Package 'smartcor'

October 26, 2026

Title Automatic Correlation Method Selection Based on Variable Types

Version 1.2.0

Description Detects variable types (continuous, count, binary, ordinal, categorical), checks outliers, normality, latent normality, and linearity, and selects the appropriate correlation method for each pair. Supports 18 methods: Pearson, Spearman, Kendall's tau, point-biserial, biserial, rank-biserial, phi, tetrachoric, Yule's Q, polychoric, polyserial, Goodman-Kruskal's gamma, the correlation ratio (eta-squared), Kruskal-Wallis eta-squared, Freeman's theta, Cramer's V, Tschuprow's T, and Theil's U, each with a confidence interval and p-value. Explains the selection rationale in the output, follows tidy data principles, and works in both interactive and scripted workflows. The methodology is described in Harshvardhan and Ranjan (2026) <doi:10.48550/arXiv.2607.22285>.

License GPL (>= 3)

URL https://harshvardhaniimi.github.io/smartcor/, https://github.com/harshvardhaniimi/smartcor

BugReports https://github.com/harshvardhaniimi/smartcor/issues

Encoding UTF-8

Depends R (>= 3.6.0)

Imports cli (>= 3.0.0), fNonlinear, generics, ggplot2 (>= 3.4.0), graphics, grDevices, lmtest, Matrix, moments, MVN (>= 6.0), mvtnorm, polycor, rlang (>= 1.0.0), stats, tibble, utils, withr

Suggests effectsize, knitr, MASS, rmarkdown, rstatix, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

RoxygenNote 7.3.3

NeedsCompilation no

Author M. Harshvardhan [aut, cre, cph], Pritam Ranjan [aut, cph]

Maintainer M. Harshvardhan <harshvardhan@aus.edu>

Repository CRAN

Date/Publication 2026-10-26 14:10:02 UTC

Contents

available_methods 2
compare_methods 3
detect_outliers 5
detect_type 6
ggcor 7
plot.smartcormat 8
print.smartcor 10
print.smartcormat 11
print.smartcor_comparison 11
print.smartcor_lintest 12
print.smartcor_normtest 12
smart_cor 13
smart_cormat 16
smart_cor_df 18
test_bivariate_normality 20
test_linearity 23
tidy.smartcor 24
tidy.smartcormat 25
tidy.smartcor_comparison 25

Index **27**

`available_methods` *Which Methods Apply to a Pair of Variable Types*

Description

Lists the methods `smart_cor()` can compute for two variable types. Use it before forcing a `method`.

Usage

```
available_methods(type_x, type_y)
```

Arguments

`type_x`, `type_y`	Character strings: `"continuous"`, `"count"`, `"binary"`, `"ordinal"`, or `"categorical"`.

Value

A tibble with columns `method` (code name) and `label` (human-readable description).

Examples

```
path = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(path)
available_methods(detect_type(gss$degree), detect_type(gss$happy))
```

`compare_methods` *Run Every Applicable Method on One Pair*

Description

Computes the correlation between `x` and `y` with every method that applies to their type combination and lines the results up side by side. Use it to check how much the method choice changes the estimate.

Usage

```
compare_methods(
  x,
  y,
  x_type = NULL,
  y_type = NULL,
  assume_latent_normal = "auto",
  alpha = 0.05,
  ordinal_threshold = 10,
  ignore_na = TRUE,
  conf_level = 0.95,
  bootstrap = "auto",
  n_boot = 500,
  n_perm = 500,
  seed = NULL,
  polyserial_ml = FALSE,
  verbose = TRUE
)
```

Arguments

`x, y`	Vectors of the same length. May be numeric, factor, character, or logical.
`x_type, y_type`	Optional character strings to override automatic type detection. One of `"continuous"`, `"count"`, `"binary"`, `"ordinal"`, or `"categorical"`.
`assume_latent_normal`	`"auto"` (default) or logical. Decides which method gets the `recommended` flag, exactly as in `smart_cor()`.
`alpha`	Significance level for the diagnostics that decide the `recommended` flag. Default: `0.05`.
`ordinal_threshold`	Integer passed to `detect_type()`. Default: `10`.

ignore_na	Logical. If TRUE (default), rows with missing values are removed. If FALSE, the function errors when NA values are present.
conf_level	Numeric between 0 and 1. Confidence level for every interval in the table. Default: 0.95.
bootstrap	Character or logical: "auto" (default) bootstraps only the methods without an analytic CI, TRUE bootstraps all of them, FALSE none.
n_boot	Integer. Bootstrap replications per method. Default: 500.
n_perm	Integer. Permutation shuffles for the Theil's U and Freeman's theta p-values. Default: 500.
seed	Optional integer seed for the bootstrap and permutation draws; the global RNG state is saved and restored.
polyserial_ml	Logical. Fit polyserial and biserial by full maximum likelihood. Default: FALSE; TRUE is recommended (see smart_cor()).
verbose	Logical. If TRUE (default), prints informative messages.

Details

The function computes every method listed by available_methods() for the detected type pair. The recommended column marks the method that smart_cor() would select under the assume_latent_normal setting, so the recommended row always matches what smart_cor() and smart_cormat() return.

When one variable is continuous or count and the other is categorical, only the Cramer's V row bins the continuous or count variable into quintiles; the two eta-squared measures use the raw values.

Methods that fail (a latent-variable optimiser that does not converge, say) are kept in the output with estimate = NA.

Value

An object of class "smartcor_comparison": a list with elements:

results A tibble::tibble() with columns method, method_label, estimate, statistic, p.value, ci_lower, ci_upper, ci_method, ci_source, and recommended (logical flag for the default method).

x_name Name of the first variable.

y_name Name of the second variable.

x_type Detected (or specified) type of x.

y_type Detected (or specified) type of y.

n_complete Number of complete observations used.

conf_level Confidence level used for the intervals.

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)

# Two ordinal variables: Kendall, Spearman, gamma, and polychoric
compare_methods(gss$degree, gss$happy,
                assume_latent_normal = FALSE, bootstrap = FALSE)
```

detect_outliers *Flag Outliers with the Outer-Fence Rule*

Description

Flags values beyond Tukey's outer fences, `Q1 - k * IQR` and `Q3 + k * IQR`, with quartiles of type 7 (the R default). `smart_cor()` uses this rule as the first diagnostic for every pair with a continuous or count variable.

Usage

```
detect_outliers(x, k = 3)

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

Arguments

x	A numeric vector (continuous, count, or ordinal). An ordered factor is replaced by its integer codes. Missing values are dropped.
k	Fence multiplier. Default: 3 (the outer fences).
...	Ignored.

Details

Values strictly below the lower fence or strictly above the upper fence are outliers. When the IQR is 0 (more than half of the values are tied) the rule is degenerate: nothing is flagged, `degenerate` is `TRUE`, and the repeated values need separate inspection.

The skewness and excess kurtosis are moment estimators with divisor n, as used by the Jarque-Bera test.

Value

An object of class `"smartcor_outliers"`: a list with `has_outliers` (logical), `outliers` (the outlying values), `index` (their positions in `x`), `lower` and `upper` (the fences), `q1`, `q3`, `iqr`, `degenerate` (`TRUE` when the IQR is 0), `skewness`, `excess_kurtosis`, and `n` (the number of non-missing values).

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)
detect_outliers(gss$coninc)
```

`detect_type` *Detect the Statistical Type of a Variable*

Description

Classifies a variable as continuous, binary, ordinal, count, or categorical from its R class and the number of distinct non-missing values. The rest of the package leans on this to pick correlation methods, but it is exported because the classification is useful on its own.

Usage

```
detect_type(x, ordinal_threshold = 10, detect_count = TRUE, verbose = FALSE)
```

Arguments

`x`	A vector (numeric, factor, character, or logical).
`ordinal_threshold`	Integer. Numeric variables with this many or fewer unique non-missing values are classified as ordinal. Default: `10`.
`detect_count`	Logical. If `TRUE` (default), numeric variables that are non-negative integer-valued and exceed `ordinal_threshold` distinct values are classified as `"count"`. Set to `FALSE` to treat all such variables as `"continuous"`.
`verbose`	Logical. If `TRUE`, prints the detection reasoning. Default: `FALSE`.

Details

The detection follows these rules, in order:

1. **Any variable** with exactly 2 unique values is `"binary"`, whatever its class (a two-level ordered factor included).
2. **Ordered factors** with more than 2 levels are `"ordinal"`.
3. **Unordered factors and character vectors** with more than 2 unique values are `"categorical"`.
4. **Logical vectors** are classified as `"binary"`.
5. **Numeric vectors** with `ordinal_threshold` (10 by default) or fewer unique values are `"ordinal"`.
6. **Numeric vectors** that are non-negative integers with more than `ordinal_threshold` unique values are classified as `"count"`. Disable by setting `detect_count = FALSE`.
7. All other numeric vectors are `"continuous"`.

The `ordinal_threshold` parameter controls the heuristic for numeric variables: if a numeric variable has this many distinct values or fewer, it is assumed to represent ordered categories (e.g., a 1–5 Likert scale stored as integers). Set to `0` to disable this heuristic and treat all numeric variables as continuous (or count, if non-negative integer-valued).

Value

A single character string: `"continuous"`, `"binary"`, `"ordinal"`, `"count"`, or `"categorical"`.

Examples

```
path = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(path)
vapply(gss, detect_type, character(1))
```

---

ggcor *ggplot2 Heatmaps for Smart Correlation Matrices*

Description

Two heatmaps for `smart_cormat()` output. `ggcor_heatmap()` shows the correlation values on a diverging scale centred at zero. `ggcor_method_heatmap()` colours each cell by the selected method.

Usage

```
ggcor_heatmap(
  x,
  palette = "blue_red",
  digits = 2,
  title = NULL,
  show_values = TRUE,
  text_size = 3.5,
  lower_only = FALSE,
  ...
)

ggcor_method_heatmap(
  x,
  title = NULL,
  show_values = TRUE,
  text_size = 3.5,
  digits = 2,
  ...
)
```

Arguments

`x`	A `smartcormat` object (from `smart_cormat()`), or a named list with a `correlations` element (e.g., from `smart_cor_df()`).
`palette`	Character string selecting the colour palette. For `ggcor_heatmap`: one of `"blue_red"` (default), `"purple_green"`, or a vector of 3 colours (low, mid, high).
`digits`	Integer. Number of decimal places in cell labels. Default: `2`.
`title`	Optional plot title. If `NULL`, a default is used.
`show_values`	Logical. Show correlation values in cells? Default: `TRUE`.

text_size	Numeric. Base text size for cell labels. Default: 3.5.
lower_only	Logical. Show only the lower triangle? Default: FALSE.
...	Currently ignored.

Details

Both functions use the **ggplot2** package.

Value

A ggplot object that can be customised further.

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)
mat = smart_cormat(
  gss[, c("age", "coninc", "degree", "sex")],
  assume_latent_normal = FALSE,
  verbose = FALSE
)

# correlation heatmap
ggcor_heatmap(mat)

# lower triangle only
ggcor_heatmap(mat, lower_only = TRUE)

# method-selection heatmap
ggcor_method_heatmap(mat)

# customise further with ggplot2
ggcor_heatmap(mat) + ggplot2::labs(caption = "Frozen GSS 2024 extract")
```

---

plot.smartcormat *Plot a Correlation Heatmap*

Description

Draws an annotated heatmap of a smart correlation matrix with base R. Each cell is coloured by the correlation and labelled with its value.

Usage

```
## S3 method for class 'smartcormat'
plot(
  x,
  palette = "blue_red",
  digits = 2,
```

```
  title = NULL,
  show_values = TRUE,
  text_size = 1,
  ...
)

plot_cor_heatmap(
  x,
  palette = "blue_red",
  digits = 2,
  title = NULL,
  show_values = TRUE,
  text_size = 1,
  ...
)
```

Arguments

x	A smartcormat object (from smart_cormat()), or a named list with a correlations element (e.g., from smart_cor_df()).
palette	Character string selecting the colour palette. One of: • "blue_red" (default): diverging blue-white-red • "purple_green": diverging purple-white-green • "viridis": sequential yellow-green-blue • "heat": sequential white-yellow-red • "cool": sequential white-cyan-blue
digits	Integer. Number of decimal places shown in cell labels. Default: 2.
title	Optional character string for the plot title. If NULL (default), a generic title is used.
show_values	Logical. Whether to display numeric values in each cell. Default: TRUE.
text_size	Numeric. Scaling factor for cell text size. Default: 1.
...	Additional arguments (currently ignored).

Details

The function works with any object that contains a correlations element (matrix or data frame). This includes objects returned by smart_cormat() and smart_cor_df().

Custom colour palettes can be supplied by passing a character vector of colours to palette instead of a named preset. The vector should contain at least 3 colours and will be interpolated to 100 steps.

Value

Invisibly returns the correlation matrix (numeric).

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)

# Plot a smart_cormat result
mat = smart_cormat(
  gss[, c("age", "coninc", "degree", "sex")],
  assume_latent_normal = FALSE,
  verbose = FALSE
)
plot(mat)

# different palettes
plot(mat, palette = "purple_green")
plot(mat, palette = "viridis")

# Plot a smart_cor_df result
res = smart_cor_df(gss[, c("age", "coninc", "degree", "sex")],
                   assume_latent_normal = FALSE)
plot_cor_heatmap(res)

# custom colour vector
plot_cor_heatmap(res, palette = c("darkblue", "white", "darkred"))
```

---

print.smartcor *Print a smartcor Object*

Description

Shows the estimate, the method that was chosen and why, the inference payload, and one line for each diagnostic that ran (outer-fence rule, Mardia or Jarque-Bera, the LR latent-normality test, and the linearity test).

Usage

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

Arguments

x	A smartcor object.
...	Ignored.

Value

Invisibly returns x.

print.smartcormat *Print a smartcormat Object*

Description

Prints the correlation matrix, the two-letter method codes for each cell, and a legend for the codes that appear.

Usage

```
## S3 method for class 'smartcormat'
print(x, digits = 2, show_methods = TRUE, ...)
```

Arguments

x	A smartcormat object.
digits	Integer. Number of significant figures. Default: 2.
show_methods	Logical. Whether to print the method matrix. Default: TRUE.
...	Ignored.

Value

Invisibly returns x.

print.smartcor_comparison *Print a Method Comparison*

Description

Print a Method Comparison

Usage

```
## S3 method for class 'smartcor_comparison'
print(x, digits = 4, ...)
```

Arguments

x	A smartcor_comparison object from compare_methods().
digits	Integer. Number of decimal places. Default: 4.
...	Ignored.

Value

Invisibly returns x.

`print.smartcor_lintest` *Print a Linearity Test*

Description

Print a Linearity Test

Usage

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

Arguments

`x`	A `smartcor_lintest` object from `test_linearity()`.
`...`	Ignored.

Value

Invisibly returns `x`.

`print.smartcor_normtest` *Print a Normality Test*

Description

Print a Normality Test

Usage

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

Arguments

`x`	A `smartcor_normtest` object from `test_bivariate_normality()`.
`...`	Ignored.

Value

Invisibly returns `x`.

smart_cor *Choose and Compute a Correlation*

Description

Detects the types of x and y, selects a suitable correlation method, and returns the estimate with its confidence interval and p-value.

Usage

```
smart_cor(
  x,
  y,
  x_type = NULL,
  y_type = NULL,
  method = NULL,
  assume_latent_normal = "auto",
  ordinal_threshold = 10,
  ignore_na = TRUE,
  conf_level = 0.95,
  alpha = 0.05,
  bootstrap = "auto",
  n_boot = 500,
  n_perm = 500,
  seed = NULL,
  polyserial_ml = FALSE,
  verbose = TRUE
)
```

Arguments

x, y	Vectors of the same length. May be numeric, factor, character, or logical.
x_type, y_type	Optional character strings ("continuous", "count", "binary", "ordinal", "categorical") to override automatic type detection.
method	Optional character string to force a specific method. One of "pearson", "spearman", "kendall", "point_biserial", "biserial", "rank_biserial", "phi", "tetrachoric", "yules_q", "polychoric", "polyserial", "gamma", "eta_squared", "kw_eta_squared", "cramers_v", "freemans_theta", "theils_u", or "tschuprows_t". See available_methods().
assume_latent_normal	"auto" (default), logical, or NULL. "auto" runs a likelihood-ratio chi-square test (via test_bivariate_normality()) for the latent-normality node; a saturated 2 x 2 table gives phi. TRUE forces the node to "yes" (biserial / polyserial / polychoric / tetrachoric); FALSE forces it to "no" (point-biserial / Kendall or Spearman / rank-biserial / Yule's Q). The outlier and Jarque-Bera nodes still run under TRUE and FALSE. NULL asks interactively (defaults to TRUE in non-interactive sessions).

`ordinal_threshold`
: Integer passed to `detect_type()`. Default: `10`.

`ignore_na`
: Logical. If `TRUE` (default), rows with missing values in either `x` or `y` are silently removed (pairwise complete observations). If `FALSE`, the function errors when `NA` values are present.

`conf_level`
: Numeric between 0 and 1. Confidence level for intervals. Default: `0.95`.

`alpha`
: Significance level for every diagnostic test (Mardia, Jarque-Bera, the LR latent-normality test, and `test_linearity()`). Default: `0.05`.

`bootstrap`
: Character or logical. Controls bootstrap inference: `"auto"` (default) bootstraps when analytic CIs are unavailable or the linearity rule recommends robust/bootstrap Pearson inference; `TRUE` always replaces the analytic CI with a percentile-bootstrap CI; `FALSE` never bootstraps. P-values remain analytic (or permutation-based for Theil's U and Freeman's theta) under every setting.

`n_boot`
: Integer. Number of bootstrap replications when bootstrapping. Default: `500`.

`n_perm`
: Integer. Number of shuffles for the permutation p-values of Theil's U and Freeman's theta. Default: `500`.

`seed`
: Optional integer seed for the bootstrap and permutation draws. The global `.Random.seed` is saved and restored around a seeded call, so the global RNG state is restored. Seeded results are reproducible across R sessions but do not match the Python package, whose numpy generator draws differently.

`polyserial_ml`
: Logical. How polyserial and biserial correlations are fitted. `FALSE` (default) calls `polycor::polyserial()` with its own defaults, the quick two-step estimator (adding only `std.err = TRUE` for the interval and p-value). We recommend `TRUE`, which fits the full maximum-likelihood estimate (the correlation and the thresholds together) with a stricter convergence tolerance, and retries from a safe starting value if the first fit fails.

`verbose`
: Logical. If `TRUE` (default), prints messages about the detection and selection process.

Details

The method selection follows Harshvardhan, M. and Ranjan, P. (2026), "smartcor: Intelligent Correlation Method Selection for Mixed Variable Types" (arXiv:2607.22285, doi:10.48550/arXiv.2607.22285); the package vignettes (`browseVignettes("smartcor")`) cover the same material in more depth:

Variable types	Default method	Alternative
Correlation methods (signed, from -1 to +1)		
continuous + continuous	Pearson (linear, no outliers)	Spearman (outliers or nonlinear), Ken
continuous + binary	Biserial (if latent normality holds)	Point-biserial; rank-biserial (outliers c
binary + binary	Pearson = phi	Tetrachoric (latent normal), Yule's Q (
continuous + ordinal	Polyserial (if latent normality holds)	Kendall's tau (m <= 5) or Spearman
ordinal + ordinal	Polychoric (if latent normality holds)	Kendall's tau (m <= 5) or Spearman; C
binary + ordinal	Polychoric (if latent normality holds)	Rank-biserial; Kendall's tau, Spearma
Association measures (unsigned, from 0 to 1)		
continuous + categorical	Correlation ratio η^2	Kruskal-Wallis η^2_H (outliers or non-no
binary + categorical	Cramer's V	–

ordinal + categorical	Freeman's theta	Cramer's V
categorical + categorical	Cramer's V	Theil's U, Tschuprow's T

Count variables are treated as continuous, so each count combination follows the corresponding continuous row. m is the number of categories of the ordinal variable (the smaller of the two for an ordinal pair).

The selection starts from diagnostics run at significance level `alpha`:

- The outer-fence rule (`detect_outliers()`, Q1 - 3 IQR and Q3 + 3 IQR) is applied to each continuous variable of a continuous-continuous pair, to the continuous variable of a continuous-ordinal pair, and to the residuals around the group means for continuous-binary and continuous-categorical pairs. It is never applied to ordinal variables.
- Normality: Mardia's test (`MVN::mardia()`) for two continuous variables; the Jarque-Bera test (`moments::jarque.test()`) on the continuous variable (continuous-ordinal) or on the within-group residuals (continuous-binary, continuous-categorical).
- Latent normality: the LR chi-square test of `test_bivariate_normality()` on the (binned) contingency table.
- Linearity: `test_linearity()` for two continuous variables.

A continuous-continuous pair gets Spearman when the outer fence flags either variable, and otherwise follows `test_linearity()`, which runs its diagnostics in both directions. When Pearson is selected and Mardia's test rejects, the rationale says the interval and p-value are approximate. For continuous-binary, continuous-ordinal, and continuous-categorical pairs, a fence flag or a Jarque-Bera rejection selects the rank-based method (rank-biserial, Kendall/Spearman, Kruskal-Wallis η^2_H). Otherwise, and for binary-ordinal and ordinal pairs, the latent-normality node decides between the latent-variable method and the observed-scale one. A binary-binary pair under `"auto"` gets phi: its 2 x 2 table is saturated (df = 0), so latent normality cannot be tested. Inconclusive tests keep Pearson and report the reason. The linearity decision does not depend on the order of `x` and `y`. An explicit `method` bypasses all diagnostics. The tests that ran are returned in `diagnostics`.

When a latent-variable estimator fails numerically (singular Hessian, sparse table), the function falls back to Kendall's tau, Spearman, or point-biserial and the returned `method`, `method_label`, and `rationale` name the method actually used, not the one that failed.

One caveat on the point-biserial interval: it reuses the Fisher-z variance 1/(n - 3), which assumes bivariate normality; a binary margin cannot satisfy that exactly, so treat the interval as approximate.

Theil's U is asymmetric: `method = "theils_u"` computes U(y|x), the proportion of the entropy in `y` explained by `x`. Swap the arguments for the other direction. Its p-value, like that of Freeman's theta, comes from a permutation test (`n_perm` shuffles), not from the bootstrap. Kruskal-Wallis η^2_H, Freeman's theta, and Theil's U have no analytic interval and get a percentile bootstrap under `bootstrap = "auto"`.

Value

An object of class `"smartcor"`: a named list with elements `estimate`, `method`, `method_label`, `x_type`, `y_type`, `n`, `n_complete`, `statistic`, `p.value`, `ci_lower`, `ci_upper`, `conf_level`, `ci_method`, `ci_source`, `p_method`, `null_hypothesis`, `p_interpretation`, `alternatives`, `rationale`, `normality_test` (the LR latent-normality test, when it ran), `diagnostics`, `outlier_flag`, `normality_p`, `latent_p`,

`linearity_p`, and `linearity_test` (when the residual diagnostic rule ran). `diagnostics` is a list with `outliers` (a `detect_outliers()` result, or a list of two for a continuous-continuous pair), `normality` (Mardia or Jarque-Bera), `latent` (the LR test), and `linearity` (the `test_linearity()` result); entries that were not computed are `NULL`. The four summary fields are `NA` when not computed. Has custom `print.smartcor()` and `tidy.smartcor()` methods.

Examples

```
path = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(path)

# Continuous and binary variables: the diagnostics pick the method.
smart_cor(gss$coninc, gss$sex, verbose = FALSE)

# Two ordinal variables use Kendall's tau when latent normality is not assumed.
smart_cor(gss$degree, gss$happy,
          assume_latent_normal = FALSE, verbose = FALSE)

# Force a method when the analysis calls for it.
smart_cor(gss$coninc, gss$age, method = "spearman", verbose = FALSE)
```

`smart_cormat` *Build a Mixed-Type Correlation Matrix*

Description

Runs `smart_cor()` on every pair of columns and records both the estimate and the method used for each pair.

Usage

```
smart_cormat(
  data,
  cols = NULL,
  types = NULL,
  assume_latent_normal = "auto",
  ordinal_threshold = 10,
  ignore_na = TRUE,
  conf_level = 0.95,
  alpha = 0.05,
  bootstrap = "auto",
  n_boot = 500,
  n_perm = 500,
  seed = NULL,
  polyserial_ml = FALSE,
  verbose = TRUE,
  nearPD = FALSE
)
```

Arguments

data	A data frame or tibble. All columns are used unless cols is specified.
cols	A character vector of column names to include. Default: all columns.
types	An optional named list mapping column names to types ("continuous", "count", "binary", "ordinal", "categorical"). Columns not listed are auto-detected.
assume_latent_normal	"auto" (default), logical, or NULL. Passed to each smart_cor() call. "auto" tests each pair individually via test_bivariate_normality(). NULL asks interactively once for all pairs.
ordinal_threshold	Integer passed to detect_type(). Default: 10.
ignore_na	Logical. If TRUE (default), missing values are handled pairwise (each pair uses its own complete observations). If FALSE, any pair with missing values fails; the failure is reported as a warning (when verbose = TRUE) and the cell is left NA.
conf_level	Numeric between 0 and 1. Confidence level passed to each smart_cor() call. Default: 0.95.
alpha	Significance level for the diagnostic tests, passed to each smart_cor() call. Default: 0.05.
bootstrap	Character or logical, passed to each smart_cor() call: "auto" (default), TRUE, or FALSE.
n_boot	Integer. Bootstrap replications per pair when bootstrapping. Default: 500.
n_perm	Integer. Permutation shuffles for the Theil's U and Freeman's theta p-values. Default: 500.
seed	Optional integer seed passed to each smart_cor() call; the global RNG state is saved and restored around seeded draws.
polyserial_ml	Logical, passed to each smart_cor() call. Default: FALSE; TRUE is recommended (see smart_cor()).
verbose	Logical. If TRUE (default), prints progress information. Set to FALSE for silent operation (useful inside scripts).
nearPD	Logical. If TRUE, adjust the completed correlation matrix with Matrix::nearPD(). Default: FALSE.

Details

For each pair of columns, smart_cor() detects variable types and selects the appropriate method. The result is a symmetric matrix of correlation/association measure estimates annotated with the method used for each cell.

Since different methods are used for different pairs, the resulting matrix may not be positive semi-definite. With nearPD = TRUE, the completed matrix is adjusted using Matrix::nearPD(corr = TRUE). The returned correlations are then positive definite with diagonal 1. original_correlations retains the estimates before adjustment; methods, details, and tidy.smartcormat() still describe those original estimates and their inference. The adjustment does not refit the pairwise models. A matrix containing missing or infinite entries cannot be adjusted.

Diagonal cells hold 1 by definition; the methods matrix records for each diagonal cell the method that a self-pair of that type would get, so every cell of the matrix is labelled the same way: `"pearson"` for continuous and count variables, `"phi"`, `"tetrachoric"`, or `"yules_q"` for binary variables (under `assume_latent_normal = "auto"`, `TRUE`, or `FALSE`), `"polychoric"` (`"auto"` or `TRUE`) or `"kendall"` (`FALSE`) for ordinal variables, and `"cramers_v"` for categorical variables.

The stored `details` omit each pair's `diagnostics` list to keep the object small; the summary fields `outlier_flag`, `normality_p`, `latent_p`, and `linearity_p` remain, so `tidy.smartcormat()` reports them.

Value

An object of class `"smartcormat"`: a list with elements:

`correlations` Numeric matrix of correlation estimates.

`methods` Character matrix of method codes used for each pair.

`types` Named character vector of detected/specified variable types.

`details` List of `smart_cor()` result objects (one per pair), without their `diagnostics` element.

`original_correlations` Matrix before adjustment; present only when `nearPD = TRUE`.

`nearPD` Adjustment diagnostics (`normF`, `iterations`, `converged`); present only when `nearPD = TRUE`.

Examples

```
path = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(path)

smart_cormat(
  gss[, c("age", "coninc", "degree", "sex")],
  assume_latent_normal = FALSE,
  verbose = FALSE
)

# Override a detected type when the study design requires it.
smart_cormat(
  gss[, c("age", "coninc", "degree")],
  types = list(age = "continuous", degree = "ordinal"),
  assume_latent_normal = FALSE,
  verbose = FALSE
)
```

---

smart_cor_df *Correlation Matrix as Plain Data Frames*

Description

A thin wrapper around `smart_cormat()` for pipeline work. It never prompts, prints nothing, and returns two plain data frames (the estimates and the method used for each pair) instead of an S3 object.

Usage

```
smart_cor_df(
  data,
  cols = NULL,
  types = NULL,
  assume_latent_normal = "auto",
  ordinal_threshold = 10,
  ignore_na = TRUE,
  conf_level = 0.95,
  alpha = 0.05,
  bootstrap = "auto",
  n_boot = 500,
  n_perm = 500,
  seed = NULL,
  polyserial_ml = FALSE,
  nearPD = FALSE
)
```

Arguments

data	A data frame or tibble.
cols	Optional character vector of column names to include. Default: all columns.
types	Optional named list mapping column names to types ("continuous", "count", "binary", "ordinal", "categorical"). Columns not listed are auto-detected.
assume_latent_normal	"auto" (default) or logical, passed to smart_cormat(). NULL is not useful here because this wrapper never prompts.
ordinal_threshold	Integer passed to detect_type(). Default: 10.
ignore_na	Logical. If TRUE (default), missing values are handled pairwise. If FALSE, pairs with missing values come back NA.
conf_level	Numeric between 0 and 1. Confidence level passed through to each smart_cor() call. Default: 0.95.
alpha	Significance level for the diagnostic tests, passed through to each smart_cor() call. Default: 0.05.
bootstrap	Character or logical, passed through: "auto" (default), TRUE, or FALSE.
n_boot	Integer. Bootstrap replications per pair. Default: 500.
n_perm	Integer. Permutation shuffles for the Theil's U and Freeman's theta p-values. Default: 500.
seed	Optional integer seed for the bootstrap and permutation draws; the global RNG state is saved and restored.
polyserial_ml	Logical, passed to smart_cormat(). Default: FALSE; TRUE is recommended (see smart_cor()).

nearPD — Logical, passed to `smart_cormat()`. Default: `FALSE`. If `TRUE`, `correlations` contains the positive-definite adjustment; `methods` describes the original pairwise estimates. Use `smart_cormat()` to retain the original matrix and adjustment diagnostics.

Details

By default it uses `assume_latent_normal = "auto"`, so each pair is screened with the diagnostics described in `smart_cor()` and the method follows them (a saturated 2 x 2 pair gets phi).

The two returned data frames share identical row and column names (the variable names from `data`). Diagonal entries in `correlations` are 1; diagonal entries in `methods` name the method a self-pair of that type would get, matching `smart_cormat()`.

Value

A named list with two elements:

`correlations` A data frame of correlation estimates (numeric).

`methods` A data frame of method codes (character) used for each pair.

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)

# Correlations and methods for four columns in the frozen GSS extract
result = smart_cor_df(
  gss[, c("age", "coninc", "degree", "sex")],
  assume_latent_normal = FALSE
)
result$correlations
result$methods

# Missing values are handled pairwise by default
example = gss[, c("age", "coninc", "sex")]
example$coninc[1:3] = NA
smart_cor_df(example, assume_latent_normal = FALSE)
```

---

test_bivariate_normality

Test Bivariate Normality for a Pair of Variables

Description

Tests the normality assumption behind Pearson's inference and behind the latent-variable correlations (polychoric, polyserial, biserial). The test depends on the variable types:

continuous + continuous Mardia's multivariate skewness and kurtosis tests, via `MVN::mardia()`.

continuous + ordinal Jarque-Bera on the continuous variable (`moments::jarque.test()`), plus the LR chi-square threshold-model test after binning the continuous variable.

continuous + binary Jarque-Bera on the residuals of the continuous variable around its group means, plus the LR chi-square test after binning (df = m1 - 2).

ordinal or binary pairs The LR chi-square test. A binary-binary table is saturated (df = 0) and cannot be tested.

Count variables are treated as continuous. Categorical variables are not supported.

Usage

```
test_bivariate_normality(
  x,
  y = NULL,
  x_type = NULL,
  y_type = NULL,
  alpha = 0.05,
  max_levels = 10L
)
```

Arguments

`x`, `y`	Vectors of the same length. Alternatively, a pre-computed contingency table (matrix or table object) can be passed as `x` with `y = NULL`, which runs the LR chi-square test on that table.
`x_type`, `y_type`	Optional variable types (`"continuous"`, `"count"`, `"binary"`, or `"ordinal"`). Detected with `detect_type()` when `NULL`.
`alpha`	Significance level for the test. Default: `0.05`.
`max_levels`	Integer. Cap on the number of categories per variable when building the contingency table from numeric vectors. A numeric variable with more than `max_levels` distinct values (e.g. a continuous or count variable) is binned into `max_levels` quantile groups before the table is formed. This keeps the maximum-likelihood threshold model tractable: an unbinned continuous variable would induce one threshold per distinct value, making the ML fit prohibitively slow and the chi-square reference distribution meaningless. Genuine ordinal/binary inputs (at most `max_levels` distinct values) are never altered. Default: `10`.

Details

The LR chi-square test follows Joreskog (2005). The bivariate normal threshold model is fitted with the two-step estimator (Olsson 1979) via `polycor::polychor()`: the thresholds come from the marginal proportions and the correlation is then estimated by one-dimensional maximum likelihood given those thresholds. The likelihood-ratio statistic $G^2 = 2 \sum_{ij} O_{ij} \log(O_{ij}/E_{ij})$ compares the observed counts O_{ij} with the expected counts E_{ij} under the fitted model (cells with zero observed count are skipped). The degrees of freedom equal $m_1 \times m_2 - m_1 - m_2$, where m_1 and m_2 are the numbers of categories of the two variables. For a 2×2 table the model is saturated (df = 0) and no test is possible.

When several component tests are run, the overall test rejects if any component rejects at `alpha`, and the overall `p.value` is the smallest component p-value. The Python pysmartcor package computes the same statistics, so the two implementations agree.

Value

An object of class `"smartcor_normtest"`, a list with elements:

statistic The LR chi-square statistic.

df Degrees of freedom of the LR chi-square test.

p.value The overall p-value (smallest component p-value).

rho The two-step estimate of the polychoric correlation.

row_thresholds Estimated thresholds for the first variable.

col_thresholds Estimated thresholds for the second variable.

n Total number of observations.

m1 Number of categories for the first variable.

m2 Number of categories for the second variable.

alpha The significance level used.

conclusion One of `"reject"`, `"fail_to_reject"`, or `"saturated"`.

test Name of the test, e.g. `"Mardia"`, `"Jarque-Bera and LR chi-square"`, or `"LR chi-square"`.

components Data frame with one row per component test and columns `test`, `statistic`, `df`, and `p.value`.

null_hypothesis The null hypothesis in words.

interpretation One plain sentence stating the result.

`statistic`, `df`, `rho`, the thresholds, `m1`, and `m2` describe the LR chi-square component and are `NA` (or empty) when it was not run.

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)

# Check the latent-normality assumption for two ordinal variables
test_bivariate_normality(gss$degree, gss$happy)

# Mardia's test for two continuous variables
test_bivariate_normality(gss$age, gss$coninc, x_type = "continuous",
                         y_type = "continuous")
```

test_linearity *Test Linearity Using Ordered Regression Residuals*

Description

Runs the sorted-residual diagnostics in both directions: `y` regressed on `x` with the residuals ordered by `x`, and `x` regressed on `y` with the residuals ordered by `y`. The relationship is treated as nonlinear only when both directions reject, so the result does not depend on which variable is passed first.

Usage

```
test_linearity(x, y, alpha = 0.05, ignore_na = TRUE)
```

Arguments

`x`, `y`	Numeric vectors of the same length.
`alpha`	Significance level, strictly between 0 and 1. Default: `0.05`.
`ignore_na`	Remove incomplete pairs when `TRUE` (default); otherwise stop if either vector contains a missing value. Infinite values are rejected.

Details

In each direction, incomplete pairs are removed, the regression is fitted, and the residuals are ordered by the regressor. Residuals that share a regressor value are averaged, so the order of tied rows does not matter. Box-Pierce and Ljung-Box tests use lag 1. The BDS p-value is the smallest of the p-values from `fNonlinear::bdsTest(residuals, m = 3)` across its four default distance thresholds. Breusch-Pagan and RESET use the defaults in the lmtest package.

A direction rejects linearity when at least one residual structure test rejects and RESET rejects. Its p-value is `max(min(p_bp, p_lb, p_bds), p_reset)`, the intersection-union p-value of that rule. Spearman is selected only when both directions reject; the combined `p.value` is the larger of the two directional p-values (`p_y_on_x`, `p_x_on_y`), so `p.value < alpha` exactly when linearity is rejected. Otherwise Pearson is selected, and the inference recommendation (standard or robust/bootstrap) comes from the direction that does not reject; when both do not reject, from the one with the larger p-value. The component p-values reported at the top level belong to that direction (`direction`).

The regression model behind the test assumes errors independent of the regressor, which can hold in only one direction unless the pair is bivariate normal. Requiring both directions to reject keeps skewed margins from being read as nonlinearity.

`linear = TRUE` means this rule did not detect nonlinearity; it does not establish linearity. Fewer than five distinct values, a constant variable, an exact linear fit, or undefined diagnostics make a direction inconclusive. If neither direction can be tested, or one rejects and the other cannot be tested, the result is inconclusive (`linear = NA`) and Pearson is retained. This function returns correlation estimates and an inference recommendation; it does not compute confidence intervals or correlation p-values.

Value

An object of class "smartcor_lintest": a list with p_bp, p_lb, p_bds, p_het, p_reset (for the direction used), p_y_on_x, p_x_on_y, direction, p.value (combined), alpha, n_complete, structure_detected, homoskedastic, linear, conclusion, decision, method, r_pearson, r_spearman, r_kendall, r_final, reason, null_hypothesis, and interpretation.

Examples

```
set.seed(42)
x = runif(100, 0, 2)
y = 2 + 3 * x + rnorm(100, sd = 0.5)
test_linearity(x, y)
```

---

tidy.smartcor *Tidy a smartcor Object*

Description

Lays a smart_cor() result out as a one-row tibble in the manner of the **broom** package, so results drop straight into a dplyr pipeline.

Usage

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

Arguments

x	A smartcor object.
...	Ignored.

Value

A one-row tibble::tibble() with columns covering identifiers (var_x, var_y, x_type, y_type), the point estimate, the chosen method (method, method_label, rationale), inference details (statistic, p.value, p_method, null_hypothesis, p_interpretation, ci_lower, ci_upper, conf_level, ci_method, ci_source), sample sizes (n, n_complete), and diagnostic summaries (outlier_flag, normality_p, latent_p, linearity_p; NA when not computed).

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)
res = smart_cor(gss$coninc, gss$age, verbose = FALSE)
tidy(res)
```

tidy.smartcormat *Tidy a smartcormat Object*

Description

Turns a `smart_cormat()` result into a long tibble with one row per variable pair (lower triangle only, no self-correlations). If `nearPD = TRUE` was used, estimates and inference refer to the original pairs, before the matrix adjustment.

Usage

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

Arguments

`x`	A `smartcormat` object.
`...`	Ignored.

Value

A `tibble::tibble()` with the same columns as `tidy.smartcor()`, one row per unique pair.

Examples

```
csv = system.file("extdata", "gss_2024_casestudy.csv", package = "smartcor")
gss = read.csv(csv)
mat = smart_cormat(
  gss[, c("age", "coninc", "sex")],
  assume_latent_normal = FALSE,
  verbose = FALSE
)
tidy(mat)
```

tidy.smartcor_comparison *Tidy a smartcor Comparison*

Description

Lays a `compare_methods()` result out as a tibble with one row per candidate method, in the manner of the **broom** package.

Usage

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

Arguments

x	A `smartcor_comparison` object.
...	Ignored.

Value

A `tibble::tibble()` with one row per method.

Index

available_methods, 2
available_methods(), *4*, *13*

compare_methods, 3
compare_methods(), *11*, *25*

detect_outliers, 5
detect_outliers(), *15, 16*
detect_type, 6
detect_type(), *3*, *14*, *17*, *19*, *21*

ggcor, 7
ggcor_heatmap *(*ggcor*)*, 7
ggcor_method_heatmap *(*ggcor*)*, 7

Matrix::nearPD(), *17*
moments::jarque.test(), *15*, *21*
MVN::mardia(), *15*, *20*

plot.smartcormat, 8
plot_cor_heatmap *(*plot.smartcormat*)*, 8
polycor::polychor(), *21*
polycor::polyserial(), *14*
print.smartcor, 10
print.smartcor(), *16*
print.smartcor_comparison, 11
print.smartcor_lintest, 12
print.smartcor_normtest, 12
print.smartcor_outliers
*(*detect_outliers*)*, 5
print.smartcormat, 11

smart_cor, 13
smart_cor(), *2–5*, *16–20*, *24*
smart_cor_df, 18
smart_cor_df(), *7*, *9*
smart_cormat, 16
smart_cormat(), *4*, *7*, *9*, *18–20*, *25*

test_bivariate_normality, 20
test_bivariate_normality(), *12, 13*, *15*, *17*
test_linearity, 23
test_linearity(), *12*, *14–16*
tibble::tibble(), *4*, *24–26*
tidy.smartcor, 24
tidy.smartcor(), *16*, *25*
tidy.smartcor_comparison, 25
tidy.smartcormat, 25
tidy.smartcormat(), *17, 18*