Package 'netcutter'

May 9, 2026

Type Package

Title Identification and Analysis of Co-Occurrence Networks

Version 0.3.1

Maintainer Federico Marotta <federico.marotta@embl.de>

Description Implementation of the NetCutter algorithm described in Müller and Mancuso (2008) <doi:10.1371/journal.pone.0003178>. The package identifies co-occurring terms in a list of containers. For example, it may be used to detect genes that co-occur across genomes.

URL https://doi.org/10.1371/journal.pone.0003178

BugReports https://github.com/fmarotta/netcutter/issues

License MIT + file LICENSE

Encoding UTF-8

Imports PoissonBinomial, rlecuyer,

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

Config/testthat/edition 3

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Heiko Müller [aut],
Francesco Mancuso [aut],
Federico Marotta [cre]

Repository CRAN

Date/Publication 2025-05-21 15:50:06 UTC

Contents

`nc_define_modules` *Define co-occurrence modules*

Description

Helper function to generate the list of co-occurrence terms grouped into modules of a specified size.

Usage

```
nc_define_modules(occ_matrix, terms_of_interest, module_size, min_occurrences)
```

Arguments

`occ_matrix`	The original occurrence matrix.
`terms_of_interest`	Vector of column names or indices representing the terms that should be included in the analysis.
`module_size`	The number of terms that should be tested for co-occurrence.
`min_occurrences`	Minimum number of occurrences of each term.

Value

A list of the valid modules.

`nc_eval` *Compute co-occurrence probabilities*

Description

The main NetCutter function. It generates p-values for all the co-occurring modules.

Usage

```
nc_eval(
  occ_matrix,
  occ_probs,
  terms_of_interest = NULL,
  module_size = 2,
  min_occurrences = 0,
  min_support = 0,
  mc.cores = 1
)
```

Arguments

occ_matrix	The original occurrence matrix.
occ_probs	The matrix of occurrence probabilities, as computed by nc_occ_probs().
terms_of_interest	Vector of column names or indices representing the terms that should be included in the analysis.
module_size	The number of terms that should be tested for co-occurrence.
min_occurrences	Minimum number of occurrences of each term.
min_support	Minimum number of occurrences of each module.
mc.cores	Number of parallel computations with mclapply() (set to 1 for serial execution)

Details

If terms_of_interest is NULL, all the terms in occ_matrix are used. If it is not null, only modules containing at least one of these terms will be considered. min_occurrences and min_support are still used to further restrict the list of terms that are considered.

Value

A data.frame with one row for each valid module, and corresponding number of co-occurrences and p-value.

Examples

```
# Generate an occurrence matrix.
m <- matrix(FALSE, 3, 9, dimnames = list(paste0("ID", 1:3), paste0("gene", 1:9)))
m[1, 1:3] <- m[2, c(1:2, 4:5)] <- m[3, c(1, 6:9)] <- TRUE
# Set the seed using the "L'Ecuyer-CMRG" random number generator.
set.seed(1, "L'Ecuyer-CMRG")
# Compute the occurrence probabilities.
occ_probs <- nc_occ_probs(m, R = 20, S = 50)
# Evaluate the co-occurrences of pairs of terms and their statistical significance.
nc_eval(m, occ_probs, module_size = 2)
# Now evaluate triples; no need to recompute the occurrence probabilities.
nc_eval(m, occ_probs, module_size = 3)
```

```
# Now consider only modules involving gene1 or gene2.
nc_eval(m, occ_probs, module_size = 2, terms_of_interest = c("gene1", "gene2"))
```

`nc_occ_probs` *Compute the occurrence probabilities*

Description

Use the EdgeSwapping method to find the probability of occurrence of each term in each container under the null hypothesis.

Usage

```
nc_occ_probs(
  occ_matrix,
  R = 500,
  S = sum(occ_matrix) * 10,
  mc.cores = getOption("mc.cores", 1L),
  n_batches = ceiling(R/30),
  verbose = FALSE
)
```

Arguments

`occ_matrix`	The original co-occurrence matrix
`R`	The number of randomisations to perform
`S`	The number of successful edge swaps for each randomisation
`mc.cores`	Number of parallel computations with mclapply() (set to 1 for serial execution)
`n_batches`	Split the computation into `n_batches` to avoid excessive memory usage
`verbose`	Print a status message when starting every new batch.

Value

The occurrence probability matrix.

Examples

```
# Generate an occurrence matrix.
m <- matrix(FALSE, 3, 9, dimnames = list(paste0("ID", 1:3), paste0("gene", 1:9)))
m[1, 1:3] <- m[2, c(1:2, 4:5)] <- m[3, c(1, 6:9)] <- TRUE
# Set the seed using the `rlecuyer` package
rlecuyer::.lec.SetPackageSeed(1:6)
# Compute the occurrence probabilities.
occ_probs <- nc_occ_probs(m, R = 20, S = 50)
# Using `n_batches=1` can speed up the computations at the cost of more RAM.
occ_probs <- nc_occ_probs(m, R = 20, n_batches = 1, mc.cores = 1)
```

nc_occ_probs_simple *Compute the occurrence probabilities*

Description

This is a simpler implementation used to check that the official implementation (nc_occ_probs()) works well.

Usage

```
nc_occ_probs_simple(occ_matrix, R, S)
```

Arguments

occ_matrix	The original co-occurrence matrix
R	The number of randomisations to perform
S	The number of successful edge swaps for each randomisation

nc_randomize *Randomize the occurrence matrix*

Description

Apply an edge-swapping algorithm.

Usage

```
nc_randomize(occ_matrix, S)
```

Arguments

occ_matrix	The original occurrence matrix.
S	The number of successful edge swaps to perform.

Value

A randomized copy of the occurrence matrix.

nc_randomize_fast *Randomize the occurrence matrix*

Description

Faster implementation that samples row and column independently

Usage

```
nc_randomize_fast(occ_matrix, S)
```

Arguments

occ_matrix	The original occurrence matrix.
S	The number of successful edge swaps to perform.

nc_randomize_R *Randomize the occurrence matrix*

Description

Old implementation in pure R, kept for testing purposes and for reproducibility of old results.

Usage

```
nc_randomize_R(occ_matrix, S)
```

Arguments

occ_matrix	The original occurrence matrix.
S	The number of successful edge swaps to perform.

nc_randomize_simple *Randomize the occurrence matrix*

Description

This is a simpler implementation used to check that the official implementation (nc_randomize()) works well.

Usage

```
nc_randomize_simple(occ_matrix, S)
```

Arguments

occ_matrix	The original occurrence matrix.
S	The number of successful edge swaps to perform.

safe_sample *Sample one item from a vector, even when the vector has length 1*

Description

Sample one item from a vector, even when the vector has length 1

Usage

```
safe_sample(x)
```

Arguments

x	Vector of values to sample

Details

When x has length 1, the sample() function thinks that we want to sample from 1 to x. However, we deal want to sample vectors of unknown length, and possibly of length 1, but we always want to sample among the values of x. This function ensures that.

Value

One value from x.

Index