot`	Integer ≥ 0. Number of subject-level bootstrap resamples when `ci_method = "bootstrap"`.
`seed`	Optional positive integer seed used for bootstrap reproducibility.
`...`	Deprecated compatibility aliases. The legacy `check_na` argument is still accepted here temporarily.

Details

Repeated-measures correlation estimates the common *within-subject* linear association between two variables measured repeatedly on the same subjects. It differs from agreement methods such as Lin's CCC or Bland-Altman analysis because those target concordance or interchangeability, whereas repeated-measures correlation targets the strength of the subject-centred association.

For subject $i = 1, \ldots, S$ and repeated observations $j = 1, \ldots, n_i$, let x_{ij} and y_{ij} denote the two responses. Define subject-specific means

$$\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}, \qquad \bar{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}.$$

The repeated-measures correlation uses within-subject centred values

$$x^*_{ij} = x_{ij} - \bar{x}_i, \qquad y^*_{ij} = y_{ij} - \bar{y}_i$$

and computes

$$r_{\mathrm{rm}} = \frac{\sum_i \sum_j x^*_{ij} y^*_{ij}}{\sqrt{\left(\sum_i \sum_j {x^*_{ij}}^2\right)\left(\sum_i \sum_j {y^*_{ij}}^2\right)}}.$$

Equivalently, this is the correlation implied by an ANCOVA model with a common slope and subject-specific intercepts:

$$y_{ij} = \alpha_i + \beta x_{ij} + \varepsilon_{ij}.$$

The returned slope is

$$\hat{\beta} = \frac{\sum_i \sum_j x^*_{ij} y^*_{ij}}{\sum_i \sum_j {x^*_{ij}}^2},$$

and the subject-specific fitted intercepts are $\hat{\alpha}_i = \bar{y}_i - \hat{\beta}\bar{x}_i$. Residual degrees of freedom are $N - S - 1$, where $N = \sum_i n_i$ after filtering to complete observations and retaining only subjects with at least two repeated pairs.

Confidence intervals are computed with a Fisher z-transformation of r_{rm} and then back-transformed to the correlation scale. In matrix mode, the same estimator is applied to every pair of selected response columns.

With `estimator = "weighted"`, the package implements the weighted repeated-measures estimator from Kondo et al. (2025) using complete observed (x, y) pairs per contrast. This estimator does not impute missing data; it uses per-subject complete-pair sets and weighted sums of explained and residual quantities from the fixed-subject ANCOVA decomposition.

Bootstrap confidence intervals use non-parametric subject-level resampling (subjects are resampled as blocks, not rows).

Value

Either a `"rmcorr"` object (exactly two responses) or a `"rmcorr_matrix"` object (pairwise results when $\geq$3 responses).

If `"rmcorr"` (exactly two responses), outputs include:

- `estimate`; repeated-measures correlation estimate.
- `p_value`; two-sided p-value for the common within-subject slope.
- `lwr`, `upr`; confidence interval limits for `estimate`.
- `slope`; common within-subject slope.
- `df`; residual degrees of freedom $N - S - 1$.
- `n_obs`; number of complete observations retained after dropping subjects with fewer than two repeated pairs.
- `n_subjects`; number of contributing subjects.
- `responses`; names of the fitted response variables.
- compatibility aliases `r`, `conf_int`, and `based.on` are reconstructed on access without duplicate storage.
- when `keep_data = TRUE`, compact source data are retained so `plot()` can lazily reconstruct `data_long`, `intercepts`, and fitted lines; these are otherwise not stored.

If `"rmcorr_matrix"` ($\geq$3 responses), outputs are:

- a symmetric numeric matrix of pairwise repeated-measures correlations.
- attributes `method`, `description`, and `package = "matrixCorr"`.
- `diagnostics`; a list with square matrices for `slope`, `p_value`, `df`, `n_complete`, `n_subjects`, `conf_low`, and `conf_high`, plus scalar `conf_level`.

Author(s)

Thiago de Paula Oliveira

References

Bakdash, J. Z., & Marusich, L. R. (2017). Repeated Measures Correlation. *Frontiers in Psychology*, 8, 456. doi:10.3389/fpsyg.2017.00456

Kondo, M., Nagashima, K., Isono, S., & Sato, Y. (2025). Weighted Repeated Measures Correlation Coefficient: A New Correlation Coefficient for Handling Missing Data With Repeated Measures. *Statistics in Medicine*, 44(10-12), e70046. doi:10.1002/sim.70046

Examples

```
set.seed(2026)
n_subjects <- 20
n_rep <- 4
subject <- rep(seq_len(n_subjects), each = n_rep)
subj_eff_x <- rnorm(n_subjects, sd = 1.5)
subj_eff_y <- rnorm(n_subjects, sd = 2.0)
within_signal <- rnorm(n_subjects * n_rep)

dat <- data.frame(
  subject = subject,
  x = subj_eff_x[subject] + within_signal + rnorm(n_subjects * n_rep, sd = 0.2),
  y = subj_eff_y[subject] + 0.8 * within_signal + rnorm(n_subjects * n_rep, sd = 0.3),
  z = subj_eff_y[subject] - 0.4 * within_signal + rnorm(n_subjects * n_rep, sd = 0.4)
)

fit_xy <- rmcorr(dat, response = c("x", "y"), subject = "subject", keep_data = TRUE)
print(fit_xy)
summary(fit_xy)
estimate(fit_xy)
tidy(fit_xy)
ci(fit_xy)
confint(fit_xy)
plot(fit_xy)

fit_mat <- rmcorr(dat, response = c("x", "y", "z"), subject = "subject")
print(fit_mat, digits = 3)
summary(fit_mat)
tidy(fit_mat)
plot(fit_mat)

if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  fit_mat_view <- rmcorr(
    dat,
    response = c("x", "y", "z"),
    subject = "subject",
    keep_data = TRUE
  )
  view_rmcorr_shiny(fit_mat_view)
```

```
}
```

`robust_ccc` *MCD based robust concordance correlation coefficient*

Description

Computes pairwise robust concordance correlation coefficients from the numeric columns of a matrix or data frame. The estimator replaces the classical means, variances, and covariance in Lin's concordance coefficient with joint minimum covariance determinant estimates.

Classical concordance can be strongly distorted by outliers because it depends directly on means, variances, and covariance. The minimum covariance determinant is a high breakdown estimator of multivariate location and scatter. It is designed primarily for data with an approximately elliptically symmetric and unimodal core distribution. It is not universally robust or distribution free.

Usage

```
robust_ccc(
  data,
  alpha = 0.75,
  ci = FALSE,
  conf_level = 0.95,
  n_boot = 500L,
  seed = NULL,
  na_method = c("error", "complete", "pairwise"),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  mcd_nsamp = 500L
)
```

Arguments

`data`	A numeric matrix or data frame containing at least two numeric columns. Non numeric columns in a data frame are ignored.
`alpha`	Numeric scalar in $[0.5, 1]$. This controls the fraction of observations used by the minimum covariance determinant estimator. Approximately `alpha * n` observations define the high quality subset. Lower values provide greater resistance to contamination. The approximate trimmed fraction is `1 - alpha`. Default `0.75`.
`ci`	Logical. If `TRUE`, attach paired nonparametric percentile bootstrap confidence intervals. Default `FALSE`.
`conf_level`	Confidence level used when `ci = TRUE`. Default `0.95`.
`n_boot`	Positive integer giving the number of bootstrap resamples used when `ci = TRUE`. Default `500`.

seed	Optional positive integer controlling both the stochastic FastMCD point estimates and bootstrap resampling. Stable offsets are used for different variable pairs. If NULL, the current R random number stream is used.
na_method	Character scalar controlling missing data handling. "error" rejects missing, NaN, and infinite values. "complete" removes rows that are incomplete in any retained numeric column before all estimates are calculated. "pairwise" selects finite rows separately for each variable pair.
output	Output representation. One of "matrix", "sparse", or "edge_list".
threshold	Nonnegative absolute value filter for sparse and edge list outputs. It must be zero for matrix output.
diag	Logical. Whether sparse and edge list outputs include diagonal entries.
mcd_nsamp	Positive integer giving the number of subsets used for initial estimates in the FastMCD search. Passed to the nsamp argument of robustbase::covMcd(). Default 500L. This is a search count, not the subset size controlled by alpha or the number of bootstrap resamples controlled by n_boot. Only numeric counts up to .Machine$integer.max are supported, not the character modes of covMcd().

Details

For each paired sample, let $\hat{\mu}_{X,MCD}$ and $\hat{\mu}_{Y,MCD}$ be the components of the joint MCD location estimate. Let $\hat{\sigma}^2_{X,MCD}$, $\hat{\sigma}^2_{Y,MCD}$, and $\hat{\sigma}_{XY,MCD}$ be entries of the joint MCD scatter estimate. The robust concordance estimate is

$$\hat{\rho}_R = \frac{2\hat{\sigma}_{XY,MCD}}{\hat{\sigma}^2_{X,MCD} + \hat{\sigma}^2_{Y,MCD} + (\hat{\mu}_{X,MCD} - \hat{\mu}_{Y,MCD})^2}.$$

The joint location and scatter are fitted with robustbase::covMcd() using raw.only = FALSE and nsamp = mcd_nsamp. Consequently, this function uses the final reweighted MCD estimates rather than the raw MCD subset estimates. No population moment rescaling is applied.

The default mcd_nsamp = 500L retains the published software convention. Larger counts increase the search effort and generally increase computation time. Changing the count can change the numerical MCD solution, and hence the coefficient and confidence interval, even with the same seed. The same count is used for every pair and every bootstrap refit. With a nonnull seed, repeated calls are reproducible for fixed data, arguments, and seed.

Reweighting is part of this estimator and is not a user option. Even when alpha = 1, the final covMcd() location and scatter can still be reweighted and must not be replaced by ordinary sample moments.

At least three finite paired observations are required for basic computability. An MCD fit can still be undefined for a very small or geometrically degenerate sample. If either robust marginal scatter is effectively zero, the corresponding coefficient is undefined and NA is returned. A diagonal entry is one only when its robust scatter is estimable and nondegenerate.

Confidence intervals are a matrixCorr inference extension. They are not an interval proposed by Bulut, Zobu, and Sağlam. Each bootstrap replicate samples paired rows with replacement and recalculates the complete MCD estimator. Failed or nonfinite replicate estimates are omitted before the percentile limits are calculated. The diagnostics attribute contains n_boot_success when

intervals are requested. Interval calculation is expensive because every variable pair requires `n_boot` additional MCD fits. Its leading cost is $O(p^2 B \, \mathrm{cost}(\text{bivariate MCD}))$.

This estimator retains the form and interpretation of Lin's concordance coefficient. It differs from the robust concordance variants of King and Chinchilli, which use alternative distance or loss functions. It also differs from the Bayesian robust concordance estimator of Feng, Baumgartner, and Svetnik, which uses heavy tailed multivariate modelling.

Value

A standard matrixCorr correlation result. Dense output has class `c("corr_matrix", "robust_ccc", "corr_result", "matrix")`. Confidence intervals are stored in the standard `ci` attribute with `ci.method = "percentile_bootstrap"`. The result records `alpha = alpha`, `mcd_nsamp = mcd_nsamp`, and `mcd_estimate = "reweighted"`.

Author(s)

Thiago de Paula Oliveira

References

Bulut H, Zobu M, Sağlam V (2026). A High-Breakdown MCD-Based Robust Concordance Correlation Coefficient. Mathematics 14(1), 196. doi:10.3390/math14010196

Hubert M, Debruyne M, Rousseeuw PJ (2018). Minimum covariance determinant and extensions. WIREs Computational Statistics 10, e1421. doi:10.1002/wics.1421

Rousseeuw PJ, Van Driessen K (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. Technometrics 41, 212 to 223. doi:10.1080/00401706.1999.10485670

King TS, Chinchilli VM (2001). Robust estimators of the concordance correlation coefficient. Journal of Biopharmaceutical Statistics 11, 83 to 105. doi:10.1081/BIP100107651

Feng D, Baumgartner R, Svetnik V (2015). A Robust Bayesian Estimate of the Concordance Correlation Coefficient. Journal of Biopharmaceutical Statistics 25, 490 to 507. doi:10.1080/10543406.2014.920342

Crawford SB and colleagues (2007). Computer programs for the concordance correlation coefficient. Computer Methods and Programs in Biomedicine 88, 62 to 74. doi:10.1016/j.cmpb.2007.07.003

See Also

`robustbase::covMcd()`, `ccc()`, `pbcor()`, `wincor()`

Examples

```
# Three measurement methods with an approximately normal core
set.seed(42)
x <- rnorm(40)
measurements <- cbind(
  reference = x,
  method_a = x + rnorm(40, sd = 0.2),
  method_b = 0.4 + 1.2 * x + rnorm(40, sd = 0.3)
)
```

```
fit <- robust_ccc(measurements, seed = 42)
print(fit)
summary(fit)
estimate(fit)
coef(fit)
tidy(fit)
plot(fit)

# Compare classical and robust CCC before and after discordant contamination
contaminated <- measurements
contaminated[1, "reference"] <- 9
contaminated[1, "method_a"] <- -8
fit_contaminated <- robust_ccc(contaminated, seed = 42)
rbind(
  clean = c(
    classical = estimate(ccc(measurements))[1, 2],
    robust = estimate(fit)[1, 2]
  ),
  contaminated = c(
    classical = estimate(ccc(contaminated))[1, 2],
    robust = estimate(fit_contaminated)[1, 2]
  )
)

# Alpha controls the retained subset fraction, not the search count
fit_half <- robust_ccc(contaminated, alpha = 0.5, seed = 42)
estimate(fit_half)

# Increase the initial subset search count while retaining reweighted MCD
fit_more <- robust_ccc(contaminated, mcd_nsamp = 1000L, seed = 42)
attr(fit_more, "mcd_nsamp")
attr(fit_more, "mcd_estimate")

# Global complete rows versus the finite overlap for each pair
incomplete <- measurements
incomplete[1, "reference"] <- NA
incomplete[2, "method_a"] <- Inf
fit_complete <- robust_ccc(incomplete, na_method = "complete", seed = 42)
fit_pairwise <- robust_ccc(incomplete, na_method = "pairwise", seed = 42)
estimate(fit_complete)
estimate(fit_pairwise)

# Thresholded outputs retain entries with absolute CCC at least 0.9
# Every MCD pair is still fitted before output conversion
fit_sparse <- robust_ccc(
  measurements, seed = 42, output = "sparse", threshold = 0.9, diag = FALSE
)
fit_edges <- robust_ccc(
  measurements, seed = 42, output = "edge_list", threshold = 0.9, diag = FALSE
)
fit_sparse
tidy(fit_edges)
```

```
# Paired percentile bootstrap intervals require many additional MCD fits

# A small count keeps this illustration short; the default is 500 resamples
fit_ci <- robust_ccc(
  measurements, ci = TRUE, conf_level = 0.95, n_boot = 100L, seed = 42
)
summary(fit_ci)
tidy(fit_ci)
ci(fit_ci)
confint(fit_ci)
plot(fit_ci)
attr(fit_ci, "diagnostics")$n_complete
# Off diagonal entries count finite bootstrap estimates
# Valid diagonal intervals are fixed at [1, 1] without refitting
attr(fit_ci, "diagnostics")$n_boot_success


# Interactive viewing requires both suggested packages
if (interactive() &&
    requireNamespace("shiny", quietly = TRUE) &&
    requireNamespace("shinyWidgets", quietly = TRUE)) {
  view_corr_shiny(fit)
}
```

`robust_dcor` *Robust Distance Correlation*

Description

Computes robust distance correlations by applying the biloop transformation to each numeric variable and then computing a U-centred robust distance-correlation statistic on the transformed variables.

Usage

```
robust_dcor(
  data,
  na_method = c("error", "pairwise", "complete"),
  p_value = FALSE,
  inference = c("none", "permutation"),
  n_perm = 999L,
  seed = NULL,
  c_const = 4,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
```

```
)

## S3 method for class 'robust_dcor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'robust_dcor'
summary(object, topn = NULL, show_ci = NULL, ...)

## S3 method for class 'robust_dcor'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns are dropped. Columns must be numeric.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each association on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
p_value	Logical (default FALSE). If TRUE, attach permutation p-values for pairwise independence.
inference	Character scalar. Use "none" (default) for estimates only or "permutation" to request permutation p-values. If inference = "permutation" is supplied with p_value = FALSE, p-values are inferred and returned.
n_perm	Positive integer; number of permutations used when p_value = TRUE. Default 999.

seed	Optional positive integer seed for permutation inference.
c_const	Positive numeric tuning constant for the biloop transformation. Default 4.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates: "matrix", "sparse", or "edge_list".
threshold	Non-negative absolute-value filter for non-matrix outputs. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Deprecated compatibility aliases. Currently only check_na is supported.
x, object	An object of class robust_dcor.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.
width	Optional display width.
show_ci	One of "yes" or "no".
title	Plot title.
low_color, high_color, mid_color	Colors used in the heatmap.
value_text_size, ci_text_size	Text sizes for plot labels.
show_value	Logical; overlay numeric values if TRUE.

Details

Permutation inference is computed for every unique column pair, so the workload grows as $p(p-1)n_{\mathrm{perm}}/2$. A warning is emitted for large requests; reduce n_perm, reduce the number of columns, or run estimates without permutation p-values when this cost is not intended.

For each variable $x = (x_1, \ldots, x_n)$, the wrapper first robustly standardises values using the median

$$m_x = \mathrm{median}(x)$$

and raw median absolute deviation

$$s_x = \mathrm{median}\{|x_i - m_x|\}.$$

If this scale is zero or non-finite, the implementation falls back to $\mathrm{IQR}(x)/1.34898$ and then the ordinary sample standard deviation. Columns with no positive finite fallback scale are treated as degenerate.

The standardised value $z_i = (x_i - m_x)/s_x$ is mapped to two bounded coordinates by the biloop transformation

$$\theta_i = 2\pi \tanh(z_i/c),$$

$$u_i = \begin{cases} c\{1-\cos(\theta_i)\}, & z_i > 0, \\ -c\{1-\cos(\theta_i)\}, & z_i < 0, \\ 0, & z_i = 0, \end{cases}$$

and

$$v_i = \sin(\theta_i).$$

Pairwise distances are Euclidean distances in this transformed two-dimensional space,

$$D_{ab}^{(j)} = \left[(u_{aj}-u_{bj})^2 + (v_{aj}-v_{bj})^2\right]^{1/2}, \quad a \neq b,$$

with zero diagonal.

The transformed distance matrix is U-centred as

$$A_{ab}^{(j)} = D_{ab}^{(j)} - \frac{r_a^{(j)} + r_b^{(j)}}{n-2} + \frac{S^{(j)}}{(n-1)(n-2)}, \quad a \neq b,$$

where $r_a^{(j)} = \sum_{b \neq a} D_{ab}^{(j)}$ and $S^{(j)} = \sum_{a \neq b} D_{ab}^{(j)}$. The diagonal of $A^{(j)}$ is zero.

For variables j and k, the U-centred robust squared distance covariance estimator is

$$\widehat{\mathrm{dCov}}_u^{\,2}(j,k) = \frac{2}{n(n-3)} \sum_{a<b} A_{ab}^{(j)} A_{ab}^{(k)}.$$

The corresponding robust statistic is a non-negative clipped ratio based on this covariance estimator and the two transformed distance variances.

Value

A symmetric correlation result. Dense output inherits from `robust_dcor`, `corr_matrix`, and `matrix`. Sparse and edge-list outputs use the package-standard `corr_sparse` and `corr_edge_list` representations.

Relationship to `dcor()`

`robust_dcor()` is not a replacement for `dcor()`. It estimates distance correlation after a bounded robust transformation. Large differences between `dcor()` and `robust_dcor()` can indicate that the classical dependence signal is driven by tail observations or outliers.

Note

`robust_dcor()` is more robust to extreme observations than classical `dcor()`, but it may down-weight genuine tail dependence. Classical `dcor()` may be preferable when tail dependence is the scientific target. Comparing both methods is recommended.

Author(s)

Thiago de Paula Oliveira

References

Leyder, J., Raymaekers, J., & Rousseeuw, P. J. (2025). Robust distance correlation through bounded transformations.

See Also

dcor(), bcdcor(), wincor(), pbcor(), skipped_corr()

Examples

```
## Non-linear dependence: both estimators detect association.
set.seed(1)
n <- 200
x <- rnorm(n)
y <- x^2 + rnorm(n, sd = 0.2)

X <- cbind(x = x, y = y)

classical <- dcor(X)
robust <- robust_dcor(X)
round(c(
  dcor = classical["x", "y"],
  robust_dcor = robust["x", "y"]
), 3)

## One diagonal outlier can inflate classical dCor more than robust dCor.
set.seed(45)
x <- rnorm(20)
y <- rnorm(20)
x[1] <- 10
y[1] <- 10
X_out <- cbind(x = x, y = y)

classical <- dcor(X_out)
robust <- robust_dcor(X_out)
round(c(
  dcor = classical["x", "y"],
  robust_dcor = robust["x", "y"]
), 3)

print(classical)
print(robust)
summary(robust)
estimate(robust)
tidy(robust)
plot(robust)

## Several variables.
set.seed(7)
z <- rnorm(120)
X_multi <- cbind(
  linear = z + rnorm(120, sd = 0.3),
  nonlinear = z^2 + rnorm(120, sd = 0.3),
  noise = rnorm(120),
  outlier = rnorm(120)
)
X_multi[1, "outlier"] <- 12
```

```
X_multi[1, "noise"] <- 12

robust_multi <- robust_dcor(X_multi)

print(robust_multi)
summary(robust_multi)
plot(robust_multi)
```

---

shrinkage_corr *Shrinkage Correlation*

Description

Computes a shrinkage correlation matrix for numeric data using a high-performance 'C++' backend. The current implementation uses the Schafer-Strimmer shrinkage estimator to stabilise Pearson correlation estimates by shrinking off-diagonal entries towards zero.

Usage

```
shrinkage_corr(
  data,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

schafer_corr(
  data,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

## S3 method for class 'shrinkage_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

```
## S3 method for class 'schafer_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'shrinkage_corr'
plot(
  x,
  title = "Schafer-Strimmer shrinkage correlation",
  cluster = TRUE,
  hclust_method = "complete",
  triangle = c("upper", "lower", "full"),
  show_value = TRUE,
  show_values = NULL,
  value_text_limit = 60,
  value_text_size = 3,
  palette = c("diverging", "viridis"),
  ...
)

## S3 method for class 'schafer_corr'
plot(
  x,
  title = "Schafer-Strimmer shrinkage correlation",
  cluster = TRUE,
  hclust_method = "complete",
  triangle = c("upper", "lower", "full"),
  show_value = TRUE,
  show_values = NULL,
  value_text_limit = 60,
  value_text_size = 3,
  palette = c("diverging", "viridis"),
  ...
)

## S3 method for class 'shrinkage_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
```

```
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'schafer_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be excluded. Columns must be numeric and contain no NAs.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to `getOption("matrixCorr.threads", 1L)`.
output	Output representation for the computed estimates. • `"matrix"` (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • `"sparse"`: sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • `"edge_list"`: long-form data frame with columns `row`, `col`, `value`; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with `abs(value) >= threshold`. Use `threshold > 0` when you want only stronger associations (typically with `output = "sparse"` or `"edge_list"`). Keep `threshold = 0` to retain all values. Must be `0` when `output = "matrix"`.
diag	Logical; whether to include diagonal entries in `"sparse"` and `"edge_list"` outputs.
x	An object of class `shrinkage_corr` or `schafer_corr`.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; `NULL` derives this from console width.
width	Optional display width; defaults to `getOption("width")`.
show_ci	One of `"yes"` or `"no"`.
...	Additional arguments passed to `ggplot2::theme()`.

title	Plot title.
cluster	Logical; if TRUE, reorder rows/cols by hierarchical clustering on distance $1 - r$.
hclust_method	Linkage method for hclust; default "complete".
triangle	One of "full", "upper", "lower". Default to upper.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles (subject to value_text_limit).
show_values	Deprecated compatibility alias for show_value. If supplied, it overrides show_value.
value_text_limit	Integer threshold controlling when values are drawn.
value_text_size	Font size for values if shown.
palette	Character; "diverging" (default) or "viridis".
object	An object of class shrinkage_corr or schafer_corr.

Details

Let R be the sample Pearson correlation matrix. The Schafer-Strimmer shrinkage estimator targets the identity in correlation space and uses $\hat{\lambda} = \frac{\sum_{i<j} \widehat{\mathrm{Var}}(r_{ij})}{\sum_{i<j} r_{ij}^2}$ (clamped to $[0, 1]$), where $\widehat{\mathrm{Var}}(r_{ij}) \approx \frac{(1-r_{ij}^2)^2}{n-1}$. The returned estimator is $R_{\mathrm{shr}} = (1 - \hat{\lambda})R + \hat{\lambda}I$.

Value

A symmetric numeric matrix of class shrinkage_corr (with compatibility class schafer_corr) where entry (i, j) is the shrunk correlation between the i-th and j-th numeric columns. Attributes:

- method = "schafer_shrinkage"
- description = "Schafer-Strimmer shrinkage correlation matrix"
- package = "matrixCorr"

Columns with zero variance are set to NA across row/column (including the diagonal), matching pearson_corr() behaviour.

Invisibly returns x.

A ggplot object.

Note

No missing values are permitted. Columns with fewer than two observations or zero variance are flagged as NA (row/column).

Author(s)

Thiago de Paula Oliveira

References

Schafer, J. & Strimmer, K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1).

See Also

print.shrinkage_corr, plot.shrinkage_corr, pearson_corr

Examples

```
## Multivariate normal with AR(1) dependence (Toeplitz correlation)
set.seed(1)
n <- 80; p <- 40; rho <- 0.6
d <- abs(outer(seq_len(p), seq_len(p), "-"))
Sigma <- rho^d

X <- MASS::mvrnorm(n, mu = rep(0, p), Sigma = Sigma)
colnames(X) <- paste0("V", seq_len(p))

Rshr <- shrinkage_corr(X)
print(Rshr, digits = 2, n = 6, max_vars = 6)
summary(Rshr)
estimate(Rshr)
tidy(Rshr)
plot(Rshr)

## Shrinkage typically moves the sample correlation closer to the truth
Rraw <- stats::cor(X)
off  <- upper.tri(Sigma, diag = FALSE)
mae_raw <- mean(abs(Rraw[off] - Sigma[off]))
mae_shr <- mean(abs(Rshr[off] - Sigma[off]))
print(c(MAE_raw = mae_raw, MAE_shrunk = mae_shr))
plot(Rshr, title = "Schafer-Strimmer shrinkage correlation")

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(Rshr)
}
```

---

skipped_corr *Pairwise skipped correlation*

Description

Computes all pairwise skipped correlation coefficients for the numeric columns of a matrix or data frame using a high-performance 'C++' backend.

Skipped correlation detects bivariate outliers using a projection rule and then computes Pearson or Spearman correlation on the retained observations. It is designed for situations where marginally robust methods can still be distorted by unusual points in the joint data cloud.

Usage

```
skipped_corr(
  data,
  method = c("pearson", "spearman"),
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  return_masks = FALSE,
  stand = TRUE,
  outlier_rule = c("idealf", "mad"),
  cutoff = sqrt(stats::qchisq(0.975, df = 2)),
  n_boot = 2000L,
  p_adjust = c("none", "hochberg", "ecp"),
  fwe_level = 0.05,
  n_mc = 1000L,
  seed = NULL,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
)

skipped_corr_masks(x, var1 = NULL, var2 = NULL)

## S3 method for class 'skipped_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 4,
  show_ci = NULL,
  show_p = c("auto", "yes", "no"),
  ...
)

## S3 method for class 'skipped_corr'
plot(
  x,
  title = "Skipped correlation heatmap",
  low_color = "indianred1",
```

```
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

## S3 method for class 'skipped_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.skipped_corr'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be excluded.
method	Correlation computed after removing projected outliers. One of "pearson" (default) or "spearman".
na_method	Character scalar controlling missing-data handling. With "error", the function requires all retained numeric columns to be free of missing or non-finite values and aborts otherwise. This is the recommended setting when you want a single common sample size across all pairs, reproducible skipped-row diagnostics on the same rows, or bootstrap inference via ci = TRUE / p_value = TRUE. "pairwise" recomputes each estimate on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
ci	Logical; if TRUE, attach percentile-bootstrap confidence intervals for each skipped correlation using the Wilcox (2015) B2 resampling scheme. Default FALSE.

p_value — Logical; if TRUE, attach bootstrap p-values for testing whether each skipped correlation is zero. Default FALSE.

conf_level — Confidence level used when ci = TRUE. Default 0.95.

n_threads — Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).

return_masks — Logical; if TRUE, attach compact pairwise skipped-row indices as an attribute. Default FALSE.

stand — Logical; if TRUE (default), each variable in the pair is centred by its median and divided by a robust scale estimate before the projection outlier search. The scale estimate is the MAD when positive, with fallback to $\mathrm{IQR}/1.34898$ and then the usual sample standard deviation if needed. This standardisation affects only outlier detection, not the final correlation computed on the retained observations.

outlier_rule — One of "idealf" (default) or "mad". The default uses the ideal-fourths interquartile width of projected distances; "mad" uses the median absolute deviation of projected distances.

cutoff — Positive numeric constant multiplying the projected spread in the outlier rule $\mathrm{med}(d_{i\cdot}) + cutoff \times s(d_{i\cdot})$. Larger values flag fewer observations as outliers; smaller values flag more. Default sqrt(qchisq(0.975, df = 2)).

n_boot — Integer ≥ 2. Number of bootstrap resamples used when ci = TRUE or p_value = TRUE. Default 2000.

p_adjust — One of "none" (default), "hochberg", or "ecp". Optional familywise-error procedure applied to the matrix of bootstrap p-values. "hochberg" corresponds to method H in Wilcox, Rousselet, and Pernet (2018); "ecp" corresponds to their simulated critical-p-value method ECP.

fwe_level — Familywise-error level used when p_adjust = "hochberg" or "ecp". Default 0.05.

n_mc — Integer ≥ 10. Number of null Monte Carlo data sets used when p_adjust = "ecp" to estimate the critical p-value. Default 1000.

seed — Optional positive integer used to seed the bootstrap resampling when ci = TRUE or p_value = TRUE. If NULL, a fresh internal seed is generated.

output — Output representation for the computed estimates.

- "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code.
- "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding.
- "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.

threshold — Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".

diag — Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.

x	An object of class summary.skipped_corr.
var1, var2	Optional column names or 1-based column indices used by skipped_corr_masks() to extract the skipped-row indices for one pair.
digits	Integer; number of digits to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
ci_digits	Integer; digits for skipped-correlation confidence limits.
show_ci	One of "yes" or "no".
show_p	One of "auto", "yes", "no". For print(), "auto" keeps the compact matrix-only display; use "yes" to also print pairwise p-values.
...	Additional arguments passed to the underlying print or plot helper.
title	Character; plot title.
low_color, high_color, mid_color	Colors used in the heatmap.
value_text_size	Numeric text size for overlaid cell values.
ci_text_size	Text size for confidence intervals in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class skipped_corr.

Details

Let $X \in \mathbb{R}^{n\times p}$ be a numeric matrix with rows as observations and columns as variables. For a given pair of columns (x, y), write the observed bivariate points as $u_i = (x_i, y_i)^\top$, $i = 1, \ldots, n$. If stand = TRUE, each margin is first centred by its median and divided by a robust scale estimate before outlier detection; otherwise the original pair is used. The robust scale is the MAD when positive, with fallback to $\mathrm{IQR}/1.34898$ and then the usual sample standard deviation if needed. Let $\tilde{u}_i$ denote the resulting points and let c be the componentwise median center of the detection cloud.

For each observation i, define the direction vector $b_i = \tilde{u}_i - c$. When $\|b_i\| > 0$, all observations are projected onto the line through c in direction b_i. The projected distances are

$$d_{ij} \;=\; \frac{|(\tilde{u}_j - c)^\top b_i|}{\|b_i\|}, \qquad j = 1, \ldots, n.$$

For each direction i, observation j is flagged as an outlier if

$$d_{ij} \;>\; \mathrm{med}(d_{i\cdot}) + g\, s(d_{i\cdot}), \qquad g = \texttt{cutoff},$$

where $s(\cdot)$ is either the ideal-fourths interquartile width (outlier_rule = "idealf") or the median absolute deviation (outlier_rule = "mad"). An observation is removed if it is flagged for at

least one projection direction. The skipped correlation is then the ordinary Pearson or Spearman correlation computed from the retained observations:

$$r_{\text{skip}}(x, y) \;=\; \text{cor}(x_{\mathcal{K}}, y_{\mathcal{K}})\,,$$

where $\mathcal{K}$ is the index set of observations not flagged as outliers.

Unlike marginally robust methods such as `pbcor()`, `wincor()`, or `bicor()`, skipped correlation is explicitly pairwise because outlier detection depends on the joint geometry of each variable pair. As a result, the reported matrix need not be positive semidefinite, even with complete data.

Computational notes. In the complete-data path, each column pair requires a full bivariate projection search, so the dominant cost is higher than for marginal robust methods. The implementation evaluates pairs in 'C++'; where available, pairs are processed with 'OpenMP' parallelism. With `na_method = "pairwise"`, each pair is recomputed on its overlap of non-missing rows. With `na_method = "complete"`, rows with any non-finite value across the retained numeric columns are removed before any pairwise outlier search. This gives a common row universe for all skipped-correlation masks.

Bootstrap inference. When `ci = TRUE` or `p_value = TRUE`, the implementation uses the percentile-bootstrap strategy studied by Wilcox (2015). Each bootstrap replicate resamples whole observation pairs with replacement, reruns the skipped-correlation outlier detection on the resampled data, and recomputes the skipped correlation on the retained observations. This corresponds to Wilcox's B2 method and avoids the statistically unsatisfactory shortcut of removing outliers only once before bootstrapping. Bootstrap inference currently requires complete data (`na_method = "error"` or `"complete"`). When `p_adjust = "hochberg"`, the bootstrap p-values are processed with Hochberg's step-up procedure (method H in Wilcox, Rousselet, and Pernet, 2018). When `p_adjust = "ecp"`, the package follows their ECP method and simulates `n_mc` null data sets from a p-variate normal distribution with identity covariance, recomputes the pairwise bootstrap p-values for each null data set, stores the minimum p-value from each run, and estimates the `fwe_level` quantile of that null distribution using the Harrell-Davis estimator. Hypotheses are then rejected when their observed bootstrap p-values are less than or equal to the estimated critical p-value. The calibrated H1 procedure from Wilcox, Rousselet, and Pernet (2018) is not currently implemented.

Value

A symmetric correlation matrix with class `skipped_corr` and attributes `method = "skipped_correlation"`, `description`, and `package = "matrixCorr"`. When `return_masks = TRUE`, the matrix also carries a `skipped_masks` attribute containing compact pairwise skipped-row indices. The `diagnostics` attribute stores per-pair complete-case counts and skipped-row counts/proportions. When `ci = TRUE` or `p_value = TRUE`, bootstrap inference matrices are attached via attributes.

Author(s)

Thiago de Paula Oliveira

References

Wilcox, R. R. (2004). Inferences based on a skipped correlation coefficient. Journal of Applied Statistics, 31(2), 131-143. doi:10.1080/0266476032000148821

Wilcox, R. R. (2015). Inferences about the skipped correlation coefficient: Dealing with heteroscedasticity and non-normality. Journal of Modern Applied Statistical Methods, 14(1), 172-188. doi:10.22237/jmasm/1430453580

Wilcox, R. R., Rousselet, G. A., & Pernet, C. R. (2018). Improved methods for making inferences about multiple skipped correlations. Journal of Statistical Computation and Simulation, 88(16), 3116-3131. doi:10.1080/00949655.2018.1501051

See Also

pbcor(), wincor(), bicor()

Examples

```
set.seed(12)
X <- matrix(rnorm(160 * 4), ncol = 4)
X[1, 1] <- 9
X[1, 2] <- -8

R <- skipped_corr(X, method = "pearson")
print(R, digits = 2)
summary(R)
plot(R)

Rm <- skipped_corr(X, method = "pearson", return_masks = TRUE)
skipped_corr_masks(Rm, 1, 2)

# Example 1:
Xm <- as.matrix(datasets::mtcars[, c("mpg", "disp", "hp", "wt")])
Rm2 <- skipped_corr(Xm, method = "spearman")
print(Rm2, digits = 2)

# Example 2:
Ri <- skipped_corr(Xm, method = "pearson", ci = TRUE, n_boot = 40, seed = 1)
ci(Ri)
confint(Ri)
tidy(Ri)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(R)
}
```

---

spearman_rho *Pairwise Spearman's rank correlation*

Description

Computes pairwise Spearman's rank correlations for the numeric columns of a matrix or data frame using a high-performance 'C++' backend. Optional confidence intervals are available via a jackknife Euclidean-likelihood method.

Usage

```
spearman_rho(
  data,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'spearman_rho'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'spearman_rho'
plot(
  x,
  title = "Spearman's rank correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  ci_text_size = 3,
  show_value = TRUE,
  ...
)

## S3 method for class 'spearman_rho'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
```

```
  ...
)

## S3 method for class 'summary.spearman_rho'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be excluded. Each column must have at least two non-missing values.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each correlation on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
ci	Logical (default FALSE). If TRUE, attach jackknife Euclidean-likelihood confidence intervals for the off-diagonal Spearman correlations.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
x	An object of class summary.spearman_rho.

digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
ci_digits	Integer; digits for Spearman confidence limits in the pairwise summary.
show_ci	One of "yes" or "no".
title	Plot title. Default is "Spearman's rank correlation heatmap".
low_color	Color for the minimum rho value. Default is "indianred1".
high_color	Color for the maximum rho value. Default is "steelblue1".
mid_color	Color for zero correlation. Default is "white".
value_text_size	Font size for displaying correlation values. Default is 4.
ci_text_size	Text size for confidence intervals in the heatmap.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class spearman_rho.

Details

For each column $j = 1, \ldots, p$, let $R_{\cdot j} \in \{1, \ldots, n\}^n$ denote the (mid-)ranks of $X_{\cdot j}$, assigning average ranks to ties. The mean rank is $\bar{R}_j = (n+1)/2$ regardless of ties. Define the centred rank vectors $\tilde{R}_{\cdot j} = R_{\cdot j} - \bar{R}_j \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. The Spearman correlation between columns i and j is the Pearson correlation of their rank vectors:

$$\rho_S(i,j) \;=\; \frac{\sum_{k=1}^{n}(R_{ki} - \bar{R}_i)(R_{kj} - \bar{R}_j)}{\sqrt{\sum_{k=1}^{n}(R_{ki} - \bar{R}_i)^2}\,\sqrt{\sum_{k=1}^{n}(R_{kj} - \bar{R}_j)^2}}.$$

In matrix form, with $R = [R_{\cdot 1}, \ldots, R_{\cdot p}]$, $\mu = (n+1)\mathbf{1}_p/2$ for $\mathbf{1}_p \in \mathbb{R}^p$, and $S_R = \big(R - \mathbf{1}\mu^\top\big)^\top \big(R - \mathbf{1}\mu^\top\big)/(n-1)$, the Spearman correlation matrix is

$$\widehat{\rho}_S \;=\; D^{-1/2} S_R D^{-1/2}, \qquad D \;=\; \mathrm{diag}(\mathrm{diag}(S_R)).$$

When there are no ties, the familiar rank-difference formula obtains

$$\rho_S(i,j) \;=\; 1 - \frac{6}{n(n^2-1)} \sum_{k=1}^{n} d_k^2, \quad d_k \;=\; R_{ki} - R_{kj},$$

but this expression does *not* hold under ties; computing Pearson on mid-ranks (as above) is the standard tie-robust approach. Without ties, $\mathrm{Var}(R_{\cdot j}) = (n^2-1)/12$; with ties, the variance is smaller.

$\rho_S(i,j) \in [-1, 1]$ and $\widehat{\rho}_S$ is symmetric positive semi-definite by construction (up to floating-point error). The implementation symmetrises the result to remove round-off asymmetry. Spearman's correlation is invariant to strictly monotone transformations applied separately to each variable.

Computation. Each column is ranked (mid-ranks) to form R. The product $R^\top R$ is computed via a 'BLAS' symmetric rank update ('SYRK'), and centred using

$$(R - \mathbf{1}\mu^\top)^\top (R - \mathbf{1}\mu^\top) \;=\; R^\top R \;-\; n\,\mu\mu^\top,$$

avoiding an explicit centred copy. Division by $n - 1$ yields the sample covariance of ranks; standardising by $D^{-1/2}$ gives $\widehat{\rho}_S$. Columns with zero rank variance (all values equal) are returned as `NA` along their row/column; the corresponding diagonal entry is also `NA`.

When `na_method = "pairwise"`, each (i, j) estimate is recomputed on the pairwise complete-case overlap of columns i and j. When `ci = TRUE`, confidence intervals are computed in 'C++' using the jackknife Euclidean-likelihood method of de Carvalho and Marques (2012). For a pairwise estimate $U = \hat{\rho}_S$, delete-one jackknife pseudo-values are formed as

$$Z_i = nU - (n-1)U_{(-i)}, \qquad i = 1, \ldots, n,$$

where $U_{(-i)}$ is the Spearman correlation after removing observation i. The confidence limits solve

$$\frac{n(U-\theta)^2}{n^{-1}\sum_{i=1}^{n}(Z_i - \theta)^2} = \chi^2_{1,\,\texttt{conf_level}}.$$

Ranking costs $O\big(p\,n \log n\big)$; forming and normalising $R^\top R$ costs $O\big(np^2\big)$ with $O(p^2)$ additional memory. The optional jackknife Euclidean-likelihood confidence intervals add per-pair delete-one recomputation work and are intended for inference rather than raw-matrix throughput.

Value

A symmetric numeric matrix where the `(i, j)`-th element is the Spearman correlation between the `i`-th and `j`-th numeric columns of the input. When `ci = TRUE`, the object also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, and `conf.level`. When pairwise-complete evaluation is used, pairwise sample sizes are stored in `attr(x, "diagnostics")$n_complete`.

Invisibly returns the `spearman_rho` object.

A `ggplot` object representing the heatmap.

Note

Missing values are rejected when `na_method = "error"`. Columns with fewer than two usable observations are excluded.

Author(s)

Thiago de Paula Oliveira

References

Spearman, C. (1904). The proof and measurement of association between two things. International Journal of Epidemiology, 39(5), 1137-1150.

de Carvalho, M., & Marques, F. (2012). Jackknife Euclidean likelihood-based inference for Spearman's rho. North American Actuarial Journal, 16(4), 487-492.

See Also

print.spearman_rho, plot.spearman_rho

Examples

```
## Monotone transformation invariance (Spearman is rank-based)
set.seed(123)
n <- 400; p <- 6; rho <- 0.6
Sigma <- rho^abs(outer(seq_len(p), seq_len(p), "-"))
L <- chol(Sigma)
X <- matrix(rnorm(n * p), n, p) %*% L
colnames(X) <- paste0("V", seq_len(p))

X_mono <- X
X_mono[, 1] <- exp(X_mono[, 1])
X_mono[, 2] <- log1p(exp(X_mono[, 2]))
X_mono[, 3] <- X_mono[, 3]^3

sp_X <- spearman_rho(X)
sp_m <- spearman_rho(X_mono)
summary(sp_X)
round(max(abs(sp_X - sp_m)), 3)
plot(sp_X)

## Confidence intervals
sp_ci <- spearman_rho(X[, 1:3], ci = TRUE)
print(sp_ci, show_ci = "yes")
summary(sp_ci)
estimate(sp_ci)
tidy(sp_ci)
ci(sp_ci)
confint(sp_ci)

## Ties handled via mid-ranks
tied <- cbind(
  a = rep(1:5, each = 20),
  b = rep(5:1, each = 20) + rnorm(100, sd = 0.1),
  c = as.numeric(gl(10, 10))
)
sp_tied <- spearman_rho(tied, ci = TRUE)
print(sp_tied, digits = 2, show_ci = "yes")

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(sp_X)
}
```

---

summary.ccc_rm_reml *Summary Method for* ccc_rm_reml *Objects*

Description

Produces a detailed summary of a "ccc_rm_reml" object, including Lin's CCC estimates and associated variance component estimates per method pair.

Usage

```
## S3 method for class 'ccc_rm_reml'
summary(
  object,
  digits = 4,
  ci_digits = 2,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.ccc_rm_reml'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

object	An object of class "ccc_rm_reml", as returned by ccc_rm_reml().
digits	Integer; number of decimal places to round CCC estimates and components.
ci_digits	Integer; decimal places for confidence interval bounds.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	Character string indicating whether to show confidence intervals: "yes" shows CI information when available and "no" suppresses it.
...	Passed to print.data.frame.
x	An object of class "summary.ccc_rm_reml".

Value

A data frame of class "summary.ccc_rm_reml" with columns: item1, item2, estimate, and optionally lwr, upr, plus canonical repeated-measures counts n_subjects and n_obs. Method-specific columns retain the scientific variance component outputs: sigma2_subject, sigma2_subject_method, sigma2_subject_time, sigma2_error, sigma2_extra, SB, and se_ccc.

---

summary.corr_edge_list *S3 Summary for Edge-List Correlation Results*

Description

Representation-first summary for edge-list outputs.

Usage

```
## S3 method for class 'corr_edge_list'
summary(object, topn = NULL, show_ci = NULL, ...)
```

Arguments

object	An edge-list correlation result.
topn	Optional number of head/tail rows when preview is truncated.
show_ci	One of "yes" or "no".
...	Unused.

Value

A standardized summary data frame with class c("summary.corr_result", "data.frame") (plus compatibility classes).

---

summary.corr_matrix *S3 Summary for Dense Correlation Results*

Description

Representation-first summary for dense correlation outputs.

Usage

```
## S3 method for class 'corr_matrix'
summary(object, topn = NULL, show_ci = NULL, ...)
```

Arguments

object	A dense correlation result (corr_matrix).
topn	Optional number of head/tail rows when preview is truncated.
show_ci	One of "yes" or "no".
...	Unused.

Value

A standardized summary data frame with class c("summary.corr_result", "data.frame") (plus compatibility classes).

---

summary.corr_sparse *S3 Summary for Sparse Correlation Results*

Description

Representation-first summary for sparse correlation outputs.

Usage

```
## S3 method for class 'corr_sparse'
summary(object, topn = NULL, show_ci = NULL, ...)
```

Arguments

object	A sparse correlation result.
topn	Optional number of head/tail rows when preview is truncated.
show_ci	One of "yes" or "no".
...	Unused.

Value

A standardized summary data frame with class c("summary.corr_result", "data.frame") (plus compatibility classes).

`tetrachoric` *Pairwise Tetrachoric Correlation*

Description

Computes the tetrachoric correlation for either a pair of binary variables or all pairwise combinations of binary columns in a matrix/data frame.

Usage

```
tetrachoric(
  data,
  y = NULL,
  na_method = c("error", "pairwise"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  correct = 0.5,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'tetrachoric_corr'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'tetrachoric_corr'
plot(
  x,
  title = "Tetrachoric correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  ...
)
```

```
## S3 method for class 'tetrachoric_corr'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.tetrachoric_corr'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

`data`	A binary vector, matrix, or data frame. In matrix/data-frame mode, only binary columns are retained.
`y`	Optional second binary vector. When supplied, the function returns a single tetrachoric correlation estimate.
`na_method`	Character scalar controlling missing-data handling. `"error"` rejects missing values. `"pairwise"` uses pairwise complete cases.
`ci`	Logical (default `FALSE`). If `TRUE`, attach model-based large-sample Wald confidence intervals derived from the observed information matrix of the latent-variable likelihood.
`p_value`	Logical (default `FALSE`). If `TRUE`, attach model-based large-sample Wald p-values and test statistics for each estimated latent correlation.
`conf_level`	Confidence level used when `ci = TRUE`. Default is `0.95`.
`correct`	Non-negative continuity correction added to zero-count cells. Default is `0.5`.
`output`	Output representation for the computed estimates. • `"matrix"` (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • `"sparse"`: sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding.

- "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.

threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Additional arguments passed to print().
x	An object of class summary.tetrachoric_corr.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
title	Plot title. Default is "Tetrachoric correlation heatmap".
low_color	Color for the minimum correlation.
high_color	Color for the maximum correlation.
mid_color	Color for zero correlation.
value_text_size	Font size used in tile labels.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class tetrachoric_corr.
ci_digits	Integer; digits for confidence limits in the pairwise summary.
p_digits	Integer; digits for p-values in the pairwise summary.

Details

The tetrachoric correlation assumes that the observed binary variables arise by dichotomising latent standard-normal variables. Let $Z_1, Z_2 \sim N(0,1)$ with latent correlation ρ, and define observed binary variables by thresholds τ_1, τ_2:

$$X = \mathbf{1}\{Z_1 > \tau_1\}, \qquad Y = \mathbf{1}\{Z_2 > \tau_2\}.$$

If the observed 2×2 table has counts n_{ij} for $i, j \in \{0, 1\}$, the marginal proportions determine the thresholds:

$$\tau_1 = \Phi^{-1}\big(P(X = 0)\big), \qquad \tau_2 = \Phi^{-1}\big(P(Y = 0)\big).$$

The estimator returned here is the maximum-likelihood estimate of the latent correlation ρ, obtained by maximizing the multinomial log-likelihood built from the rectangle probabilities of the bivariate normal distribution:

$$\ell(\rho) = \sum_{i=0}^{1} \sum_{j=0}^{1} n_{ij} \log \pi_{ij}(\rho; \tau_1, \tau_2),$$

where π_{ij} are the four bivariate-normal cell probabilities implied by ρ and the fixed thresholds. The implementation evaluates the likelihood over $\rho \in (-1, 1)$ by a coarse search followed by Brent refinement in C++.

The argument `correct` adds a continuity correction only to zero-count cells before threshold estimation and likelihood evaluation. This stabilises the estimator for sparse tables and mirrors the conventional `correct = 0.5` continuity-correction behaviour used in several latent-correlation implementations. When `correct = 0` and the observed contingency table contains zero cells, the fit is non-regular and may be boundary-driven. In those cases the returned object stores sparse-fit diagnostics, including whether the fit was classified as `boundary` or `near_boundary`.

Assumptions. The coefficient is appropriate when both observed binary variables are viewed as thresholded versions of jointly normal latent variables. The optional p-values and confidence intervals adopt this latent-normal interpretation and use the same likelihood that defines the tetrachoric estimate. These inferential quantities are therefore model-based and should not be interpreted as distribution-free summaries.

Inference. When `ci = TRUE` or `p_value = TRUE`, the function refits the pairwise tetrachoric model by maximum likelihood and obtains the observed information matrix numerically in C++. The reported confidence interval is a Wald interval $\hat{\rho} \pm z_{1-\alpha/2}\,\mathrm{SE}(\hat{\rho})$, and the reported p-value is from the large-sample Wald z-test for $H_0 : \rho = 0$. These inferential quantities are only computed when explicitly requested.

In matrix/data-frame mode, all pairwise tetrachoric correlations are computed between binary columns. Diagonal entries are 1 for non-degenerate columns and `NA` for columns with fewer than two observed levels. Variable-specific latent thresholds are stored in the `thresholds` attribute, and pairwise sparse-fit diagnostics are stored in `diagnostics`.

Computational complexity. For p binary variables, the matrix path evaluates $p(p-1)/2$ pairwise likelihoods. Each pair uses a one-dimensional optimisation with negligible memory overhead beyond the output matrix.

Value

If `y` is supplied, a numeric scalar with attributes `diagnostics` and `thresholds`. Otherwise a symmetric matrix of class `tetrachoric_corr` with attributes `method`, `description`, `package = "matrixCorr"`, `diagnostics`, `thresholds`, and `correct`. When `p_value = TRUE`, the returned object also carries an `inference` attribute with elements `estimate`, `statistic`, `parameter`, `p_value`, and `n_obs`. When `ci = TRUE`, it also carries a `ci` attribute with elements `est`, `lwr.ci`, `upr.ci`, `conf.level`, and `ci.method`, plus `attr(x, "conf.level")`. Scalar outputs keep the same point estimate and gain the same metadata only when inference is requested. In matrix mode, `output = "edge_list"` returns a data frame with columns `row`, `col`, `value`; `output = "sparse"` returns a symmetric sparse matrix.

Author(s)

Thiago de Paula Oliveira

References

Pearson, K. (1900). Mathematical contributions to the theory of evolution. VII. On the correlation of characters not quantitatively measurable. *Philosophical Transactions of the Royal Society A*, 195, 1-47.

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, 44(4), 443-460.

Examples

```
set.seed(123)
n <- 1000
Sigma <- matrix(c(
  1.00, 0.55, 0.35,
  0.55, 1.00, 0.45,
  0.35, 0.45, 1.00
), 3, 3, byrow = TRUE)

Z <- mnormt::rmnorm(n = n, mean = rep(0, 3), varcov = Sigma)
X <- data.frame(
  item1 = Z[, 1] > stats::qnorm(0.70),
  item2 = Z[, 2] > stats::qnorm(0.60),
  item3 = Z[, 3] > stats::qnorm(0.50)
)

tc <- tetrachoric(X)
print(tc, digits = 3)
summary(tc)
estimate(tc)
tidy(tc)
tc_ci <- tetrachoric(X, ci = TRUE)
ci(tc_ci)
confint(tc_ci)
plot(tc)
tetrachoric(X, output = "edge_list", diag = FALSE)
tetrachoric(X, output = "sparse", threshold = 0.4, diag = FALSE)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(tc)
}

# latent Pearson correlations used to generate the binary items
round(stats::cor(Z), 2)
```

---

view_corr_shiny *Interactive Shiny viewer for matrixCorr objects*

Description

Launches an interactive Shiny gadget that displays correlation heatmaps with filtering, clustering, and hover inspection. The viewer accepts any matrixCorr correlation result (for example the outputs from `pearson_corr()`, `spearman_rho()`, `kendall_tau()`, `bicor()`, `pbcor()`, `wincor()`, `skipped_corr()`, `pcorr()`, `dcor()`, `bcdcor()`, or `shrinkage_corr()`), a plain matrix, or a named list of such objects. When a list is supplied the gadget offers a picker to switch between results.

Usage

```
view_corr_shiny(x, title = NULL, default_max_vars = 40L)
```

Arguments

x	A correlation result, a numeric matrix, or a named list of those objects. Each element must be square with matching row/column names.
title	Optional character title shown at the top of the gadget.
default_max_vars	Integer; maximum number of variables pre-selected when the app opens. Defaults to 40 so very wide matrices start collapsed.

Details

This helper lives in Suggests; it requires the shiny and shinyWidgets packages at runtime and will optionally convert the plot to an interactive widget when plotly is installed. Variable selection uses a searchable picker, and clustering controls let you reorder variables via hierarchical clustering on either absolute or signed correlations with a choice of linkage methods.

Value

Invisibly returns NULL; the function is called for its side effect of launching a Shiny gadget.

Examples

```
if (interactive()) {
  data <- mtcars
  results <- list(
    Pearson = pearson_corr(data),
    Spearman = spearman_rho(data),
    Kendall = kendall_tau(data)
  )
  view_corr_shiny(results)
}
```

---

view_rmcorr_shiny *Interactive Shiny Viewer for Repeated-Measures Correlation*

Description

Launches a dedicated Shiny gadget for repeated-measures correlation matrix objects of class "rmcorr_matrix". The viewer combines the correlation heatmap with a pairwise scatterplot panel that rebuilds the corresponding two-variable "rmcorr" fit for user-selected variables.

Usage

```
view_rmcorr_shiny(x, title = NULL, default_max_vars = 40L)
```

Arguments

x	An object of class "rmcorr_matrix" or a named list of such objects.
title	Optional character title shown at the top of the gadget.
default_max_vars	Integer; maximum number of variables pre-selected in the heatmap view when the app opens. Defaults to 40.

Details

This helper requires the **shiny** and **shinyWidgets** packages at runtime and will optionally use **plotly** for the heatmap when available. The pairwise panel reuses the package's regular plot.rmcorr() method, so the Shiny scatterplot matches the standard pairwise repeated-measures correlation plot. To rebuild pairwise fits from a returned "rmcorr_matrix" object, the matrix must have been created with keep_data = TRUE.

Value

Invisibly returns NULL; the function is called for its side effect of launching a Shiny gadget.

Examples

```
if (interactive()) {
  set.seed(2026)
  n_subjects <- 20
  n_rep <- 4
  subject <- rep(seq_len(n_subjects), each = n_rep)
  subj_eff_x <- rnorm(n_subjects, sd = 1.5)
  subj_eff_y <- rnorm(n_subjects, sd = 2.0)
  within_signal <- rnorm(n_subjects * n_rep)

  dat <- data.frame(
    subject = subject,
    x = subj_eff_x[subject] + within_signal + rnorm(n_subjects * n_rep, sd = 0.2),
    y = subj_eff_y[subject] + 0.8 * within_signal + rnorm(n_subjects * n_rep, sd = 0.3),
    z = subj_eff_y[subject] - 0.4 * within_signal + rnorm(n_subjects * n_rep, sd = 0.4)
  )

  fit_mat <- rmcorr(
    dat,
    response = c("x", "y", "z"),
    subject = "subject",
    keep_data = TRUE
  )
  view_rmcorr_shiny(fit_mat)
}
```

`weighted_kappa` *Pairwise Weighted Cohen's Kappa for Ordered Ratings*

Description

Computes weighted Cohen's kappa for either a pair of ordinal rating vectors or all pairwise combinations of ordinal columns in a matrix or data frame.

Usage

```
weighted_kappa(
  data,
  y = NULL,
  weights = c("quadratic", "linear", "unweighted"),
  levels = NULL,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)

## S3 method for class 'weighted_kappa'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)
```

Arguments

`data`	In matrix mode, a matrix or data frame whose rows are observational units and whose columns are raters or classifiers. Each column contains ordered category ratings. In two-vector mode, the first ordinal rating vector.
`y`	Optional second ordinal rating vector. When supplied, the function returns a single weighted kappa estimate for `data` and `y`.

weights	Weight specification. The default "quadratic" uses Fleiss-Cohen-style agreement weights. "linear" uses equal-spacing agreement weights. "unweighted" reduces to ordinary Cohen's kappa. Custom numeric weight matrices must be symmetric agreement weights with diagonal 1 and entries in [0, 1].
levels	Optional ordered category levels. Weighted kappa depends on category order and will not silently alphabetise arbitrary labels. If omitted, order is inferred only when all involved ratings are ordered factors with identical levels or when all involved ratings are numeric or integer and can be ordered by their sorted unique values.
na_method	Character scalar controlling missing-data handling. "error" rejects missing values. "complete" applies listwise deletion across retained columns before computing the matrix. In matrix mode, "pairwise" uses pair-specific complete observations.
ci	Logical; if TRUE, attach large-sample delta-method confidence intervals.
p_value	Logical; if TRUE, attach large-sample delta-method Wald test statistics and p-values for $H_0 : \kappa_w = 0$.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
n_threads	Integer ≥ 1. Number of OpenMP threads used by the matrix C++ backend. Defaults to getOption("matrixCorr.threads", 1L).
output	Output representation for matrix mode. • "matrix" (default): full dense matrix. • "sparse": sparse matrix from **Matrix** containing only retained entries after thresholding. • "edge_list": long-form data frame representation.
threshold	Non-negative absolute-value filter for non-matrix outputs: retain entries with abs(value) >= threshold. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in sparse and edge-list outputs.
...	Reserved for future extensions. Unsupported extra arguments are rejected.
x	A matrix-style weighted_kappa object.
digits	Integer; number of decimal places for displayed values.
n	Optional preview row threshold.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns.
width	Optional display width.
show_ci	One of "yes" or "no".

Details

This implementation uses agreement/similarity weights internally:

- "unweighted": $w_{ij} = 1(i = j)$
- "linear": $w_{ij} = 1 - |i - j|/(K - 1)$
- "quadratic": $w_{ij} = 1 - (i - j)^2/(K - 1)^2$

The "quadratic" scheme is the default and corresponds to Fleiss-Cohen-style agreement weights. The "linear" scheme corresponds to equal-spacing agreement weights.

For matrix mode, rows are shared observational units and columns are raters or classifiers. A common category map is resolved in R and the main computation is performed in C++. Missing-data handling follows the usual **matrixCorr** na_method conventions. Pairwise complete counts are stored in attr(x, "diagnostics")$n_complete.

Confidence intervals and standard errors. Confidence intervals and p-values use the exact large-sample multinomial delta-method formula. Let $A_w = \sum_{a,b} w_{ab} p_{ab}$ be the observed weighted agreement, $E_w = \sum_{a,b} w_{ab} q_a r_b$ the expected weighted agreement, and $D = 1 - E_w$. Define the weighted margin summaries

$$u_a = \sum_b w_{ab} r_b, \qquad v_b = \sum_a w_{ab} q_a.$$

For each cell (a, b), the backend uses

$$g_{ab} = \frac{D\, w_{ab} + (A_w - 1)\,(u_a + v_b)}{D^2}.$$

The variance estimator is

$$\widehat{\mathrm{Var}}(\hat{\kappa}_w) = \frac{1}{n}\left(\sum_{a,b} p_{ab} g_{ab}^2 - \Big(\sum_{a,b} p_{ab} g_{ab}\Big)^2\right),$$

where n is the number of complete paired ratings. The standard error is $\widehat{\mathrm{se}}(\hat{\kappa}_w) = \sqrt{\widehat{\mathrm{Var}}(\hat{\kappa}_w)}$ and the CI is the Wald interval

$$\hat{\kappa}_w \pm z_{1-\alpha/2}\, \widehat{\mathrm{se}}(\hat{\kappa}_w),$$

truncated to $[-1, 1]$ in the returned result. Use cohen_kappa() for unordered nominal categories where all disagreements are equally serious.

Value

If y is supplied, a scalar S3 object of class c("weighted_kappa", "numeric") is returned with diagnostics and optional ci and inference attributes. Otherwise a symmetric matrix-style result with estimator class weighted_kappa.

Author(s)

Thiago de Paula Oliveira

References

Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4), 213-220. doi:10.1037/h0026256

See Also

cohen_kappa() for unweighted two-rater nominal agreement; multirater_kappa() for panel-level nominal agreement among three or more raters.

Examples

```
raters <- data.frame(
  r1 = ordered(c("low", "low", "mid", "high", "high"),
               levels = c("low", "mid", "high")),
  r2 = ordered(c("low", "mid", "mid", "high", "high"),
               levels = c("low", "mid", "high")),
  r3 = ordered(c("low", "low", "high", "high", "mid"),
               levels = c("low", "mid", "high"))
)

wk <- weighted_kappa(raters)
print(wk)
summary(wk)
estimate(wk)
tidy(wk)
plot(wk)

x <- raters$r1
y <- raters$r2
weighted_kappa(x, y, weights = "linear")
```

---

wincor *Pairwise Winsorized correlation*

Description

Computes all pairwise Winsorized correlation coefficients for the numeric columns of a matrix or data frame using a high-performance 'C++' backend.

This function Winsorizes each margin at proportion `tr` and then computes ordinary Pearson correlation on the Winsorized values. It is a simple robust alternative to Pearson correlation when the main concern is unusually large or small observations in the marginal distributions.

Usage

```
wincor(
  data,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_threads = getOption("matrixCorr.threads", 1L),
  tr = 0.2,
  n_boot = 500L,
  seed = NULL,
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE
```

```
)

## S3 method for class 'wincor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
  ...
)

## S3 method for class 'wincor'
plot(
  x,
  title = "Winsorized correlation heatmap",
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  show_value = TRUE,
  ...
)

## S3 method for class 'wincor'
summary(
  object,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  p_digits = 4,
  show_ci = NULL,
  ...
)

## S3 method for class 'summary.wincor'
print(
  x,
  digits = NULL,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  show_ci = NULL,
```

```
  ...
)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be excluded.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each estimate on its own pairwise complete-case overlap. This is permissive, but different matrix entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes the estimator on the common complete sample.
ci	Logical (default FALSE). If TRUE, attach percentile bootstrap confidence intervals for each pairwise estimate.
p_value	Logical (default FALSE). If TRUE, attach the method-specific large-sample test statistic and two-sided p-value for each pairwise estimate.
conf_level	Confidence level used when ci = TRUE. Default 0.95.
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
tr	Winsorization proportion in [0, 0.5). For a sample of size n, let $g = \lfloor tr \cdot n \rfloor$; the g smallest observations are set to the $(g+1)$-st order statistic and the g largest observations are set to the $(n-g)$-th order statistic. Default 0.2.
n_boot	Integer ≥ 1. Number of bootstrap resamples used when ci = TRUE. Default 500.
seed	Optional positive integer used to seed the bootstrap resampling when ci = TRUE. If NULL, the current random-number stream is used.
output	Output representation for the computed estimates. • "matrix" (default): full dense matrix; best when you need matrix algebra, dense heatmaps, or full compatibility with existing code. • "sparse": sparse matrix from **Matrix** containing only retained entries; best when many values are dropped by thresholding. • "edge_list": long-form data frame with columns row, col, value; convenient for filtering, joins, and network-style workflows.
threshold	Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger associations (typically with output = "sparse" or "edge_list"). Keep threshold = 0 to retain all values. Must be 0 when output = "matrix".
diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
x	An object of class summary.wincor.
digits	Integer; number of digits to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.

max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
show_ci	One of "yes" or "no".
...	Additional arguments passed to the underlying print or plot helper.
title	Character; plot title.
low_color, high_color, mid_color	Colors used in the heatmap.
value_text_size	Numeric text size for overlaid cell values.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class wincor.
ci_digits	Integer; digits used for confidence limits in pairwise summaries.
p_digits	Integer; digits used for p-values in pairwise summaries.

Details

Let $X \in \mathbb{R}^{n \times p}$ be a numeric matrix with rows as observations and columns as variables. For a column $x = (x_i)_{i=1}^n$, write the order statistics as $x_{(1)} \le \cdots \le x_{(n)}$ and let $g = \lfloor tr \cdot n \rfloor$. The Winsorized values can be written as

$$x_i^{(w)} \;=\; \max\bigl\{x_{(g+1)},\, \min(x_i, x_{(n-g)})\bigr\}.$$

For two columns x and y, the Winsorized correlation is the ordinary Pearson correlation computed from $x^{(w)}$ and $y^{(w)}$:

$$r_w(x,y) \;=\; \frac{\sum_{i=1}^n (x_i^{(w)} - \bar{x}^{(w)})(y_i^{(w)} - \bar{y}^{(w)})}{\sqrt{\sum_{i=1}^n (x_i^{(w)} - \bar{x}^{(w)})^2}\,\sqrt{\sum_{i=1}^n (y_i^{(w)} - \bar{y}^{(w)})^2}}.$$

In matrix form, let $X^{(w)}$ contain the Winsorized columns and define the centred, unit-norm columns

$$z_{\cdot j} = \frac{x_{\cdot j}^{(w)} - \bar{x}_j^{(w)}\mathbf{1}}{\sqrt{\sum_{i=1}^n (x_{ij}^{(w)} - \bar{x}_j^{(w)})^2}}, \qquad j = 1, \ldots, p.$$

If $Z = [z_{\cdot 1}, \ldots, z_{\cdot p}]$, then the Winsorized correlation matrix is

$$R_w \;=\; Z^\top Z.$$

Winsorization acts on each margin separately, so it guards against marginal outliers and heavy tails but does not target unusual points in the joint cloud. This implementation Winsorizes each column in 'C++', centres and normalises it, and forms the complete-data matrix from cross-products. With na_method = "pairwise", each pair is recomputed on its overlap of non-missing rows. As with Pearson correlation, the complete-data path yields a symmetric positive semidefinite matrix, whereas pairwise deletion can break positive semidefiniteness. If the Winsorized variance of a column is zero, correlations involving that column are returned as NA.

When p_value = TRUE, inference follows the method-specific test based on

$$T_{ij} = r_{w,ij}\sqrt{\frac{n_{ij}-2}{1-r_{w,ij}^2}},$$

evaluated against a t-distribution with $n_{ij} - 2g_{ij} - 2$ degrees of freedom, where $g_{ij} = \lfloor tr \cdot n_{ij} \rfloor$ and n_{ij} is the pairwise complete-case sample size for the corresponding column pair. The p-value is reported only when the pair is not identical and the resulting degrees of freedom are positive. When ci = TRUE, the interval is a percentile bootstrap interval based on n_{boot} resamples drawn from the pairwise complete cases. If $\tilde{r}_{w,(1)} \le \cdots \le \tilde{r}_{w,(B)}$ denotes the sorted bootstrap sample of finite estimates with B retained resamples, the reported limits are

$$\tilde{r}_{w,(\ell)} \quad \text{and} \quad \tilde{r}_{w,(u)},$$

where $\ell = \lfloor (\alpha/2)B + 0.5 \rfloor$ and $u = \lfloor (1-\alpha/2)B + 0.5 \rfloor$ for $\alpha = 1 - \mathrm{conf_level}$. Resamples that yield undefined estimates are discarded before the percentile limits are formed.

Computational complexity. In the complete-data path, Winsorizing the columns requires sorting within each column, and forming the cross-product matrix costs $O(np^2)$ with $O(p^2)$ output storage. When ci = TRUE, the bootstrap cost is incurred separately for each column pair.

Value

A symmetric correlation matrix with class wincor and attributes method = "winsorized_correlation", description, and package = "matrixCorr". When ci = TRUE, the returned object also carries a ci attribute with elements est, lwr.ci, upr.ci, conf.level, and ci.method, plus attr(x, "conf.level"). When p_value = TRUE, it also carries an inference attribute with elements estimate, statistic, parameter, p_value, n_obs, and alternative. When either inferential option is requested, the object also carries diagnostics$n_complete.

Author(s)

Thiago de Paula Oliveira

References

Wilcox, R. R. (1993). Some results on a Winsorized correlation coefficient. British Journal of Mathematical and Statistical Psychology, 46(2), 339-349. doi:10.1111/j.20448317.1993.tb01020.x

Wilcox, R. R. (2012). Introduction to Robust Estimation and Hypothesis Testing (3rd ed.). Academic Press.

See Also

pbcor(), skipped_corr(), bicor()

Examples

```
set.seed(11)
X <- matrix(rnorm(180 * 4), ncol = 4)
X[sample(length(X), 6)] <- X[sample(length(X), 6)] - 12
```

```
R <- wincor(X, tr = 0.2)
print(R, digits = 2)
summary(R)
estimate(R)
tidy(R)
plot(R)

## Bootstrap confidence intervals
R_ci <- wincor(X, tr = 0.2, ci = TRUE, n_boot = 49, seed = 11)
ci(R_ci)
confint(R_ci)

# Interactive viewing (requires shiny)
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  view_corr_shiny(R)
}
```

`xi_corr` *Pairwise Chatterjee Rank Correlation*

Description

Computes the directed Chatterjee rank correlation coefficient for numeric vectors or for all directed column pairs of a numeric matrix/data frame. The matrix orientation is `result[i, j] = xi(V_i, V_j)`, where `V_i` is the predictor/sorting variable and `V_j` is the response/ranked variable.

Usage

```
xi_corr(
  data,
  y = NULL,
  na_method = c("error", "pairwise", "complete"),
  ci = FALSE,
  conf_level = 0.95,
  ci_method = c("auto", "dette_kroll", "n_choose_m"),
  bootstrap_reps = 999L,
  m = NULL,
  large_sample_cutoff = 1000L,
  bias_correction = c("none", "upper_bound"),
  tie_method = c("random", "first"),
  seed = NULL,
  n_threads = getOption("matrixCorr.threads", 1L),
  output = c("matrix", "sparse", "edge_list"),
  threshold = 0,
  diag = TRUE,
  ...
)
```

```
## S3 method for class 'chatterjee_xi'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  ci_digits = 3,
  show_ci = NULL,
  ...
)

## S3 method for class 'chatterjee_xi_scalar'
print(x, digits = 4, ci_digits = 4, ...)

## S3 method for class 'chatterjee_xi_scalar'
summary(object, digits = 4, ci_digits = 4, ...)

## S3 method for class 'summary.chatterjee_xi_scalar'
print(x, digits = NULL, ci_digits = NULL, ...)
```

Arguments

data	A numeric matrix or a data frame with at least two numeric columns, or a numeric predictor vector when y is supplied. All non-numeric columns will be excluded in matrix/data-frame mode. Each retained column must have at least two non-missing values.
y	Optional numeric response vector for two-vector mode. When supplied, xi_corr(data, y) estimates $\xi(data, y)$, where data is the sorting/predictor variable and y is the response/ranked variable.
na_method	Character scalar controlling missing-data handling. "error" rejects missing, NaN, and infinite values. "pairwise" recomputes each directed pair on its own pairwise complete-case overlap. This is permissive, but different directed entries may be based on different rows. "complete" performs listwise deletion once across the retained numeric columns and then computes all directed entries on the same complete sample.
ci	Logical (default FALSE). If TRUE, attach m-out-of-n bootstrap confidence intervals for the directed Chatterjee estimates. In two-vector mode the scalar estimate carries a ci attribute.
conf_level	Confidence level used when ci = TRUE. Default is 0.95.
ci_method	Confidence interval method: • "auto" (default): use "dette_kroll" when the pair-specific complete-case sample size is less than or equal to large_sample_cutoff, and "n_choose_m" for larger samples.

- "dette_kroll": m-out-of-n variance estimation followed by a normal interval, following Dette and Kroll.
- "n_choose_m": non-parametric m-out-of-n basic interval based on the centred and scaled bootstrap statistic, following the large-sample n-choose-m approach used by Dalitz, Arning and Goebbels.

bootstrap_reps Number of m-out-of-n bootstrap replicates used when ci = TRUE. Larger values reduce Monte Carlo noise but increase runtime.

m Optional subsample size for the m-out-of-n bootstrap. If NULL, method-specific defaults are used: floor(sqrt(n)) for "dette_kroll" and round(2 * sqrt(n)) for "n_choose_m", each bounded to [2, n - 1].

large_sample_cutoff

For ci_method = "auto", use "n_choose_m" when pair-specific n_complete is greater than this cutoff; otherwise use "dette_kroll".

bias_correction

Finite-sample normalisation:

- "none" (default): return Chatterjee's original finite-sample statistic without rescaling. This reproduces the estimator as originally defined; its sample maximum is below one for finite continuous samples.
- "upper_bound": divide by the raw finite-sample upper-bound estimate $\xi_n(Y, Y)$ and return $\max(-1, \xi_n(X, Y)/\xi_n(Y, Y))$. Use this option when you want the bias-reduced version proposed by Dalitz, Arning and Goebbels (2024); it is not the default.

tie_method How ties in the sorting variable X are broken:

- "random" (default): randomly permute observations within tied X groups, matching Chatterjee's random tie-breaking definition.
- "first": preserve stable input order within tied X groups, useful for deterministic diagnostics and reproducible comparisons.

seed Optional positive integer seed for reproducible tie breaking and bootstrap resampling.

n_threads Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L). Random tie breaking and confidence-interval paths are evaluated serially to preserve R RNG semantics.

output Output representation for the computed directed estimates.

- "matrix" (default): full dense directed matrix; best when you need matrix algebra, heatmaps, or full compatibility with existing code.
- "sparse": sparse matrix from **Matrix** containing retained directed entries; useful when thresholding leaves relatively few values.
- "edge_list": long-form data frame with columns row, col, and value; convenient for filtering, joins, and graph/network workflows. Because $\xi(X, Y)$ is directed, both directions are represented when retained.

threshold Non-negative absolute-value filter for non-matrix outputs: keep entries with abs(value) >= threshold. Use threshold > 0 when you want only stronger directed associations with output = "sparse" or "edge_list". Must be 0 when output = "matrix".

diag	Logical; whether to include diagonal entries in "sparse" and "edge_list" outputs.
...	Compatibility arguments. The deprecated check_na alias is accepted.
x	An object of class chatterjee_xi.
digits	Integer; number of decimal places to print.
n	Optional row threshold for compact preview output.
topn	Optional number of leading/trailing rows to show when truncated.
max_vars	Optional maximum number of visible columns; NULL derives this from console width.
width	Optional display width; defaults to getOption("width").
ci_digits	Integer; digits for confidence limits.
show_ci	One of "yes" or "no".
object	An object of class chatterjee_xi_scalar.

Details

Let $(X_k, Y_k), k = 1, \ldots, n$, be complete paired observations. Chatterjee's rank correlation is directed: $\xi(X, Y)$ measures how well Y is functionally determined by X. It is not a symmetric association measure, so in matrix mode

$$\text{result}_{ij} = \xi(X_{\cdot i}, X_{\cdot j})$$

need not equal result_{ji}. The row variable is always the sorting/predictor variable and the column variable is the response/ranked variable.

For a given directed pair, observations are sorted by X. Let r_i be the number of response values $Y_j \leq Y_i$, and let l_i be the number of response values $Y_j \geq Y_i$, evaluated for the i-th observation in sorted-by-X order. The tied-response finite-sample estimator is

$$\xi_n = 1 - \frac{n \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{2 \sum_{i=1}^{n} l_i (n - l_i)}.$$

If all response values are equal, the denominator is zero and the estimate is NA. When there are no ties in Y, this reduces to the familiar rank-difference expression

$$\xi_n = 1 - \frac{3 \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1}.$$

The default bias_correction = "none" returns Chatterjee's original finite-sample estimator. This raw statistic is not forced to attain one under perfect deterministic dependence. In continuous data without ties, its finite-sample upper bound is

$$\xi_{\max} = \xi_n(Y, Y) = \frac{n - 2}{n + 1}.$$

For example, when $n = 20$, the largest possible raw value is $18/21 \approx 0.857$. This is a property of the estimator rather than an implementation error. Use bias_correction = "upper_bound" when you want the bias-reduced Dalitz-Arning-Goebbels normalisation, which rescales by $\xi_n(Y, Y)$.

Ties in the sorting variable X are handled before computing adjacent rank differences. Chatterjee's definition uses random tie breaking, provided here by `tie_method = "random"`. For reproducible diagnostics or when the input order should define tied-X ordering, use `tie_method = "first"`. Ties in the response variable Y are handled by the general r_i, l_i formula above and do not require random tie breaking.

Confidence intervals. The ordinary n-out-of-n bootstrap is not used for Chatterjee's coefficient. Both available intervals use m-out-of-n subsampling without replacement, consistent with the bootstrap family considered for Chatterjee's rank correlation. With `ci_method = "dette_kroll"`, subsamples of size m are drawn without replacement and the limiting standard deviation is estimated as

$$\widehat{\sigma} = \sqrt{m}\,\mathrm{sd}(\xi_m^*).$$

The reported interval is the normal interval

$$\xi_n \pm z_{1-\alpha/2}\widehat{\sigma}/\sqrt{n}, \quad \alpha = 1 - \texttt{conf_level}.$$

If `m = NULL`, this method uses $m = \lfloor\sqrt{n}\rfloor$, bounded to $[2, n-1]$.

With `ci_method = "n_choose_m"`, subsamples are also drawn without replacement, but the interval is obtained by inverting the empirical quantiles of the centred and scaled statistic

$$T^* = \sqrt{m}(\xi_m^* - \xi_n).$$

If $q_{\alpha/2}$ and $q_{1-\alpha/2}$ are the bootstrap quantiles, the basic m-out-of-n limits are

$$\xi_n - q_{1-\alpha/2}/\sqrt{n} \quad \text{and} \quad \xi_n - q_{\alpha/2}/\sqrt{n}.$$

If finite Monte Carlo quantiles fall entirely on one side of zero, the inversion is anchored at zero so that the reported interval contains the observed estimate; the bounds are not clipped to the population parameter range. If `m = NULL`, this method uses $m = \mathrm{round}(2\sqrt{n})$, bounded to $[2, n-1]$, following the implementation rule used by Dalitz, Arning and Goebbels.

Which CI method should be used? The Dette-Kroll interval is the more conservative default for small to moderate complete-case sample sizes in this implementation, because it uses the m-out-of-n bootstrap only to estimate a normal-approximation standard error. The non-parametric n-choose-m interval is useful for larger samples when a direct m-out-of-n bootstrap interval is preferred, but Dalitz, Arning and Goebbels report that this interval may need fairly large n to approach nominal coverage in some settings. Therefore `ci_method = "auto"` uses `"dette_kroll"` when pair-specific `n_complete <= large_sample_cutoff` and `"n_choose_m"` when `n_complete > large_sample_cutoff`. Increase `bootstrap_reps` for final analyses to reduce Monte Carlo error, and consider setting `m` explicitly when a study protocol requires a fixed subsample size.

Computation. For complete finite matrices without confidence intervals, the C++ backend precomputes each column's sorting order and response ranks and reuses them across directed pairs. This costs $O(p\,n\log n + p^2 n)$ time and $O(pn + p^2)$ memory. Pairwise missing-data evaluation and bootstrap confidence intervals recompute on the relevant complete-case samples and are correspondingly more expensive.

Value

A directed numeric matrix where the `(i, j)`-th element is Chatterjee's ξ from the `i`-th numeric column to the `j`-th numeric column. The dense matrix inherits from `c("corr_matrix", "chatterjee_xi",`

"corr_result", "matrix"). When ci = TRUE, the object also carries a ci attribute with elements est, lwr.ci, upr.ci, conf.level, ci.method, se, m, and bootstrap_reps. When pairwise-complete evaluation is used, pairwise sample sizes are stored in attr(x, "diagnostics")$n_complete. In two-vector mode, a numeric scalar is returned; when ci = TRUE, it carries confidence-interval attributes.

Author(s)

Thiago de Paula Oliveira

References

Chatterjee, S. (2021). A New Coefficient of Correlation. *Journal of the American Statistical Association*, 116, 2009-2022.

Dette, H. and Kroll, M. (2025). A simple bootstrap for Chatterjee's rank correlation. *Biometrika*, 112(1), asae045. doi:10.1093/biomet/asae045.

Lin, Z. and Han, F. (2025). Limit theorems of Chatterjee's rank correlation. arXiv:2204.08031v4. doi:10.48550/arXiv.2204.08031.

Dalitz, C., Arning, J. and Goebbels, S. (2024). A Simple Bias Reduction for Chatterjee's Correlation.

Examples

```
## Example 1: independence versus functional dependence
## Chatterjee's xi targets whether Y is determined by X.
set.seed(1)
n <- 300
x <- runif(n, -1, 1)
y_independent <- rnorm(n)
y_function <- sin(2 * pi * x)
c(
  independent = xi_corr(x, y_independent, tie_method = "first"),
  functional = xi_corr(x, y_function, tie_method = "first")
)

## Example 2: non-monotone functional dependence is directed
## x determines x^2, but x^2 does not determine the sign of x.
## The reverse raw finite-sample estimate can therefore be much smaller,
## and may be negative when sorted response ranks oscillate strongly.
x <- seq(-1, 1, length.out = 300)
y <- x^2
c(
  xi_x_to_y = xi_corr(x, y, tie_method = "first"),
  xi_y_to_x = xi_corr(y, x, tie_method = "first")
)

## Example 3: the raw finite-sample diagonal is not forced to one
## With n = 20 and no ties, the raw finite-sample maximum is 18 / 21.
## Use the Dalitz et al. upper-bound normalisation for the bias-reduced scale.
z <- 1:20
```

```
c(
  raw = xi_corr(z, z, tie_method = "first"),
  upper_bound = xi_corr(
    z, z,
    tie_method = "first",
    bias_correction = "upper_bound"
  )
)

## Example 4: directed matrix workflow
X <- cbind(
  x = x,
  square = x^2,
  sine = sin(2 * pi * x),
  noise = rnorm(length(x))
)
xi <- xi_corr(X, tie_method = "first")
print(xi, digits = 3)
summary(xi)
estimate(xi)
tidy(xi)

## Example 5: Dette-Kroll bootstrap interval for a moderate sample

xi_ci <- xi_corr(
  X[, c("x", "sine")],
  ci = TRUE,
  ci_method = "dette_kroll",
  bootstrap_reps = 49,
  seed = 1,
  tie_method = "first"
)
summary(xi_ci)
ci(xi_ci)
confint(xi_ci)
plot(xi_ci)


## Example 6: n-choose-m interval for larger samples

set.seed(2)
x_large <- runif(1200, -1, 1)
y_large <- sin(2 * pi * x_large) + rnorm(1200, sd = 0.2)
xi_corr(
  x_large,
  y_large,
  ci = TRUE,
  ci_method = "n_choose_m",
  bootstrap_reps = 99,
  seed = 2,
  tie_method = "first"
)
```

Index

[[.summary.corr_result, 4
$.summary.corr_result, 4

as.data.frame.corr_edge_list, 5

ba, 5, *37*, *41*, *68*
ba(), *110*, *118*
ba_rm, 11, *41*
ba_rm(), *110*, *118*
bcdcor, 22
bcdcor(), *84*, *187*, *211*
bicor, 26
bicor(), *137*, *198*, *211*, *221*
biserial, 33
biweight_mid_corr
        *(*deprecated-matrixCorr*)*, 85
bland_altman *(*deprecated-matrixCorr*)*, 85
bland_altman_repeated
        *(*deprecated-matrixCorr*)*, 85

ccc, *10*, 37, *62*, *68*
ccc(), *110*, *118*, *181*
ccc_glmm, 41
ccc_lmm_reml *(*deprecated-matrixCorr*)*, 85
ccc_pairwise_u_stat
        *(*deprecated-matrixCorr*)*, 85
ccc_rm_reml, *10*, *41*, 46, *61*, *62*
ccc_rm_reml(), *74*, *110*, *118*, *204*
ccc_rm_ustat, *10*, *41*, *55*, 59
ci *(*estimate*)*, 90
cia, *41*, 63
cia(), *72*, *74*
cia_rm, 69
coef.ccc_glmm *(*estimate*)*, 90
coef.corr_result *(*estimate*)*, 90
cohen_kappa, 76
cohen_kappa(), *100*, *131*, *132*, *216*
confint.ba *(*estimate*)*, 90
confint.ba_matrix *(*estimate*)*, 90
confint.ba_repeated *(*estimate*)*, 90
confint.ba_repeated_matrix *(*estimate*)*,
        90
confint.ccc *(*estimate*)*, 90
confint.ccc_ci *(*estimate*)*, 90
confint.ccc_glmm *(*estimate*)*, 90
confint.chatterjee_xi_scalar
        *(*estimate*)*, 90
confint.cia *(*estimate*)*, 90
confint.cia_ci *(*estimate*)*, 90
confint.cia_rm *(*estimate*)*, 90
confint.cohen_kappa *(*estimate*)*, 90
confint.corr_result *(*estimate*)*, 90
confint.gwet_ac *(*estimate*)*, 90
confint.icc *(*estimate*)*, 90
confint.icc_overall *(*estimate*)*, 90
confint.icc_rm_reml *(*estimate*)*, 90
confint.krippendorff_alpha *(*estimate*)*,
        90
confint.multirater_kappa *(*estimate*)*, 90
confint.partial_corr *(*estimate*)*, 90
confint.prob_agree *(*estimate*)*, 90
confint.rmcorr *(*estimate*)*, 90
confint.rmcorr_matrix *(*estimate*)*, 90
confint.summary.corr_result *(*estimate*)*,
        90
confint.summary.matrixCorr *(*estimate*)*,
        90
confint.weighted_kappa *(*estimate*)*, 90

dcor, 80
dcor(), *25*, *186*, *187*, *211*
deprecated-matrixCorr, 85
diag.bicor *(*bicor*)*, 26
distance_corr *(*deprecated-matrixCorr*)*,
        85

estimate, 90

ggplot2::theme(), *154*
gwet_ac, 95

gwet_ac(), *128*

hsic, 101

icc, 105
icc_rm_reml, 110
icc_rm_reml(), *74*

kendall_tau, 118
kendall_tau(), *211*
krippendorff_alpha, 124

multirater_kappa, 128
multirater_kappa(), *80*, *100*, *128*, *216*

partial_correlation
(deprecated-matrixCorr), 85
pbcor, 133
pbcor(), *181*, *187*, *198*, *211*, *221*
pcorr, 138
pcorr(), *211*
pearson_corr, *40*, 146, *192*
pearson_corr(), *211*
plot.ba, *10*
plot.ba (ba), 5
plot.ba_matrix (ba), 5
plot.ba_repeated (ba_rm), 11
plot.ba_repeated_matrix (ba_rm), 11
plot.bcdcor (bcdcor), 22
plot.bicor (bicor), 26
plot.biserial_corr (biserial), 33
plot.ccc, *41*, *62*
plot.ccc (ccc), 37
plot.cia (cia), 63
plot.cia_rm (cia_rm), 69
plot.cohen_kappa (cohen_kappa), 76
plot.corr_edge_list, 151
plot.corr_matrix, 152
plot.corr_sparse, 153
plot.dcor (dcor), 80
plot.gwet_ac (gwet_ac), 95
plot.hsic (hsic), 101
plot.kendall_matrix, *123*
plot.kendall_matrix (kendall_tau), 118
plot.krippendorff_alpha
(krippendorff_alpha), 124
plot.multirater_kappa
(multirater_kappa), 128
plot.partial_corr (pcorr), 138
plot.pbcor (pbcor), 133
plot.pearson_corr, *150*
plot.pearson_corr (pearson_corr), 146
plot.polychoric_corr (polychoric), 154
plot.polyserial_corr (polyserial), 159
plot.prob_agree, 153
plot.rmcorr (print.rmcorr), 166
plot.rmcorr_matrix
(print.rmcorr_matrix), 167
plot.robust_dcor (robust_dcor), 183
plot.schafer_corr (shrinkage_corr), 188
plot.shrinkage_corr, *192*
plot.shrinkage_corr (shrinkage_corr),
188
plot.skipped_corr (skipped_corr), 192
plot.spearman_rho, *203*
plot.spearman_rho (spearman_rho), 198
plot.tetrachoric_corr (tetrachoric), 207
plot.wincor (wincor), 217
polychoric, 154
polyserial, 159
print.ba, *10*
print.ba (ba), 5
print.ba_matrix (ba), 5
print.ba_repeated (ba_rm), 11
print.ba_repeated_matrix (ba_rm), 11
print.bcdcor (bcdcor), 22
print.bicor (bicor), 26
print.biserial_corr (biserial), 33
print.ccc, *41*, *62*
print.ccc (ccc), 37
print.ccc_ci, 163
print.chatterjee_xi (xi_corr), 222
print.chatterjee_xi_scalar (xi_corr),
222
print.cia (cia), 63
print.cia_ci (cia), 63
print.cia_rm (cia_rm), 69
print.cohen_kappa (cohen_kappa), 76
print.corr_edge_list, 164
print.data.frame, *204*
print.dcor (dcor), 80
print.gwet_ac (gwet_ac), 95
print.hsic (hsic), 101
print.icc (icc), 105
print.icc_overall (icc), 105
print.icc_rm_reml (icc_rm_reml), 110
print.kendall_matrix, *123*

print.kendall_matrix *(kendall_tau)*, 118
print.krippendorff_alpha *(krippendorff_alpha)*, 124
print.matrixCorr_ccc, 164
print.matrixCorr_ccc_ci, 165
print.multirater_kappa *(multirater_kappa)*, 128
print.partial_corr *(pcorr)*, 138
print.pbcor *(pbcor)*, 133
print.pearson_corr, *150*
print.pearson_corr *(pearson_corr)*, 146
print.polychoric_corr *(polychoric)*, 154
print.polyserial_corr *(polyserial)*, 159
print.rmcorr, 166
print.rmcorr_matrix, 167
print.robust_dcor *(robust_dcor)*, 183
print.schafer_corr *(shrinkage_corr)*, 188
print.shrinkage_corr, *192*
print.shrinkage_corr *(shrinkage_corr)*, 188
print.skipped_corr *(skipped_corr)*, 192
print.spearman_rho, *203*
print.spearman_rho *(spearman_rho)*, 198
print.summary.ba *(ba)*, 5
print.summary.ba_matrix *(ba)*, 5
print.summary.bcdcor *(bcdcor)*, 22
print.summary.bicor *(bicor)*, 26
print.summary.biserial_corr *(biserial)*, 33
print.summary.ccc *(ccc)*, 37
print.summary.ccc_rm_reml *(summary.ccc_rm_reml)*, 203
print.summary.chatterjee_xi_scalar *(xi_corr)*, 222
print.summary.cia *(cia)*, 63
print.summary.cia_rm *(cia_rm)*, 69
print.summary.cohen_kappa *(cohen_kappa)*, 76
print.summary.corr_result, 169
print.summary.dcor *(dcor)*, 80
print.summary.gwet_ac *(gwet_ac)*, 95
print.summary.hsic *(hsic)*, 101
print.summary.icc *(icc)*, 105
print.summary.icc_overall *(icc)*, 105
print.summary.icc_rm_reml *(icc_rm_reml)*, 110
print.summary.kendall_matrix *(kendall_tau)*, 118
print.summary.krippendorff_alpha *(krippendorff_alpha)*, 124
print.summary.latent_corr, 170
print.summary.matrixCorr, 171
print.summary.multirater_kappa *(multirater_kappa)*, 128
print.summary.partial_corr *(pcorr)*, 138
print.summary.pbcor *(pbcor)*, 133
print.summary.pearson_corr *(pearson_corr)*, 146
print.summary.polychoric_corr *(polychoric)*, 154
print.summary.polyserial_corr *(polyserial)*, 159
print.summary.rmcorr *(print.rmcorr)*, 166
print.summary.rmcorr_matrix *(print.rmcorr_matrix)*, 167
print.summary.skipped_corr *(skipped_corr)*, 192
print.summary.spearman_rho *(spearman_rho)*, 198
print.summary.tetrachoric_corr *(tetrachoric)*, 207
print.summary.wincor *(wincor)*, 217
print.tetrachoric_corr *(tetrachoric)*, 207
print.weighted_kappa *(weighted_kappa)*, 214
print.wincor *(wincor)*, 217
prob_agree, *9, 10*, *40, 41*, *68*, 172

rmcorr, 175
rmcorr(), *110*, *118*
rmcorr_weighted *(rmcorr)*, 175
robust_ccc, *41*, 179
robust_dcor, 183
robust_dcor(), *25*, *84*
robustbase::covMcd(), *180, 181*

schafer_corr *(shrinkage_corr)*, 188
shrinkage_corr, 188
shrinkage_corr(), *211*
skipped_corr, 192
skipped_corr(), *137*, *187*, *211*, *221*
skipped_corr_masks *(skipped_corr)*, 192
skipped_corr_masks(), *196*
spearman_rho, *40*, 198
spearman_rho(), *211*
summary.ba *(ba)*, 5

summary.ba_matrix *(ba)*, 5
summary.bcdcor *(bcdcor)*, 22
summary.bicor *(bicor)*, 26
summary.biserial_corr *(biserial)*, 33
summary.ccc *(ccc)*, 37
summary.ccc_rm_reml, 203
summary.chatterjee_xi_scalar *(xi_corr)*, 222
summary.cia *(cia)*, 63
summary.cia_rm *(cia_rm)*, 69
summary.cohen_kappa *(cohen_kappa)*, 76
summary.corr_edge_list, 205
summary.corr_matrix, 205
summary.corr_sparse, 206
summary.dcor *(dcor)*, 80
summary.gwet_ac *(gwet_ac)*, 95
summary.hsic *(hsic)*, 101
summary.icc *(icc)*, 105
summary.icc_overall *(icc)*, 105
summary.icc_rm_reml *(icc_rm_reml)*, 110
summary.kendall_matrix *(kendall_tau)*, 118
summary.krippendorff_alpha *(krippendorff_alpha)*, 124
summary.multirater_kappa *(multirater_kappa)*, 128
summary.partial_corr *(pcorr)*, 138
summary.pbcor *(pbcor)*, 133
summary.pearson_corr *(pearson_corr)*, 146
summary.polychoric_corr *(polychoric)*, 154
summary.polyserial_corr *(polyserial)*, 159
summary.rmcorr *(print.rmcorr)*, 166
summary.rmcorr_matrix *(print.rmcorr_matrix)*, 167
summary.robust_dcor *(robust_dcor)*, 183
summary.schafer_corr *(shrinkage_corr)*, 188
summary.shrinkage_corr *(shrinkage_corr)*, 188
summary.skipped_corr *(skipped_corr)*, 192
summary.spearman_rho *(spearman_rho)*, 198
summary.tetrachoric_corr *(tetrachoric)*, 207
summary.wincor *(wincor)*, 217

tetrachoric, 207
tidy *(estimate)*, 90
view_corr_shiny, 211
view_rmcorr_shiny, 212

weighted_kappa, 214
weighted_kappa(), *77*, *79*, *80*, *100*, *131*, *132*
wincor, 217
wincor(), *137*, *181*, *187*, *198*, *211*

xi_corr, 222