Package 'matrixCorr'

April 3, 2026

Type Package

Title Collection of Correlation and Association Estimators

Version 0.10.0

Description Compute correlation, association, and agreement measures for small to high-dimensional datasets through a consistent matrix-oriented interface. Supports classical correlations (Pearson, Spearman, Kendall), distance correlation, partial correlation with regularised estimators, shrinkage correlation for p >= n settings, robust correlations including biweight mid-correlation, percentage-bend, and skipped correlation, latent-variable methods for binary and ordinal data, repeated-measures correlation, and agreement analyses based on Bland-Altman methods and Lin's concordance correlation coefficient, including repeated-measures extensions. 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>.

License MIT + file LICENSE

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

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

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

Enhances plotly

RoxygenNote 7.3.3

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

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

Config/testthat/edition 3

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2026-04-03 17:30:02 UTC

Contents

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.

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,
  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,
  ...
)
```

Arguments

group1, group2	Numeric vectors of equal length.
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).
verbose	Logical; if TRUE, prints how many OpenMP threads are used.
x	A "ba" 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.

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.

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).

Value

An object of class `"ba"` (list) with elements:

- `means`, `diffs`: numeric vectors
- `groups`: data.frame used after NA removal
- `based.on`: integer, number of pairs used
- `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`

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`,`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)
plot(fit_ba)
```

`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 $\geq$ 3 methods. With exactly two methods it returns a single fitted BA object. With $\geq$ 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,
  loa_multiplier = 1.96,
  conf_level = 0.95,
  include_slope = FALSE,
  use_ar1 = FALSE,
  ar1_rho = NA_real_,
  max_iter = 200L,
  tol = 1e-06,
  verbose = FALSE
)

## 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.
`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.
conf_level	Confidence level for Wald confidence intervals for the reported bias and both LoA endpoints. Must lie in (0, 1). Default 0.95.
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.
verbose	Logical. If TRUE, print progress for the pairwise $\geq$ 3-method path. It has no material effect in the exactly-two-method path.
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 when present in the fitted object (two-method path) or when they can be reconstructed from x$data_long and x$mapping (pairwise path). 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.
- based.on: integer number of complete subject-time pairs used.
- 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`.
- `data_long`: stored long-format data frame used downstream for plotting and reconstruction. It uses canonical internal column names `.response`, `.subject`, `.method`, and `.time`.
- `mapping`: named list identifying those stored canonical column names for `response`, `subject`, `method`, and `time`.

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 $\texttt{loa\_lower}[j, k] = -\texttt{loa\_upper}[k, j]$ and $\texttt{loa\_upper}[j, k] = -\texttt{loa\_lower}[k, j]$.
- `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`: stored long-format data frame used downstream for plotting and reconstruction. It uses canonical internal column names `.response`, `.subject`, `.method`, and `.time`.
- `mapping`: named list identifying those stored canonical column names for `response`, `subject`, `method`, and `time`.

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)
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)

# 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)
```

---

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.

Usage

```
bicor(
  data,
  c_const = 9,
  max_p_outliers = 1,
  pearson_fallback = c("hybrid", "none", "all"),
  na_method = c("error", "pairwise"),
  mad_consistent = FALSE,
  w = NULL,
  sparse_threshold = NULL,
  n_threads = getOption("matrixCorr.threads", 1L)
)

diag.bicor(x, ...)

## S3 method for class 'bicor'
print(
  x,
  digits = 4,
  n = NULL,
  topn = NULL,
  max_vars = NULL,
  width = NULL,
  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,
  show_value = TRUE,
  na_fill = "grey90",
  ...
)

## S3 method for class 'bicor'
summary(
  object,
  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".
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.
na_method	One of "error" (default, fastest) or "pairwise". With "pairwise", each (j, k) correlation is computed on the intersection of non-missing rows for the pair.
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**).
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
x	An object of class 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").
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).
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.

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).

Basic properties. $\mathrm{bicor}(ax + b,\ cy + d) = \mathrm{sign}(ac)\,\mathrm{bicor}(x, y)$. 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. 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)

# 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, check_na = TRUE)

## 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,
  show_value = TRUE,
  ...
)

## S3 method for class 'biserial_corr'
summary(
  object,
  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.
check_na	Logical (default TRUE). If TRUE, missing values are rejected. If FALSE, pairwise complete cases are used.
x	An object of class 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".
...	Additional arguments passed to print().
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.
show_value	Logical; if TRUE (default), overlay numeric values on the heatmap tiles.
object	An object of class biserial_corr.

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.

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".

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(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)
print(bs, digits = 3)
summary(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, 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
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 $|\rho_c| \le |r| \le 1$, $\rho_c = r$ 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 (delta-method approximation). 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 values are not allowed; inputs must be numeric with at least two distinct non-missing values per column.

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 method1, method2, estimate and (optionally) lwr, upr.

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

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)
print(result_mvn)
summary(result_mvn)
plot(result_mvn)

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

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,
  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
)
```

Arguments

`data`	A data frame.
`response`	Character. Response variable name.
`rind`	Character. Subject ID variable name (random intercept).
`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).
`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.
`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`).

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, choose the printing mechanism, where TRUE uses message() (respects sink(), easily suppressible via suppressMessages()), whereas FALSE uses cat() to stdout. Default TRUE.

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 $R_i = \sigma^2_E\, C_i(\rho)$. 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 $L \ge 2$, the tridiagonal inverse has

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

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:

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

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

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

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

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 $d = (d_1^\top, \ldots, d_{n_t}^\top)^\top$ 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

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

  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 $a = (a_1, \ldots, a_{n_t})^\top$ with $a_t \geq 0$, 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

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

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

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

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

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

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

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

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 $(\sigma_A^2, \sigma_{A\times M}^2, \sigma_{A\times T}^2)$ covariance block from the subject-level empirical covariance, add the $\widehat{\mathrm{Var}}(\hat{\sigma}_E^2)$ 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 $(\hat{\beta}, \hat{G}, \hat{\sigma}_E^2)$. 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. On OpenBLAS and Apple's Accelerate this is handled automatically. On Intel MKL builds the guard is disabled by default, but you can also 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", verbose = TRUE)
summary(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)

ccc_rm_reml(dat_ar4,
             response = "y", rind = "id", method = "method", time = "time",
             ar = "ar1", ar_rho = 0.6, verbose = TRUE)


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

Arguments

`data`	A data frame containing the repeated-measures dataset.
`response`	Character. Name of the numeric outcome column.
`method`	Character. Name of the method column (factor with L $\geq$ 2 levels).
`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.
`time`	Character or NULL. Name of the time/repetition column. If NULL, one time point is assumed.
`Dmat`	Optional numeric weight matrix (T $\times$ T) for timepoints. Defaults to identity.
`delta`	Numeric. Power exponent used in the distance computations between method trajectories across time points. This controls the contribution of differences between measurements: • `delta = 1` (default) uses **absolute differences**. • `delta = 2` uses **squared differences**, more sensitive to larger deviations. • `delta = 0` reduces to a **binary distance** (presence/absence of disagreement), analogous to a repeated-measures version of the kappa statistic. The choice of `delta` should reflect the penalty you want to assign to measurement disagreement.
`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).

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", method = "method",
                             subject = "subject", time = "time")
print(ccc1)
summary(ccc1)
plot(ccc1)

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

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

#-------------------------------------------------------------------------
# Choosing delta based on distance sensitivity
#-------------------------------------------------------------------------
# Absolute distance (L1 norm) - robust
ccc_rm_ustat(df, response = "y", method = "method",
                    subject = "subject", time = "time", delta = 1)

# Squared distance (L2 norm) - amplifies large deviations
ccc_rm_ustat(df, response = "y", method = "method",
                    subject = "subject", time = "time", delta = 2)

# Presence/absence of disagreement (like kappa)
ccc_rm_ustat(df, response = "y", method = "method",
```

```
                    subject = "subject", time = "time", delta = 0)
```

---

dcor *Pairwise Distance Correlation (dCor)*

Description

Computes pairwise distance correlations for the numeric columns of a matrix or data frame using a high-performance 'C++' backend. Distance correlation detects general dependence, including non-linear relationships.

Usage

```
dcor(data, check_na = 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 = "Distance correlation heatmap",
  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,
```

```
  ...
)
```

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.
check_na	Logical (default TRUE). When TRUE, inputs must be free of NA/NaN/Inf. Set to FALSE only if you have already handled missingness upstream.
x	An object of class 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".
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
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

Let $x \in \mathbb{R}^n$ and $D^{(x)}$ be the pairwise distance matrix with zero diagonal: $D^{(x)}_{ii} = 0$, $D^{(x)}_{ij} = |x_i - x_j|$ for $i \neq j$. Define row sums $r^{(x)}_i = \sum_{k \neq i} D^{(x)}_{ik}$ and grand sum $S^{(x)} = \sum_{i \neq k} D^{(x)}_{ik}$. The U-centred matrix is

$$A^{(x)}_{ij} = \begin{cases} D^{(x)}_{ij} - \dfrac{r^{(x)}_i + r^{(x)}_j}{n-2} + \dfrac{S^{(x)}}{(n-1)(n-2)}, & i \neq j, \\ 0, & i = j\ . \end{cases}$$

For two variables x, y, the unbiased distance covariance and variances are

$$\widehat{\mathrm{dCov}}^2_u(x,y) = \frac{2}{n(n-3)} \sum_{i<j} A^{(x)}_{ij} A^{(y)}_{ij} \ = \ \frac{1}{n(n-3)} \sum_{i \neq j} A^{(x)}_{ij} A^{(y)}_{ij},$$

with $\widehat{\mathrm{dVar}}^2_u(x)$ defined analogously from $A^{(x)}$. The unbiased distance correlation is

$$\widehat{\mathrm{dCor}}_u(x,y) = \frac{\widehat{\mathrm{dCov}}_u(x,y)}{\sqrt{\widehat{\mathrm{dVar}}_u(x)\,\widehat{\mathrm{dVar}}_u(y)}} \in [0,1].$$

Computation. All heavy lifting (distance matrices, U-centering, and unbiased scaling) is implemented in C++ (`ustat_dcor_matrix_cpp`), so the R wrapper only validates/coerces the input. OpenMP parallelises the upper-triangular loops. The implementation includes a Huo-Szekely style univariate $O(n \log n)$ dispatch for pairwise terms. We also have an exact unbiased $O(n^2)$ fallback retained for robustness in small-sample or non-finite-path cases; no external dependencies are used.

Value

A symmetric numeric matrix where the `(i, j)` entry is the unbiased distance correlation between the `i`-th and `j`-th numeric columns. The object has class `dcor` with attributes `method = "distance_correlation"`, `description`, and `package = "matrixCorr"`.

Invisibly returns `x`.

A `ggplot` object representing the heatmap.

Note

Requires $n \geq 4$. Columns with (near) zero unbiased distance variance yield `NA` in their row/column. Typical per-pair cost uses the $O(n \log n)$ fast path, with $O(n^2)$ fallback when needed.

Author(s)

Thiago de paula Oliveira

References

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

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

Examples

```
##Independent variables -> dCor ~ 0
set.seed(1)
X <- cbind(a = rnorm(200), b = rnorm(200))
D <- dcor(X)
print(D, digits = 3)
summary(D)

## Non-linear dependence: Pearson ~ 0, but unbiased dCor > 0
set.seed(42)
n <- 200
x <- rnorm(n)
y <- x^2 + rnorm(n, sd = 0.2)
XY <- cbind(x = x, y = y)
D2 <- dcor(XY)
# Compare Pearson vs unbiased distance correlation
round(c(pearson = cor(XY)[1, 2], dcor = D2["x", "y"]), 3)
summary(D2)
plot(D2, title = "Unbiased distance correlation (non-linear example)")
```

```
## Small AR(1) multivariate normal example
set.seed(7)
p <- 5; n <- 150; rho <- 0.6
Sigma <- rho^abs(outer(seq_len(p), seq_len(p), "-"))
X3 <- MASS::mvrnorm(n, mu = rep(0, p), Sigma = Sigma)
colnames(X3) <- paste0("V", seq_len(p))
D3 <- dcor(X3)
print(D3[1:3, 1:3], digits = 2)

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

`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,
  verbose = FALSE
)

bland_altman_repeated(
  data = NULL,
  response,
  subject,
  method,
  time,
  two = 1.96,
  conf_level = 0.95,
  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"),
  mad_consistent = FALSE,
  w = NULL,
  sparse_threshold = NULL,
  n_threads = getOption("matrixCorr.threads", 1L)
)

distance_corr(data, check_na = TRUE)

partial_correlation(
  data,
  method = c("oas", "ridge", "sample"),
  lambda = 0.001,
  return_cov_precision = FALSE,
  ci = FALSE,
  conf_level = 0.95
)

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.
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 used by `bicor()`.

`mad_consistent` Logical; if `TRUE`, uses the consistency-corrected MAD.

`w` Optional vector of case weights.

`sparse_threshold` Optional threshold controlling sparse output.

`n_threads` Integer number of OpenMP threads.

`check_na` Logical validation flag used by `dcor()`.

`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 for `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.

`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,
  check_na = TRUE,
  ci = FALSE,
  conf_level = 0.95,
  ci_method = c("fieller", "if_el", "brown_benedetti")
)

## 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.
check_na	Logical (default TRUE). If TRUE, inputs must be free of missing/undefined values. Use FALSE only when missingness has already been handled upstream.

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".
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".
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
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 check_na = FALSE, 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 not allowed when check_na = TRUE. Columns with fewer than two 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)

# 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)
}
```

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,
  beta = 0.2,
  na_method = c("error", "pairwise"),
  n_threads = getOption("matrixCorr.threads", 1L)
)

## 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,
  show_ci = NULL,
  ...
)
```

Arguments

data	A numeric matrix or data frame containing numeric columns.
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.
na_method	One of "error" (default) or "pairwise". With "pairwise", each correlation is computed on the overlapping complete rows for the column pair.
n_threads	Integer ≥ 1. Kept for API consistency with the other robust correlation wrappers. It is currently validated but not used by the exact percentage-bend implementation.
x	An object of class 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.

Details

For a column $x = (x_i)_{i=1}^n$, let $m = \mathrm{med}(x)$ and let $\omega_\beta(x)$ be the $\lfloor (1-\beta)n \rfloor$-th order statistic of $|x_i - m|$. The constant beta determines the cutoff $\omega_\beta(x)$ used to standardise deviations from the median. As beta increases, the selected cutoff becomes smaller, so a larger fraction of observations is truncated to the bounds -1 and 1. This makes the correlation more resistant to marginal outliers. The one-step percentage-bend location is

$$\hat{\theta}_{pb}(x) = \frac{\sum_{i:|\psi_i| \leq 1} x_i + \omega_\beta(x)(i_2 - i_1)}{n - i_1 - i_2}, \qquad \psi_i = \frac{x_i - m}{\omega_\beta(x)},$$

where $i_1 = \sum \mathbf{1}(\psi_i < -1)$ and $i_2 = \sum \mathbf{1}(\psi_i > 1)$.

The bent scores are

$$a_i = \max\{-1, \min(1, (x_i - \hat{\theta}_{pb}(x))/\omega_\beta(x))\},$$

and likewise b_i for a second variable y. The percentage bend correlation is

$$r_{pb}(x, y) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}.$$

When `na_method = "error"`, bent scores are computed once per column and the matrix is formed from their cross-products. When `na_method = "pairwise"`, each pair is recomputed on its complete-case overlap, which can break positive semidefiniteness as with pairwise Pearson correlation.

Value

A symmetric correlation matrix with class `pbcor` and attributes `method = "percentage_bend_correlation"`, `description`, and `package = "matrixCorr"`.

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)
plot(R)

# 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"),
  lambda = 0.001,
  return_cov_precision = FALSE,
  return_p_value = FALSE,
  ci = FALSE,
  conf_level = 0.95
)

## 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".
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.
return_cov_precision	Logical; if TRUE, also return the covariance (cov) and precision (precision) matrices used to form the partial correlations. Default to FALSE
return_p_value	Logical; if TRUE, also return the matrix of two-sided p-values for testing whether each sample partial correlation is zero. This option is available only for method = "sample" and requires $n > p$. Default to FALSE.
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.
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 $1 - \lvert pcor \rvert$. 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

$$t_{ij} = \mathrm{pcor}_{ij}\sqrt{\frac{n-p}{1-\mathrm{pcor}_{ij}^2}}$$

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 $z = \mathrm{atanh}(r_{xy\cdot Z})$ 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

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)

# 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(pc$pcor[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)
gfit_2 <- pcorr(X_g, method = "glasso", lambda = 0.08,
                return_cov_precision = 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)
pc_ridge <-
 pcorr(X_hd, method = "ridge", lambda = 1e-2,
                     return_cov_precision = TRUE)
pc_samp  <-
 pcorr(X_hd, method = "sample", return_cov_precision = TRUE)
```

```
pc_glasso <-
 pcorr(X_hd, method = "glasso", lambda = 5e-3,
                     return_cov_precision = 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, check_na = TRUE, ci = FALSE, conf_level = 0.95)

## 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.
check_na	Logical (default TRUE). If TRUE, inputs must be free of NA/NaN/Inf. Set to FALSE only when the caller already handled missingness.
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.

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".
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
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 check_na = FALSE, 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}:

$$z_{ij} = \operatorname{atanh}(r_{ij}), \qquad \operatorname{SE}(z_{ij}) = \frac{1}{\sqrt{n_{ij} - 3}}.$$

With $z_{1-\alpha/2} = \Phi^{-1}(1 - \alpha/2)$, the confidence limits are

$$\tanh\big(z_{ij} - z_{1-\alpha/2}\operatorname{SE}(z_{ij})\big) \quad \text{and} \quad \tanh\big(z_{ij} + z_{1-\alpha/2}\operatorname{SE}(z_{ij})\big).$$

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

Missing values are not allowed when check_na = TRUE. Columns with fewer than two observations are excluded.

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)
plot(pr)

## 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)
}
```

`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, correct = 0.5, check_na = 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,
  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.
correct	Non-negative continuity correction added to zero-count cells. Default is `0.5`.
check_na	Logical (default `TRUE`). If `TRUE`, missing values are rejected. If `FALSE`, pairwise complete cases are used.
x	An object of class `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".
...	Additional arguments passed to print().
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.

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.

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`.

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)
plot(pc)

# 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, check_na = TRUE)

## 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,
  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.
check_na	Logical (default TRUE). If TRUE, missing values are rejected. If FALSE, pairwise complete cases are used.
x	An object of class 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".
...	Additional arguments passed to print().
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.

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++.

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"`.

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)
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.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_matrix` *Summary Method for Correlation Matrices*

Description

Prints compact summary statistics returned by matrix-style `summary()` methods in **matrixCorr**.

Usage

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

Arguments

`x`	An object of class `summary_corr_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")`.
`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`.

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

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`.
`conf_level`	Confidence level used for Wald confidence intervals on the repeated-measures correlation (default `0.95`).
`check_na`	Logical (default `TRUE`). If `TRUE`, missing values in the selected response columns or `subject` are rejected. If `FALSE`, each pairwise repeated-measures correlation uses pairwise complete cases and drops subjects with fewer than two complete pairs for that specific contrast.
`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`, matrix-mode outputs retain a compact copy of the numeric response matrix plus encoded subject identifiers so that `view_rmcorr_shiny()` can rebuild pairwise scatterplots from the returned object alone.
`verbose`	Logical; if `TRUE`, prints a short note about the thread count used in matrix mode.

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.

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:

- `r`; repeated-measures correlation estimate.
- `p_value`; two-sided p-value for the common within-subject slope.
- `conf_int`; confidence interval for `r`.
- `slope`; common within-subject slope.
- `df`; residual degrees of freedom $N - S - 1$.
- `based.on`; 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.
- `intercepts`; subject-specific fitted intercepts for the common slope.
- `data_long`; filtered long data used for fitting and plotting.

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

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")
print(fit_xy)
summary(fit_xy)
plot(fit_xy)

fit_mat <- rmcorr(dat, response = c("x", "y", "z"), subject = "subject")
print(fit_mat, digits = 3)
summary(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)
}
```

`schafer_corr` *Schafer-Strimmer shrinkage correlation*

Description

Computes a Schafer-Strimmer shrinkage correlation matrix for numeric data using a high-performance 'C++' backend. This stabilises Pearson correlation estimates by shrinking off-diagonal entries towards zero.

Usage

```
schafer_corr(data)
```

```
## 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 '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 '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.
x	An object of class 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 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 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.schafer_corr, plot.schafer_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 <- schafer_corr(X)
print(Rshr, digits = 2, n = 6, max_vars = 6)
summary(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"),
  stand = TRUE,
  outlier_rule = c("idealf", "mad"),
  cutoff = sqrt(stats::qchisq(0.975, df = 2)),
  n_threads = getOption("matrixCorr.threads", 1L),
  return_masks = FALSE,
  ci = FALSE,
  p_value = FALSE,
  conf_level = 0.95,
  n_boot = 2000L,
  p_adjust = c("none", "hochberg", "ecp"),
  fwe_level = 0.05,
  n_mc = 1000L,
  seed = NULL
)

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	One of "error" (default) or "pairwise". 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. With "pairwise", each variable pair is computed on its own overlap of finite rows. This is more permissive for incomplete data, but different pairs can be based on different effective samples and different skipped-row sets, so the resulting matrix is less directly comparable across entries.
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_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`.
`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_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.
`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:

$$r_{\mathrm{skip}}(x, y) \;=\; \mathrm{cor}(x_{\mathcal{K}}, y_{\mathcal{K}})\,,$$

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.

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"`). 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)
Ri$ci

# 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, check_na = TRUE, ci = FALSE, conf_level = 0.95)

## 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.
check_na	Logical (default TRUE). If TRUE, the input is required to be free of NA/NaN/Inf. Set to FALSE only when the caller already handled missingness.
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.
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".
...	Additional arguments passed to ggplot2::theme() or other ggplot2 layers.
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

$$\widehat{\rho}_S \;=\; D^{-1/2} S_R D^{-1/2}, \qquad D \;=\; \operatorname{diag}(\operatorname{diag}(S_R)).$$

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, $\operatorname{Var}(R_{\cdot j}) = (n^2-1)/12$; 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 `check_na = FALSE`, 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 not allowed when `check_na = TRUE`. Columns with fewer than two 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)

## 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: method1, method2, estimate, and optionally lwr, upr, as well as variance component estimates: sigma2_subject, sigma2_subject_method, sigma2_subject_time, sigma2_error, sigma2_extra, SB, se_ccc.

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, correct = 0.5, check_na = 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,
  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.
correct	Non-negative continuity correction added to zero-count cells. Default is 0.5.
check_na	Logical (default TRUE). If TRUE, missing values are rejected. If FALSE, pairwise complete cases are used.
x	An object of class 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".
...	Additional arguments passed to print().
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.

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` behaviour used in several psychometric 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`.

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`.

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)
plot(tc)

# 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()`, or `schafer_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)
}
```

`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,
  tr = 0.2,
  na_method = c("error", "pairwise"),
  n_threads = getOption("matrixCorr.threads", 1L)
)

## 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,
  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.
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.
na_method	One of "error" (default) or "pairwise".
n_threads	Integer ≥ 1. Number of OpenMP threads. Defaults to getOption("matrixCorr.threads", 1L).
x	An object of class 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.

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.

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.

Value

A symmetric correlation matrix with class `wincor` and attributes `method = "winsorized_correlation"`, `description`, and `package = "matrixCorr"`.

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)
plot(R)

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

Index