Package ‘matrixCorr’

March 21, 2026

Type Package

Title Collection of Correlation and Association Estimators

Version 0.8.5

Description Compute correlation and other association matrices from small to high-dimensional datasets with relative simple functions and sensible defaults. Includes options for shrinkage and robustness to improve results in noisy or high-dimensional settings (p >= n), plus convenient print/plot methods for inspection. Implemented with optimised C++ backends using BLAS/OpenMP and memory-aware symmetric updates. Works with base matrices and data frames, returning standard R objects via a consistent S3 interface. Useful across genomics, agriculture, and machine-learning workflows. Supports Pearson, Spearman, Kendall, distance correlation, partial correlation, and robust biweight mid-correlation; Bland–Altman analyses and Lin's concordance correlation coefficient (including repeated-measures extensions). Methods based on Ledoit and Wolf (2004) <doi:10.1016/S0047-259X(03)00096-4>; Schäfer and Strimmer (2005) <doi:10.2202/1544-6115.1175>; Lin (1989) <doi:10.2307/2532051>.

License MIT + file LICENSE

Encoding UTF-8

LinkingTo cpp11, Rcpp, RcppArmadillo

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

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

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 2026-03-21 01:20:02 UTC

Contents

`biweight_mid_corr` *Biweight mid-correlation (bicor)*

Description

Use biweight mid-correlatio when you want a Pearson-like measure that is robust to outliers and heavy-tailed noise. Bicor down-weights extreme observations via Tukey's biweight while preserving location/scale invariance, making it well suited to high-throughput data (e.g., gene expression) where occasional gross errors or platform artefacts occur. Prefer Spearman/Kendall for purely ordinal structure or strongly non-linear monotone relations.

Prints a matrix with a compact header, optional truncation for large matrices, and a small summary of off-diagonal values.

Produces a **ggplot2** heatmap of the biweight mid-correlation matrix. Optionally reorders variables via hierarchical clustering on $1 - r_{\text{bicor}}$, and can show only a triangle.

Usage

```
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)
)

diag.biweight_mid_corr(x, ...)

## S3 method for class 'biweight_mid_corr'
print(
  x,
  digits = 4,
  max_rows = NULL,
  max_cols = NULL,
  width = getOption("width", 80L),
  na_print = "NA",
  ...
)

## S3 method for class 'biweight_mid_corr'
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,
  na_fill = "grey90",
  ...
)
```

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 `biweight_mid_corr`.
`...`	Additional arguments passed to `ggplot2::theme()` or other layers.
`digits`	Integer; number of decimal places used for the matrix.
`max_rows`	Optional integer; maximum number of rows to display (default shows all).
`max_cols`	Optional integer; maximum number of columns to display (default shows all).
`width`	Integer; target console width for wrapping header text.
`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 re-ordered 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).
`na_fill`	Fill colour for `NA` cells. Default `"grey90"`.

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 $q_L = Q_x(\mathrm{max\_p\_outliers})$ and $q_U = Q_x(1 - \mathrm{max\_p\_outliers})$ 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 `biweight_mid_corr` (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 <- biweight_mid_corr(X, c_const = 9, max_p_outliers = 1,
                      pearson_fallback = "hybrid")
print(attr(R, "method"))

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

`bland_altman` *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

```
bland_altman(
  group1,
  group2,
  two = 1.96,
  mode = 1L,
  conf_level = 0.95,
  verbose = FALSE
)

## S3 method for class 'ba'
print(x, digits = 3, ci_digits = 3, ...)

## 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,
  ...
)
```

Arguments

`group1, group2`	Numeric vectors of equal length.
`two`	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).
`...`	Passed to `ggplot2::theme()` (ggplot path) or `plot()`.
`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.

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 `two`) 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
- two, 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_pairwise_u_stat, ccc_lmm_reml

Examples

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

---

bland_altman_repeated *Bland-Altman for repeated measurements*

Description

Repeated-measures Bland-Altman (BA) for method comparison based on a mixed-effects model fitted to **subject-time matched paired differences**. A subject-specific random intercept accounts for clustering, and (optionally) an AR(1) process captures serial correlation across replicates within subject. The function returns bias (mean difference), limits of agreement (LoA), confidence intervals, and variance components, for either two methods or all pairwise contrasts when $\geq$3 methods are supplied.

Required columns / vectors

- `response`: numeric measurements.
- `subject`: subject identifier (integer/factor/numeric).
- `method`: method label with $\geq$2 levels (factor/character/integer).
- `time`: replicate index used to form pairs; only records where both methods are present for the same `subject` and `time` contribute to a given pairwise BA (rows with missing components are dropped before fitting each pair).

Usage

```
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
)

## S3 method for class 'ba_repeated'
print(x, digits = 3, ci_digits = 3, ...)

## S3 method for class 'ba_repeated_matrix'
print(x, digits = 3, ci_digits = 3, 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,
  ...
)

## 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,
  ...
)
```

Arguments

`data`	Optional `data.frame`/`data.table` containing required columns.
`response`	Numeric vector (stacked outcomes) **or** a single character string giving the column name in `data`.
`subject`	Subject ID (integer/factor/numeric) **or** a single character string giving the column name in `data`.
`method`	Method label (factor/character/integer; N >= 2 levels) **or** a single character string giving the column name in `data`.
`time`	Integer/numeric replicate/time index (pairs within subject) **or** a single character string giving the column name in `data`.
`two`	Positive scalar; LoA multiple of SD (default 1.96).
`conf_level`	Confidence level for CIs (default 0.95).
`include_slope`	Logical. If `TRUE`, the model includes the pair mean as a fixed effect and estimates a proportional-bias slope. Look at details for deeper information. We recommend fit both with and without the slope. If β_1 is materially non-zero over a wide level range, consider a scale transformation (e.g., log or percent-logit) and re-fit without the slope.
`use_ar1`	Logical; AR(1) within-subject residual correlation.
`ar1_rho`	AR(1) parameter (|rho|<1).

max_iter, tol	EM control for the backend (defaults 200, 1e-6).
verbose	Logical; print brief progress.
x	A "ba_repeated_matrix" object.
digits	Number of digits for estimates (default 3).
ci_digits	Number of digits for CI bounds (default 3).
...	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.
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

The function implements a repeated-measures Bland–Altman (BA) analysis for two or more methods using a linear mixed model fitted to *subject–time matched paired differences*. For any selected pair of methods (a, b), let

$$d_{it} = y_{itb} - y_{ita}, \qquad m_{it} = \tfrac{1}{2}(y_{itb} + y_{ita}),$$

where y_{itm} denotes the observed value from method $m \in \{a, b\}$ on subject i at replicate/time t; d_{it} is the paired difference (method b minus method a); and m_{it} is the corresponding pair mean. Here $i = 1, \ldots, S$ indexes subjects and t indexes replicates/time within subject. Only records with both methods present at the same `subject` and `time` contribute to that pair.

The fitted per-pair model is

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

with random intercept $u_i \sim \mathcal{N}(0, \sigma_u^2)$ and within-subject residual vector ε_i having covariance $\mathrm{Cov}(\varepsilon_i) = \sigma_e^2\, \mathbf{R}_i$. When `use_ar1 = FALSE`, $\mathbf{R}_i = \mathbf{I}_{n_i}$ (i.i.d.). When `use_ar1 = TRUE`, $\mathbf{R}_i = \mathbf{C}_i^{-1}$ and $\mathbf{C}_i$ encodes an AR(1) *precision* structure over time (see below). Setting `include_slope = FALSE` drops the regressor m_{it} (i.e., $\beta_1 = 0$).

AR(1) within-subject correlation: For each subject, observations are first ordered by `time`. Contiguous blocks satisfy $t_{k+1} = t_k + 1$; non-contiguous gaps and any negative/NA times are treated as singletons (i.i.d.).

For a contiguous block of length L and AR(1) parameter ρ (clipped to $(-0.999, 0.999)$), the blockwise *precision* matrix $\mathbf{C}$ has entries

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

The full $\mathbf{C}_i$ is block-diagonal over contiguous segments, with a small ridge added to the diagonal for numerical stability. Residual covariance is $\sigma_e^2 \mathbf{R}_i = \sigma_e^2 \mathbf{C}_i^{-1}$.

If `use_ar1 = TRUE` and `ar1_rho` is `NA`, ρ is estimated in two passes (i) fit the i.i.d. model; (ii) compute detrended residuals within each contiguous block (remove block-specific intercept and linear time), form lag-1 correlations, apply a small-sample bias adjustment $(1-\rho^2)/L$, pool with Fisher's z (weights $\approx L - 3$), and refit with the pooled $\hat{\rho}$.

Estimation by stabilised EM/GLS: Let $\mathbf{X}_i$ be the fixed-effects design for subject i (intercept, and optionally the pair mean). Internally, the pair mean regressor is centred and scaled before fitting to stabilise the slope; estimates are back-transformed to the original units afterwards.

Given current (σ_u^2, σ_e^2), the marginal precision of d_i integrates u_i via a Woodbury/Sherman-Morrison identity:

$$\mathbf{V}_i^{-1} \;=\; \sigma_e^{-2}\mathbf{C}_i \;-\; \sigma_e^{-2}\mathbf{C}_i\mathbf{1}\left(\sigma_u^{-2} + \sigma_e^{-2}\mathbf{1}^\top\mathbf{C}_i\mathbf{1}\right)^{-1}\mathbf{1}^\top\mathbf{C}_i\sigma_e^{-2}.$$

The GLS update is

$$\hat\beta \;=\; \Big(\sum_i \mathbf{X}_i^\top\mathbf{V}_i^{-1}\mathbf{X}_i\Big)^{-1}\Big(\sum_i \mathbf{X}_i^\top\mathbf{V}_i^{-1}d_i\Big).$$

Writing $r_i = d_i - \mathbf{X}_i\hat\beta$, the E-step gives the BLUP of the random intercept and its conditional second moment:

$$M_i \;=\; \sigma_u^{-2} + \sigma_e^{-2}\mathbf{1}^\top\mathbf{C}_i\mathbf{1}, \qquad \tilde u_i \;=\; M_i^{-1}\,\sigma_e^{-2}\mathbf{1}^\top\mathbf{C}_i r_i, \qquad \mathbb{E}[u_i^2 \mid y] \;=\; \tilde u_i^2 + M_i^{-1}.$$

The M-step updates the variance components by

$$\hat\sigma_u^2 \;=\; \frac{1}{m}\sum_i \mathbb{E}[u_i^2 \mid y], \qquad \hat\sigma_e^2 \;=\; \frac{1}{n}\sum_i \Big\{(r_i - \mathbf{1}\tilde u_i)^\top\mathbf{C}_i(r_i - \mathbf{1}\tilde u_i) + M_i^{-1}\,\mathbf{1}^\top\mathbf{C}_i\mathbf{1}\Big\}.$$

The algorithm employs positive-definite inverses with ridge fallback, trust-region damping of variance updates (ratio-bounded), and clipping of parameters to prevent degeneracy.

Point estimates reported: The reported BA *bias* is the *subject-equal* mean of the paired differences,

$$\mu_0 \;=\; \frac{1}{m}\sum_{i=1}^{m}\bar d_i, \qquad \bar d_i = \frac{1}{n_i}\sum_{t=1}^{n_i} d_{it},$$

which is robust to unbalanced replicate counts (n_i) and coincides with the GLS intercept when subjects contribute equally. If `include_slope = TRUE`, the proportional-bias slope is estimated against the pair mean m_{it}; internally m_{it} is centred and scaled and the slope is returned on the original scale.

Proportional bias slope (`include_slope`): When `include_slope = TRUE`, the model augments the mean difference with the pair mean $m_{it} = \frac{1}{2}(y_{itb} + y_{ita})$:

$$d_{it} = \beta_0 + \beta_1 m_{it} + u_i + \varepsilon_{it}.$$

Internally m_{it} is centred and scaled for numerical stability; the reported `beta_slope` and intercept are back-transformed to the original scale.

$\beta_1 \neq 0$ indicates level-dependent (proportional) bias, where the difference between methods changes with the overall measurement level. The Bland–Altman limits of agreement produced by this function remain the conventional, *horizontal* bands centred at the subject-equal bias μ_0; they are not regression-adjusted LoA.

On an appropriate scale and when the two methods have comparable within-subject variances, the null expectation is $\beta_1 \approx 0$. However, because m_{it} contains measurement error from both methods, unequal precisions can produce a spurious slope even if there is no true proportional bias. Under a simple no-bias data-generating model $y_{itm} = \theta_{it} + c_m + e_{itm}$ with independent errors of variances σ_a^2, σ_b^2, the expected OLS slope is approximately

$$\mathbb{E}[\hat\beta_1] \;\approx\; \frac{\frac{1}{2}(\sigma_b^2 - \sigma_a^2)}{\mathrm{Var}(\theta_{it}) + \frac{1}{4}(\sigma_a^2 + \sigma_b^2)},$$

showing zero expectation when $\sigma_a^2 = \sigma_b^2$ and attenuation as the level variability $\mathrm{Var}(\theta_{it})$ increases.

Limits of Agreement (LoA): A *single new paired measurement* for a random subject has variance

$$\mathrm{Var}(d^{\star}) \;=\; \sigma_u^2 + \sigma_e^2,$$

so the LoA are

$$\mathrm{LoA} \;=\; \mu_0 \;\pm\; \texttt{two}\,\sqrt{\sigma_u^2 + \sigma_e^2}.$$

The argument `two` is the SD multiplier (default $z_{1-\alpha/2}$ implied by `conf_level`, e.g., 1.96 at 95%).

Wald/Delta confidence intervals: CIs for *bias* and each LoA use a delta method around the EM estimates. The standard error of μ_0 is based on the dispersion of subject means $\mathrm{Var}(\mu_0) \approx \mathrm{Var}(\bar{d}_i)/m$. For $\mathrm{sd}_{\mathrm{LoA}} = \sqrt{V}$ with $V = \sigma_u^2 + \sigma_e^2$, we can define

$$\mathrm{Var}(\mathrm{sd}) \;\approx\; \frac{\mathrm{Var}(V)}{4V}, \quad \mathrm{Var}(V) \;\approx\; \mathrm{Var}(\hat{\sigma}_u^2) + \mathrm{Var}(\hat{\sigma}_e^2) + 2\,\mathrm{Cov}(\hat{\sigma}_u^2, \hat{\sigma}_e^2).$$

The per-subject contributions used to approximate these terms are: $A_i = \mathbb{E}[u_i^2 \mid y]$ and $B_i = \{(r_i - \mathbf{1}\tilde{u}_i)^\top \mathbf{C}_i (r_i - \mathbf{1}\tilde{u}_i) + M_i^{-1} \mathbf{1}^\top \mathbf{C}_i \mathbf{1}\}/n_i$. Empirical variances of A_i and B_i (with replicate-size weighting for B_i) and their covariance yield $\mathrm{Var}(\hat{\sigma}_u^2)$, $\mathrm{Var}(\hat{\sigma}_e^2)$ and $\mathrm{Cov}(\hat{\sigma}_u^2, \hat{\sigma}_e^2)$. For a LoA bound $L_\pm = \mu_0 \pm \texttt{two} \cdot \mathrm{sd}$, the working variance is

$$\mathrm{Var}(L_\pm) \;\approx\; \mathrm{Var}(\mu_0) + \texttt{two}^2\,\mathrm{Var}(\mathrm{sd}),$$

and Wald CIs use the normal quantile at `conf_level`. These are large-sample approximations; with very small m they may be conservative.

Three or more methods: When ≥ 3 methods are supplied, the routine performs the above fit for each unordered pair (j, k) and reassembles matrices. Specifically,

- *Orientation* is defined to be *row minus column*, i.e., `bias`$[j, k]$ estimates $\mathbb{E}(y_k - y_j)$.
- `bias` is antisymmetric ($b_{jk} = -b_{kj}$); `sd_loa` and `width` are symmetric; LoA obey `loa_lower`$[j, k] = -$`loa_upper`$[k, j]$.
- `n`$[j, k]$ is the number of subject-time pairs used for that contrast (complete cases where both methods are present).

Missingness, time irregularity, and safeguards: Pairs are formed only where both methods are observed at the same `subject` and `time`; other records do not influence that pair. AR(1) structure is applied only over contiguous time sequences within subject; gaps break contiguity and revert those positions to i.i.d. Numerically, inverses use a positive-definite solver with adaptive ridge and pseudo-inverse fallback; variance updates are clamped and ratio-damped; ρ is clipped to $(-0.999, 0.999)$.

Value

Either a `"ba_repeated"` object (exactly two methods) or a `"ba_repeated_matrix"` object (pairwise results when $\geq$3 methods).

If `"ba_repeated_matrix"` **(N$\geq$3 methods)**, outputs are:

- `bias` $(m \times m)$; estimated mean difference (row - column). Diagonal is `NA`.
- `sd_loa` $(m \times m)$; estimated SD of a *single new* paired difference for the (row, column) methods, accounting for the random subject intercept and (if enabled) AR(1) correlation.

- `loa_lower`, `loa_upper` $(m \times m)$; limits of agreement for a single measurement pair, computed as bias $\pm$ two $\times$ sd_loa. Signs follow the row - column convention (e.g., `loa_lower[j,i] = -loa_upper[i,j]`).
- `width` $(m \times m)$; LoA width, `loa_upper - loa_lower` (= `2 * two * sd_loa`).
- `n` $(m \times m)$; number of subject-time pairs used in each pairwise BA (complete cases where both methods are present).
- **CI matrices at nominal** `conf_level` (delta method):
  - `mean_ci_low`, `mean_ci_high`; CI for the bias.
  - `loa_lower_ci_low`, `loa_lower_ci_high`; CI for the lower LoA.
  - `loa_upper_ci_low`, `loa_upper_ci_high`; CI for the upper LoA.
- `slope` ($m \times m$; optional); proportional-bias slope (difference vs pair mean) when `include_slope = TRUE`.
- `sigma2_subject` $(m \times m)$; estimated variance of the subject-level random intercept (on differences).
- `sigma2_resid` $(m \times m)$; estimated residual variance of a single difference after accounting for the random intercept (and AR(1), if used).
- `use_ar1` *(scalar logical)*; whether AR(1) modeling was requested.
- `ar1_rho` *(scalar numeric or* `NA`*)*; user-supplied ρ if a single common value was provided; `NA` otherwise.
- `ar1_rho_pair` ($m \times m$; optional); ρ actually used per pair (may be estimated from data or equal to the supplied value). AR(1) blocks require consecutive integer time indices for each subject.
- `ar1_estimated` ($m \times m$; optional logical); for each pair, `TRUE` if ρ was estimated internally; `FALSE` if supplied.
- `methods` *(character)*; method level names; matrix rows/columns follow this order.
- `two` *(scalar)*; LoA multiplier used (default `1.96`).
- `conf_level` *(scalar)*; nominal confidence level used for CIs.
- `data_long` *(data.frame)*; the long data used for fitting (`response`, `subject`, `method`, `time`). Included to facilitate plotting/reproducibility; not required for summary methods.

If `"ba_repeated"` **(exactly two methods)**, outputs are:

- `mean.diffs` *(scalar)*; estimated bias (method 2 - method 1).
- `lower.limit`, `upper.limit` *(scalars)*; LoA $\mu \pm$ two $\times$ SD for a single new pair.
- `critical.diff` *(scalar)*; `two * SD`; LoA half-width.
- `two`, `conf_level` *(scalars)*; as above.
- `CI.lines` *(named numeric)*; CI bounds for bias and both LoA (`*.ci.lower`, `*.ci.upper`) at `conf_level`.
- `means`, `diffs` *(vectors)*; per-pair means and differences used by plotting helpers.
- `based.on` *(integer)*; number of subject-time pairs used.
- `include_slope`, `beta_slope`; whether a proportional-bias slope was estimated and its value (if requested).
- `sigma2_subject`, `sigma2_resid`; variance components as above.
- `use_ar1`, `ar1_rho`, `ar1_estimated`; AR(1) settings/results as above (scalars for the two-method fit).

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 <- bland_altman_repeated(
  data = data,
  response = "y", subject = "subject", method = "method", time = "time",
  two = 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 <- bland_altman_repeated(
  response = data$y, subject = data$subject, method = data$method, time = data$time,
  two = 1.96, conf_level = 0.95,
  include_slope = FALSE,         # estimate proportional bias per pair
  use_ar1 = TRUE # model AR(1) within-subject
)

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

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

# -------- Two-method path (A vs B only) -----------------------------------
data_AB <- subset(data, method %in% c("A","B"))
baAB <- bland_altman_repeated(
  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)
```

`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 `bland_altman`).

Usage

```
ccc(data, ci = FALSE, conf_level = 0.95, verbose = FALSE)

## S3 method for class 'ccc'
print(x, digits = 4, ci_digits = 4, show_ci = c("auto", "yes", "no"), ...)

## S3 method for class 'ccc'
```

```
summary(
  object,
  digits = 4,
  ci_digits = 2,
  show_ci = c("auto", "yes", "no"),
  ...
)

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

## 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,
  ...
)
```

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).
`show_ci`	One of `"auto"`, `"yes"`, `"no"`. • `"auto"` (default): include CI columns only if the object has non-NA CIs. • `"yes"`: always include CI columns (may contain NA). • `"no"`: never include CI columns.
`...`	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.

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`, `bland_altman`

For repeated measurements look at `ccc_lmm_reml`, `ccc_pairwise_u_stat` or `bland_altman_repeated`

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_lmm_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_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
)
```

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, $D_m = (1/n_t)\, 11^\top$. • `"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 $R_i = \sigma_E^2\, C_i(\rho)$. With `ar="none"`, $C_i = I$. With `ar="ar1"`, within-subject residuals follow a *discrete* AR(1) process along the visit index after sorting by increasing time level. Ties retain input order, and any `NA` time code breaks the series so each contiguous block of non-`NA` times forms a run. The correlation between *adjacent observed visits* in a run is ρ; we do not use calendar-time gaps. Internally we work with the *precision* of the AR(1) correlation: for a run of length $L \ge 2$, 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 $j = 1, \dots, q_Z$,

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

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

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

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

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

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

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

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

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

together with the per-column update for $\sigma^2_{Z,j}$ 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 $a = (a_1, \dots, a_{n_t})^\top$ with $a_t \ge 0$, set

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

  so that

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

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

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

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

  giving

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

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

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

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

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

Concordance correlation coefficient. The CCC used is

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

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

CIs / SEs (delta method for CCC). Let

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

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

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

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

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

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 $\sigma_{Z,j}^2$, but cross-covariances among Z columns are set to zero (diagonal block). Column rescaling changes the implied prior on $b_{i,\mathrm{extra}}$ 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_pairwise_u_stat` 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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_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_lmm_reml(dat_bothRS, "y", "id", method = "method", time = "time",
             slope = "custom", slope_Z = Z_custom, verbose = TRUE)
```

`ccc_pairwise_u_stat` *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_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

`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 (≥ 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

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

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_lmm_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_lmm_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_pairwise_u_stat(df, response = "y", method = "method",
                            subject = "subject", time = "time")
print(ccc1)
summary(ccc1)
plot(ccc1)

# With confidence intervals
ccc2 <- ccc_pairwise_u_stat(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_pairwise_u_stat(df, response = "y", method = "method",
                    subject = "subject", time = "time", delta = 1)

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

# Presence/absence of disagreement (like kappa)
ccc_pairwise_u_stat(df, response = "y", method = "method",
                    subject = "subject", time = "time", delta = 0)
```

`distance_corr` *Pairwise Distance Correlation (dCor)*

Description

Computes all pairwise *distance correlations* using the unbiased U-statistic estimator for the numeric columns of a matrix or data frame, via a high-performance 'C++' backend ('OpenMP'-parallelised). Distance correlation detects general (including non-linear and non-monotonic) dependence between variables; unlike Pearson or Spearman, it is zero (in population) if and only if the variables are independent.

Prints a summary of the distance correlation matrix with optional truncation for large objects.

Generates a ggplot2 heatmap of the distance correlation matrix. Distance correlation is non-negative; the fill scale spans `[0, 1]`.

Usage

```
distance_corr(data, check_na = TRUE)

## S3 method for class 'distance_corr'
print(x, digits = 4, max_rows = NULL, max_cols = NULL, ...)

## S3 method for class 'distance_corr'
plot(
  x,
  title = "Distance correlation heatmap",
  low_color = "white",
  high_color = "steelblue1",
  value_text_size = 4,
  ...
)
```

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 `distance_corr`.
`digits`	Integer; number of decimal places to print.
`max_rows`	Optional integer; maximum number of rows to display. If `NULL`, all rows are shown.
`max_cols`	Optional integer; maximum number of columns to display. If `NULL`, all columns are shown.
`...`	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`.

Details

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

$$A_{ij}^{(x)} = \begin{cases} D_{ij}^{(x)} - \dfrac{r_i^{(x)} + r_j^{(x)}}{n-2} + \dfrac{S^{(x)}}{(n-1)(n-2)}, & i \neq j, \\ 0, & i = j\ . \end{cases}$$

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

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

with $\widehat{\mathrm{dVar}}_u^2(x)$ 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 `distance_corr` 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 <- distance_corr(X)
print(D, digits = 3)

## 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 <- distance_corr(XY)
# Compare Pearson vs unbiased distance correlation
round(c(pearson = cor(XY)[1, 2], dcor = D2["x", "y"]), 3)
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 <- distance_corr(X3)
print(D3[1:3, 1:3], digits = 2)
```

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

---

kendall_tau *Pairwise (or Two-Vector) Kendall's Tau Rank Correlation*

Description

Computes Kendall's tau rank correlation either for **all pairs of numeric columns** in a matrix/data frame, or for **two numeric vectors** directly (scalar path).

This function uses a scalable algorithm implemented in 'C++' to compute Kendall's tau-b (tie-robust). When there are no ties, tau-b reduces to tau-a. The implementation follows the Knight (1966) $O(n \log n)$ scheme, where a single sort on one variable, in-block sorting of the paired variable within tie groups, and a global merge-sort–based inversion count with closed-form tie corrections.

Prints a summary of the Kendall's tau correlation matrix, including description and method metadata.

Generates a ggplot2-based heatmap of the Kendall's tau correlation matrix.

Usage

```
kendall_tau(data, y = NULL, check_na = TRUE)

## S3 method for class 'kendall_matrix'
print(x, digits = 4, max_rows = NULL, max_cols = 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,
  ...
)
```

Arguments

`data`	For matrix/data frame, it is expected a numeric matrix or a data frame with at least two numeric columns. All non-numeric columns will be 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 you have already filtered or imputed them.
`x`	An object of class `kendall_matrix`.
`digits`	Integer; number of decimal places to print
`max_rows`	Optional integer; maximum number of rows to display. If `NULL`, all rows are shown.
`max_cols`	Optional integer; maximum number of columns to display. If `NULL`, all columns are shown.
`...`	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`.

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`).

Performance:

- In the **two-vector mode** (`y` supplied), the C++ backend uses a raw-double path (no intermediate 2×2 matrix, no discretisation).
- In the **matrix/data-frame mode**, columns are discretised once and all pairwise correlations are computed via the Knight $O(n\log n)$ procedure; where available, pairs are evaluated in parallel.

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.
- 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.

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.

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)
plot(kt)

# Two-vector mode (scalar path)
x <- rnorm(1000); y <- 0.5 * x + rnorm(1000)
kendall_tau(x, y)

# With a large data frame
df <- data.frame(x = rnorm(1e4), y = rnorm(1e4), z = rnorm(1e4))
kendall_tau(df)

# Including ties
tied_df <- data.frame(
  v1 = rep(1:5, each = 20),
  v2 = rep(5:1, each = 20),
  v3 = rnorm(100)
)
```

```
kt <- kendall_tau(tied_df)
print(kt)
plot(kt)

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

`partial_correlation` *Partial correlation matrix (sample / ridge / OAS)*

Description

Computes the Gaussian partial correlation matrix from a numeric data frame or matrix. The covariance matrix can be estimated using:

- **Unbiased sample covariance**: the standard empirical covariance estimator.
- **Ridge-regularised covariance**: adds a positive ridge penalty to improve stability when the covariance matrix is near-singular.
- **OAS shrinkage to a scaled identity**: recommended when $p \gg n$, as it reduces estimation error by shrinking towards a scaled identity matrix.

The method uses a high-performance 'C++' backend.

Prints only the partial correlation matrix (no attribute spam), with an optional one-line header stating the estimator used.

Produces a **ggplot2**-based heatmap of the partial correlation matrix stored in `x$pcor`. Optionally masks the diagonal and/or reorders variables via hierarchical clustering of $1 - |pcor|$.

Usage

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

## S3 method for class 'partial_corr'
print(x, digits = 3, show_method = TRUE, max_rows = NULL, max_cols = NULL, ...)

## S3 method for class 'partial_corr'
plot(
  x,
  title = NULL,
  low_color = "indianred1",
  high_color = "steelblue1",
  mid_color = "white",
  value_text_size = 4,
  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 `"oas"`, `"ridge"`, `"sample"`. Default `"oas"`.
`lambda`	Numeric ≥ 0; ridge penalty added to the covariance diagonal when `method = "ridge"`. 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`
`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`.
`max_rows`, `max_cols`	Optional integer limits for display; if provided, the printed matrix is truncated with a note about omitted rows/cols.
`...`	Additional arguments passed to `ggplot2::theme()` or other **ggplot2** layers.
`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`.
`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

  $$\Sigma = (1-\rho)\,\hat{\Sigma}_{\mathrm{MLE}} + \rho\,\mu_I I_p.$$

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:

$$\mathrm{pcor}_{ij} \;=\; -\frac{\theta_{ij}}{\sqrt{\theta_{ii}\,\theta_{jj}}}, \qquad \mathrm{pcor}_{ii} = 1.$$

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.

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.
- `method`: the estimator used (`"oas"`, `"ridge"`, or `"sample"`).
- `lambda`: ridge 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 `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.

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 <- partial_correlation(X, method = "oas")

# 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)

## 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   <-
 partial_correlation(X_hd, method = "oas",   return_cov_precision = TRUE)
pc_ridge <-
 partial_correlation(X_hd, method = "ridge", lambda = 1e-2,
                     return_cov_precision = TRUE)
pc_samp  <-
 partial_correlation(X_hd, method = "sample", 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)

## 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 all pairwise Pearson correlation coefficients for the numeric columns of a matrix or data frame using a high-performance 'C++' backend.

This function uses a direct Pearson formula implementation in 'C++' to achieve fast and scalable correlation computations, especially for large datasets.

Prints a summary of the Pearson correlation matrix, including description and method metadata.

Generates a ggplot2-based heatmap of the Pearson correlation matrix.

Usage

```
pearson_corr(data, check_na = TRUE)

## S3 method for class 'pearson_corr'
print(x, digits = 4, max_rows = NULL, max_cols = 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,
  ...
)
```

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.
`x`	An object of class `pearson_corr`.
`digits`	Integer; number of decimal places to print in the concordance
`max_rows`	Optional integer; maximum number of rows to display. If `NULL`, all rows are shown.
`max_cols`	Optional integer; maximum number of columns to display. If `NULL`, all columns are shown.
`...`	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`.

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`. No missing values are permitted in X; columns must have at least two distinct, non-missing values.

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.

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)
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)
}
```

`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, show_ci = c("auto", "yes", "no"), ...)
```

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.
`show_ci`	One of `"auto"`, `"yes"`, `"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.

`schafer_corr` *Schafer-Strimmer shrinkage correlation*

Description

Computes a shrinkage correlation matrix using the Schafer-Strimmer approach with an analytic, data-driven intensity $\hat{\lambda}$. The off-diagonals of the sample Pearson correlation R are shrunk towards zero, yielding $R_{\mathrm{shr}} = (1-\hat{\lambda})R + \hat{\lambda}I$ with $\mathrm{diag}(R_{\mathrm{shr}}) = 1$, stabilising estimates when $p \geq n$.

This function uses a high-performance 'C++' backend that forms $X^\top X$ via 'BLAS' 'SYRK', applies centring via a rank-1 update, converts to Pearson correlation, estimates $\hat{\lambda}$, and shrinks the off-diagonals: $R_{\mathrm{shr}} = (1-\hat{\lambda})R + \hat{\lambda}I$.

Prints a summary of the shrinkage correlation matrix with optional truncation for large objects.

Heatmap of the shrinkage correlation matrix with optional hierarchical clustering and triangular display. Uses **ggplot2** and `geom_raster()` for speed on larger matrices.

Usage

```
schafer_corr(data)

## S3 method for class 'schafer_corr'
print(x, digits = 4, max_rows = NULL, max_cols = 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_values = FALSE,
  value_text_limit = 60,
  value_text_size = 3,
  palette = c("diverging", "viridis"),
  ...
)
```

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 `NA`s.
`x`	An object of class `schafer_corr`.
`digits`	Integer; number of decimal places to print.
`max_rows`	Optional integer; maximum number of rows to display. If `NULL`, all rows are shown.
`max_cols`	Optional integer; maximum number of columns to display. If `NULL`, all columns are shown.
`...`	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_values`	Logical; print correlation values inside tiles (only if matrix dimension $\leq$ `value_text_limit`).
`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"`.

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, max_rows = 6, max_cols = 6)
plot(Rshr)

## Shrinkage typically moves the sample correlation closer to the truth
Raw <- 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)
}
```

`spearman_rho` *Pairwise Spearman's rank correlation*

Description

Computes all pairwise Spearman's rank correlation coefficients for the numeric columns of a matrix or data frame using a high-performance 'C++' backend.

This function ranks the data and computes Pearson correlation on ranks, which is equivalent to Spearman's rho. It supports large datasets and is optimized in 'C++' for performance.

Prints a summary of the Spearman's correlation matrix, including description and method metadata.

Generates a ggplot2-based heatmap of the Spearman's rank correlation matrix.

Usage

```
spearman_rho(data, check_na = TRUE)

## S3 method for class 'spearman_rho'
print(x, digits = 4, max_rows = NULL, max_cols = 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,
  ...
)
```

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.
`x`	An object of class `spearman_rho`.
`digits`	Integer; number of decimal places to print.
`max_rows`	Optional integer; maximum number of rows to display. If `NULL`, all rows are shown.

`max_cols`	Optional integer; maximum number of columns to display. If `NULL`, all columns are shown.
`...`	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`.

Details

For each column $j = 1, \ldots, p$, let $R_{\cdot j} \in \{1, \ldots, n\}^n$ denote the (mid-)ranks of $X_{\cdot j}$, assigning average ranks to ties. The mean rank is $\bar{R}_j = (n+1)/2$ regardless of ties. Define the centred rank vectors $\tilde{R}_{\cdot j} = R_{\cdot j} - \bar{R}_j \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. The Spearman correlation between columns i and j is the Pearson correlation of their rank vectors:

$$\rho_S(i,j) \;=\; \frac{\sum_{k=1}^n (R_{ki} - \bar{R}_i)(R_{kj} - \bar{R}_j)}{\sqrt{\sum_{k=1}^n (R_{ki} - \bar{R}_i)^2}\,\sqrt{\sum_{k=1}^n (R_{kj} - \bar{R}_j)^2}}.$$

In matrix form, with $R = [R_{\cdot 1}, \ldots, R_{\cdot p}]$, $\mu = (n+1)\mathbf{1}_p/2$ for $\mathbf{1}_p \in \mathbb{R}^p$, and $S_R = \big(R - \mathbf{1}\mu^\top\big)^\top \big(R - \mathbf{1}\mu^\top\big)/(n-1)$, the Spearman correlation matrix is

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

When there are no ties, the familiar rank-difference formula obtains

$$\rho_S(i,j) \;=\; 1 - \frac{6}{n(n^2-1)} \sum_{k=1}^n d_k^2, \quad d_k \;=\; R_{ki} - R_{kj},$$

but this expression does *not* hold under ties; computing Pearson on mid-ranks (as above) is the standard tie-robust approach. Without ties, $\mathrm{Var}(R_{\cdot j}) = (n^2-1)/12$; with ties, the variance is smaller.

$\rho_S(i,j) \in [-1,1]$ and $\widehat{\rho}_S$ is symmetric positive semi-definite by construction (up to floating-point error). The implementation symmetrises the result to remove round-off asymmetry. Spearman's correlation is invariant to strictly monotone transformations applied separately to each variable.

Computation. Each column is ranked (mid-ranks) to form R. The product $R^\top R$ is computed via a 'BLAS' symmetric rank update ('SYRK'), and centred using

$$(R - \mathbf{1}\mu^\top)^\top (R - \mathbf{1}\mu^\top) \;=\; R^\top R \;-\; n\,\mu\mu^\top,$$

avoiding an explicit centred copy. Division by $n-1$ yields the sample covariance of ranks; standardising by $D^{-1/2}$ gives $\widehat{\rho}_S$. Columns with zero rank variance (all values equal) are returned as `NA` along their row/column; the corresponding diagonal entry is also `NA`.

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. 'OpenMP' parallelism is used across columns for ranking, and a 'BLAS' 'SYRK' kernel is used for the matrix product when available.

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.

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.

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
# AR(1) correlation
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))

# Monotone transforms to some columns
X_mono <- X
# exponential
X_mono[, 1] <- exp(X_mono[, 1])
# softplus
X_mono[, 2] <- log1p(exp(X_mono[, 2]))
# odd monotone polynomial
X_mono[, 3] <- X_mono[, 3]^3

sp_X <- spearman_rho(X)
sp_m <- spearman_rho(X_mono)

# Spearman should be (nearly) unchanged under monotone transformations
round(max(abs(sp_X - sp_m)), 3)
# heatmap of Spearman correlations
```

```
plot(sp_X)

## Ties handled via mid-ranks
tied <- cbind(
  # many ties
  a = rep(1:5, each = 20),
  # noisy reverse order
  b = rep(5:1, each = 20) + rnorm(100, sd = 0.1),
  # ordinal with ties
  c = as.numeric(gl(10, 10))
)
sp_tied <- spearman_rho(tied)
print(sp_tied, digits = 2)

## Bivariate normal, theoretical Spearman's rho
## For BVN with Pearson correlation r, rho_S = (6/pi) * asin(r/2).
r_target <- c(-0.8, -0.4, 0, 0.4, 0.8)
n2 <- 200
est <- true_corr <- numeric(length(r_target))
for (i in seq_along(r_target)) {
  R2 <- matrix(c(1, r_target[i], r_target[i], 1), 2, 2)
  Z  <- matrix(rnorm(n2 * 2), n2, 2) %*% chol(R2)
  s  <- spearman_rho(Z)
  est[i]  <- s[1, 2]
  true_corr[i] <- (6 / pi) * asin(r_target[i] / 2)
}
cbind(r_target, est = round(est, 3), theory = round(true_corr, 3))

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

summary.ccc_lmm_reml *Summary Method for* ccc_lmm_reml *Objects*

Description

Produces a detailed summary of a "ccc_lmm_reml" object, including Lin's CCC estimates and associated variance component estimates per method pair.

Usage

```
## S3 method for class 'ccc_lmm_reml'
summary(
  object,
  digits = 4,
  ci_digits = 2,
  show_ci = c("auto", "yes", "no"),
  ...
)
```

Arguments

`object`	An object of class `"ccc_lmm_reml"`, as returned by `ccc_lmm_reml()`.
`digits`	Integer; number of decimal places to round CCC estimates and components.
`ci_digits`	Integer; decimal places for confidence interval bounds.
`show_ci`	Character string indicating whether to show confidence intervals: `"auto"` (default) shows only if non-NA CIs exist, `"yes"` always shows CIs, `"no"` never shows them.
`...`	Additional arguments (ignored).

Value

A data frame of class `"summary.ccc_lmm_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`.

`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()`, `biweight_mid_corr()`, `partial_correlation()`, `distance_corr()`, 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)
}
```

Index