Package 'corsym'

September 17, 2026

Title Correlation Estimation for Exchangeable/Symmetrical Variables

Version 1.0.1

Description We implement a new correlation estimator, CorSym, designed for exchangeable variables, where the ordering of the two values in the pair is arbitrary. This kind of data arises frequently in the study of assortative pairing (for example, income in a couple). The standard Pearson estimator is sensitive to such ordering and can be highly biased when the order is biased (when the first value tends to have lower or higher values than the second value). CorSym gives the same estimate deterministically for all orders within each pair, and estimates the desired correlation without bias (variables must be exchangeable). The package also includes utilities to simulate biased orders and test for order bias. Described in Kennedy and Ochoa (2026) <doi:10.64898/2026.08.22.746446>.

License GPL (>= 3)

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ggplot2

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/OchoaLab/corsym

BugReports https://github.com/OchoaLab/corsym/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Alejandro Ochoa [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4928-3403>),
Gabriel Kennedy [aut] (ORCID: <https://orcid.org/0000-0003-4412-3777>)

Maintainer Alejandro Ochoa <alejandro.ochoa@duke.edu>

Repository CRAN

Date/Publication 2026-09-17 11:50:02 UTC

Contents

`add_cor_CI` *Calculate confidence intervals (CIs) for correlation estimates*

Description

This function calculates CIs based on the Fisher transformation and theory originally derived for Pearson estimator, now validated empirically for CorSym. Only the mean estimates and the sample sizes are needed for these calculations.

Usage

```
add_cor_CI(r, n, alpha = 0.05, ci_offset = 3)
```

Arguments

`r`	Correlation estimates. Can be scalar or vector, each non-NA value must be between -1 and 1, inclusive. Cases with NA `r` have all `NA` outputs (means and CIs).
`n`	Sample sizes. Can be scalar or vector. If both `r` and `n` are vectors, they must have the same length. Cases with NAs or with `n` < `ci_offset` have NA CIs.
`alpha`	The complement of the confidence level (default 0.05 results in 1 - 0.05 = 95% CIs).
`ci_offset`	The amount by which the sample size gets reduced in the variance conversion formula. For Normal data, Pearson should use 3 and CorSym 1.5, but different values may be better for other distributions.

Value

A matrix with a row for every input (single row if `r` and `n` are both scalars, as many rows as the lengths of either `r` or `n` otherwise), and three named columns: `r` repeats the input, and `CIL` and `CIU` are the lower and upper CIs.

See Also

`corsym()` and `pearson()` use this function internally.

Examples

```
# make up values for example
# a vector of pre-calculated correlation estimates
rs <- c( 0.3, 0.5, 0.9 )
# and sample sizes
ns <- c( 10, 100, 1000 )

# calculate CIs for these three cases:
out <- add_cor_CI( rs, ns )
```

`corsym` *Calculate CorSym estimates with confidence intervals (CIs)*

Description

This function estimates the correlation of two exchangeable variables without bias. The output is deterministic, the same for all orders within individual pairs. Pairs with any missingness are automatically ignored.

Usage

```
corsym(x, y = NULL, alpha = 0.05, ci_offset = 1.5)
```

Arguments

`x`	First vector of variables, or a 2-column matrix with the variables of interest
`y`	Second vector of variables
`alpha`	The complement of the confidence level (default 0.05 results in 1 - 0.05 = 95% CIs).
`ci_offset`	The amount by which the sample size gets reduced in the variance conversion formula. For Normal data, Pearson should use 3 and CorSym 1.5, but different values may be better for other distributions.

Value

A vector with three values: the CorSym correlation estimate between `x` and `y`, and the lower and upper CIs.

Examples

```
# generate some random independent data, just for example
n <- 53
x <- rnorm( n )
y <- rnorm( n )
# calculate CorSym correlation with CIs
corsym( x, y )
```

order_bias_test *Test for order bias*

Description

This function performs a test for whether data is appearing in a biased order or not. The test calculates a two-sided Binomial p-value under the null hypothesis that the true frequency of the counted order is `f0` (0.5 by default).

Usage

```
order_bias_test(data, f0 = 0.5)
```

Arguments

`data`	A data.frame (including tibble) with at least two columns: `n` the sample size (number of pairs), and `x` the number of pairs in a given order.
`f0`	The expected frequency of pairs in the given order, under the null hypothesis.

Value

The input data frame with two more columns: `f` is the sample order frequency (equal to `x/n`), and `pval` the two-sided Binomial p-value.

Examples

```
# toy data used in the test
data <- data.frame(
  n = c( 30, 50, 100 ),
  x = c( 12, 20,  13 )
)

# add frequencies and p-values!
data <- order_bias_test( data )
```

partial_order *Apply partial ordering to the rows of a matrix*

Description

This function reorders a proportion of rows so values appear in increasing order.

Usage

```
partial_order(X, q = 1)
```

Arguments

X	Matrix of data.
q	Proportion of rows to force to have extreme order (sorted in increasing order).

Value

The input matrix with a random proportion q of rows sorted in increasing order, the rest equals the input data.

Examples

```
# dummy independent data in random order
X <- matrix( rnorm( 20 ), nrow = 10, ncol = 2 )

# same data but in extreme order by default
X_extreme <- partial_order( X )

# apply a partial ordering
X_partial <- partial_order( X, 0.5 )
```

---

pearson *Calculate Pearson estimates with confidence intervals (CIs)*

Description

This function is a simple wrapper around `stats::cor()` with the same interface and missingness handling of `corsym()`, particularly it adds CIs to the estimates. Pairs with any missingness are automatically ignored (matching `stats::cor()` with non-default option `use = 'complete.obs'`). Also, unlike `stats::cor()`, this function accepts only two variables as inputs and does not calculate correlation matrices, just the correlation between the two variables. Pearson is included in this package to contrast to `corsym()` in handling exchangeable variables.

Usage

```
pearson(x, y = NULL, alpha = 0.05, ci_offset = 3)
```

Arguments

x	First vector of variables, or a 2-column matrix with the variables of interest
y	Second vector of variables
alpha	The complement of the confidence level (default 0.05 results in 1 - 0.05 = 95% CIs).
ci_offset	The amount by which the sample size gets reduced in the variance conversion formula. For Normal data, Pearson should use 3 and CorSym 1.5, but different values may be better for other distributions.

Value

A vector with three values: the Pearson correlation estimate between `x` and `y`, and the lower and upper CIs.

Examples

```
# generate some random independent data, just for example
n <- 53
x <- rnorm( n )
y <- rnorm( n )
# calculate Pearson correlation with CIs
pearson( x, y )
```

---

randomize_order *Randomize ordering of each row of a matrix*

Description

This function shuffles the values in each row, so they appear in a random order.

Usage

```
randomize_order(X)
```

Arguments

`X`	Matrix of data.

Value

The matrix with randomized rows.

Examples

```
# dummy data in order
X <- matrix( 1:20, nrow = 10, ncol = 2, byrow = TRUE )

# same data but with each row in random order
X_randomized <- randomize_order( X )
```

Index