Package: TriGer (via r-universe)

November 23, 2026

Type Package

Title Structured Cross-Omics Interaction Discovery with a Triple-Graph Model

Version 0.99.0

Description Detects modules of features that are densely associated across two omics layers and coherent within each layer. Dense cross-omics blocks are found by greedy peeling of the cross-omics association matrix, and large blocks are refined by greedy peeling of the within-omics correlation matrices. Input can be numeric matrices or SummarizedExperiment objects. Plotting functions show the detected modules alongside the within-omics correlation structure. The method is described in Yu, Lin and Chen (2026) <doi:10.64898/2026.09.10.749540>.

License MIT + file LICENSE

Depends R (>= 4.5.0)

Imports graphics, grDevices, methods, stats, utils, SummarizedExperiment

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

VignetteBuilder knitr

biocViews Software, Clustering, GraphAndNetwork, Network, Transcriptomics, Metabolomics, Microbiome, SystemsBiology, Visualization

URL https://github.com/MATRIX-Research-Lab/TriGer

BugReports https://github.com/MATRIX-Research-Lab/TriGer/issues

Encoding UTF-8

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-11-22 21:11:45 UTC

RemoteUrl https://github.com/BiocStaging/TriGer

RemoteRef HEAD

RemoteSha 72e59323299930eb158087fe7155fbaad0fffae4

Contents

greedy_3step 2
greedy_peeling_X_all 4
greedy_peeling_X_one 4
greedy_peeling_XY_all 5
greedy_peeling_XY_one 6
matlab_jet 6
plot_all_result 7
plot3in1_v1 8
sort_result 9
triger 10
triger_association 11

Index **13**

`greedy_3step` *Detect cross-omics modules from association matrices*

Description

The TriGer pipeline on precomputed matrices. Step 1 peels dense blocks from the cross-omics part of `P` with `greedy_peeling_XY_all()`. Blocks with `X_size[1]` or fewer X features, or `Y_size[1]` or fewer Y features, are dropped. In steps 2 and 3, the X and Y sides of blocks with more than `X_size[2]` or `Y_size[2]` features are refined by within-omics peeling of the signed correlations in `R`: the side is replaced by the communities with at least `min_community_size` features, each ordered by within- community sums. `triger()` computes `P` and `R` from data and calls this function.

Usage

```
greedy_3step(
  P,
  R,
  m,
  n,
  lambda1 = 1.5,
  lambda2 = 1.5,
  lambda3 = 1.5,
  X_size = c(10, 200),
  Y_size = c(10, 200),
  min_community_size = 2,
```

```
  keep_unrefined_if_empty = TRUE,
  max_blocks = 5
)
```

Arguments

`P`	`(m + n) x (m + n)` non-negative association matrix over the X and Y features (for example -log10 p-values of their correlations); only `P[1:m, (m + 1):(m + n)]` is used.
`R`	`(m + n) x (m + n)` signed correlation matrix over the X and Y features, used for refinement.
`m`, `n`	Numbers of X and Y features.
`lambda1`	Cross-omics size penalty (step 1).
`lambda2`, `lambda3`	Within-omics size penalties for X and Y (steps 2 and 3).
`X_size`, `Y_size`	`c(min, max)`: blocks with `min` or fewer features on either side are dropped, and sides with more than `max` features are refined.
`min_community_size`	Smallest within-omics community kept during refinement.
`keep_unrefined_if_empty`	If refinement keeps no community, keep that side of the block unrefined (`TRUE`) or empty it (`FALSE`).
`max_blocks`	Maximum number of blocks found in step 1.

Value

A list of modules, each `list(X, Y, density)`, with `X` in `1:m`, `Y` in `1:n`, and `density` the penalized density of the step-1 block before refinement.

Examples

```
set.seed(1)
N <- 150
m <- 80
n <- 120
f <- rnorm(N)
X <- matrix(rnorm(N * m), N)
Y <- matrix(rnorm(N * n), N)
X[, 1:25] <- X[, 1:25] + 1.5 * f
Y[, 1:40] <- Y[, 1:40] + 1.5 * f
R <- cor(cbind(X, Y))
P <- abs(R)
diag(P) <- 0
modules <- greedy_3step(P, R, m, n, lambda1 = 1.1, lambda2 = 1.8,
    lambda3 = 1.8)
lengths(modules[[1]][c("X", "Y")])
```

greedy_peeling_X_all *Peel dense communities from a within-omics matrix repeatedly*

Description

Applies `greedy_peeling_X_one()` repeatedly, removing each accepted community before the next round, while the community's penalized density exceeds that of the full matrix.

Usage

```
greedy_peeling_X_all(Wp, lam, min_size = 1)
```

Arguments

`Wp`	Symmetric matrix of within-omics associations, for example signed correlations.
`lam`	Size penalty, typically in [1, 2]. Higher values give smaller, denser blocks.
`min_size`	Smallest community returned. Communities with fewer features are peeled but not returned. The default, 1, returns every community.

Value

A list of communities, each `list(X, density)`, with indices into the rows of `Wp`.

Examples

```
set.seed(1)
S <- matrix(runif(50 * 50, -0.2, 0.2), 50, 50)
S[1:12, 1:12] <- S[1:12, 1:12] + 0.9
S <- (S + t(S)) / 2
comms <- greedy_peeling_X_all(S, 1.5)
comms[[1]]$X
```

greedy_peeling_X_one *Peel one dense community from a within-omics matrix*

Description

Greedily removes the feature with the smallest remaining sum and keeps the intermediate subset with the highest penalized density, `sum / size^lam`.

Usage

```
greedy_peeling_X_one(Wp, lam)
```

Arguments

Wp	Symmetric matrix of within-omics associations, for example signed correlations.
lam	Size penalty, typically in [1, 2]. Higher values give smaller, denser blocks.

Value

A list with density and X (indices of the community).

Examples

```
set.seed(1)
S <- matrix(runif(50 * 50, -0.2, 0.2), 50, 50)
S[1:12, 1:12] <- S[1:12, 1:12] + 0.9
S <- (S + t(S)) / 2
greedy_peeling_X_one(S, 1.5)$X
```

---

greedy_peeling_XY_all *Peel dense blocks from a cross-omics matrix repeatedly*

Description

Applies greedy_peeling_XY_one() repeatedly, removing the rows and columns of each accepted block before the next round. A block is accepted while its penalized density exceeds that of the full matrix, up to max_blocks blocks.

Usage

```
greedy_peeling_XY_all(XY, lam = 1.5, max_blocks = 5)
```

Arguments

XY	Non-negative cross-omics association matrix.
lam	Size penalty, typically in [1, 2]. Higher values give smaller, denser blocks.
max_blocks	Maximum number of blocks returned.

Value

A list of blocks, each list(X, Y, density), with indices into the rows (X) and columns (Y) of XY.

Examples

```
set.seed(1)
W <- matrix(runif(60 * 80, 0, 0.5), 60, 80)
W[6:20, 11:30] <- W[6:20, 11:30] + 4
blocks <- greedy_peeling_XY_all(W, 1.5)
lengths(blocks[[1]][c("X", "Y")])
```

greedy_peeling_XY_one *Peel one dense block from a cross-omics matrix*

Description

Greedily removes the X feature (row) or Y feature (column) with the weakest remaining association and keeps the intermediate submatrix with the highest penalized density, `sum / (|X| * |Y|)^(lam / 2)`. A row is removed when its sum, scaled by the current ratio `|X| / |Y|`, is at most the smallest column sum; otherwise a column is removed.

Usage

```
greedy_peeling_XY_one(Wp, lam)
```

Arguments

`Wp`	Non-negative matrix of cross-omics associations (X features in rows, Y features in columns), for example -log10 p-values.
`lam`	Size penalty, typically in [1, 2]. Higher values give smaller, denser blocks.

Value

A list with `density` (the penalized density of the block), `X` (row indices) and `Y` (column indices).

Examples

```
set.seed(1)
W <- matrix(runif(60 * 80, 0, 0.5), 60, 80)
W[6:20, 11:30] <- W[6:20, 11:30] + 4
blk <- greedy_peeling_XY_one(W, 1.5)
range(blk$X)
range(blk$Y)
```

matlab_jet *Jet colour map*

Description

The jet colour map used by the plotting functions.

Usage

```
matlab_jet(m = 256)
```

Arguments

`m`	Number of colours.

Value

A character vector of `m` colours.

Examples

```
matlab_jet(8)
```

`plot_all_result` *Heatmap of detected blocks*

Description

Shows `Wp` reordered so that the features of the detected blocks come first. With `full = TRUE` the whole matrix is shown with one box around the selected region; otherwise only the selected region is shown, with one box per block.

Usage

```
plot_all_result(Wp, result, full = FALSE, line = TRUE, crange = NULL)
```

Arguments

`Wp`	The matrix the blocks were detected in.
`result`	A list of blocks, each with X indices first and Y indices second; for a symmetric `Wp`, only the first element is used.
`full`	Show the full matrix.
`line`	Draw boxes around the blocks.
`crange`	Colour limits; defaults to the range of the plotted values.

Value

Invisibly, the plotted matrix.

Examples

```
set.seed(1)
W <- matrix(runif(60 * 80, 0, 0.5), 60, 80)
W[6:20, 11:30] <- W[6:20, 11:30] + 4
plot_all_result(W, greedy_peeling_XY_all(W, 1.5), full = TRUE)
```

plot3in1_v1 *Plot detected modules with their within-omics correlations*

Description

Draws the cross-omics matrix restricted to the selected features as a heatmap, with the X-X and Y-Y matrices of the same features as rotated triangles on the top and right. Draws into the current graphics device; a 5.6 x 4.2 inch device matches the default layout.

Usage

```
plot3in1_v1(
  X,
  Y,
  XY,
  result,
  line = FALSE,
  step = NULL,
  bar = FALSE,
  figsize = c(5.6, 4.2),
  cbar_pos = c(0.15, 0.13, 0.013, 0.45),
  clim_tri = NULL,
  clim_mid = NULL,
  box_lw = 2
)
```

Arguments

`X`, `Y`	Within-omics matrices for the X and Y features.
`XY`	Cross-omics matrix (X features in rows).
`result`	A list of blocks, each with X indices first and Y indices second, as returned by `greedy_3step()` or `greedy_peeling_XY_all()`, or the `modules` element returned by `triger()`.
`line`	Draw a box around each block.
`step`	`c(x, y)` tick spacing; defaults to `c(100, 10)`.
`bar`	Add a colour bar for the middle panel.
`figsize`	Device size in inches, used to keep the panels square.
`cbar_pos`	Colour bar position `c(left, bottom, width, height)` in figure units, or `"right"` to place it beside the plot.
`clim_tri`, `clim_mid`	Colour limits for the triangles and the middle panel. By default `c(-1, 1)` for the triangles, and `c(0, 10)` for the middle panel if its maximum exceeds 1, otherwise `c(-1, 1)`.
`box_lw`	Line width of the block boxes, in points.

Value

Invisibly, a list with the plotted X and Y indices.

Examples

```
set.seed(1)
N <- 150
m <- 80
n <- 60
f <- rnorm(N)
X <- matrix(rnorm(N * m), N)
Y <- matrix(rnorm(N * n), N)
X[, 1:25] <- X[, 1:25] + 1.5 * f
Y[, 1:20] <- Y[, 1:20] + 1.5 * f
R <- cor(cbind(X, Y))
P <- abs(R)
diag(P) <- 0
modules <- greedy_3step(P, R, m, n, 1.1, 1.8, 1.8)
plot3in1_v1(R[1:m, 1:m], R[m + 1:n, m + 1:n], R[1:m, m + 1:n], modules,
    line = TRUE, step = c(10, 5))
```

---

sort_result *Order features within detected blocks*

Description

`sort_result()` orders the features of each block by their within-block sums in `Wp`; for a cross-omics `Wp`, rows and columns are ordered by the absolute values of their sums. `sort_result_XandY()` orders X features by their sums within `X[blk$X, blk$X]` and Y features by their sums within `Y[blk$Y, blk$Y]`.

Usage

```
sort_result(Wp, blocks, symmetric = NULL)

sort_result_XandY(X, Y, blocks)
```

Arguments

`Wp`	The matrix the blocks were detected in.
`blocks`	A list of blocks, as returned by `greedy_peeling_XY_all()`, `greedy_peeling_X_all()` or `greedy_3step()`.
`symmetric`	Whether `Wp` is a within-omics matrix. Defaults to `TRUE` for square matrices.
`X, Y`	Within-omics matrices for the X and Y features.

Value

`blocks` with the features of each block reordered.

Examples

```
set.seed(1)
W <- matrix(runif(60 * 80, 0, 0.5), 60, 80)
W[6:20, 11:30] <- W[6:20, 11:30] + 4
blocks <- sort_result(W, greedy_peeling_XY_all(W, 1.5))
head(blocks[[1]]$X)
```

`triger` *Detect cross-omics modules with TriGer*

Description

Runs the TriGer pipeline on two omics layers measured on the same samples. `triger_association()` computes the cross- and within-omics correlations and their -log10 p-values, and `greedy_3step()` finds blocks of X and Y features that are densely associated across the two layers and refines large blocks by within-omics peeling.

Usage

```
triger(
  x,
  y,
  lambda1 = 1.5,
  lambda2 = 1.5,
  lambda3 = 1.5,
  X_size = c(10, 200),
  Y_size = c(10, 200),
  method = c("pearson", "spearman"),
  assay_x = 1,
  assay_y = 1,
  ...
)
```

Arguments

`x`, `y`	The two omics layers: numeric matrices or data frames with samples in rows and features in columns, or SummarizedExperiment::SummarizedExperiment objects with features in rows and samples in columns.
`lambda1`	Cross-omics size penalty (step 1). Higher values give smaller, denser blocks.
`lambda2`, `lambda3`	Within-omics size penalties for X and Y (steps 2 and 3).
`X_size`, `Y_size`	`c(min, max)` passed to `greedy_3step()`: blocks with `min` or fewer features on either side are dropped, and sides with more than `max` features are refined.
`method`	Correlation coefficient: `"pearson"` or `"spearman"`.
`assay_x`, `assay_y`	Assay used when `x` or `y` is a `SummarizedExperiment`, by name or index.
`...`	Further arguments to `greedy_3step()`: `min_community_size`, `keep_unrefined_if_empty`, `max_blocks`.

Details

Samples are matched by name when both layers carry sample names (row names of a matrix, column names of a `SummarizedExperiment`); `y` is then reordered to the sample order of `x`. Without sample names, the samples must be in the same order.

Value

A list with

`modules` A list of modules, each `list(X, Y, density, X_names, Y_names)`, where `X` and `Y` index the features of `x` and `y` and `density` is the penalized density of the step-1 block.

`R`, `P`, `m`, `n` The output of `triger_association()`.

Examples

```
set.seed(1)
N <- 100
f <- rnorm(N)
x <- matrix(rnorm(N * 40), N, dimnames = list(NULL, paste0("g", 1:40)))
y <- matrix(rnorm(N * 30), N, dimnames = list(NULL, paste0("m", 1:30)))
x[, 1:12] <- x[, 1:12] + f
y[, 1:8] <- y[, 1:8] + f
res <- triger(x, y, lambda1 = 1.3, X_size = c(3, 50), Y_size = c(3, 50))
res$modules[[1]]$X_names
res$modules[[1]]$Y_names
```

`triger_association` *Association matrices for two omics layers*

Description

Computes the signed correlation matrix and the matrix of -log10 two-sided p-values over the features of two omics layers measured on the same samples. These are the inputs `R` and `P` of `greedy_3step()`.

Usage

```
triger_association(x, y, method = c("pearson", "spearman"))
```

Arguments

`x`	Numeric matrix of the first omics layer, samples in rows and features in columns.
`y`	Numeric matrix of the second omics layer, with the same samples in the same order.
`method`	Correlation coefficient: `"pearson"` or `"spearman"`.

Details

The p-value of a correlation coefficient r estimated from N samples is computed from $t = |r|\sqrt{(N-2)/(1-r^2)}$ with $N-2$ degrees of freedom, as in `stats::cor.test()` (for Spearman correlations, the same t approximation). r^2 is capped at $1 - 10^{-12}$ so that perfectly correlated features receive a large finite value, and the diagonal of `P` is set to 0.

Value

A list with `R` and `P`, both `(m + n) x (m + n)` with the `m` features of `x` first and the `n` features of `y` second, and `m` and `n`. Row and column names are the column names of `x` and `y`, or `X1, X2, ...` and `Y1, Y2, ...` when these are missing.

Examples

```
set.seed(1)
x <- matrix(rnorm(50 * 5), 50)
y <- matrix(rnorm(50 * 4), 50)
assoc <- triger_association(x, y)
round(assoc$P[1:5, 6:9], 2)
```

Index

greedy_3step, 2
greedy_3step(), *8–11*
greedy_peeling_X_all, 4
greedy_peeling_X_all(), *9*
greedy_peeling_X_one, 4
greedy_peeling_X_one(), *4*
greedy_peeling_XY_all, 5
greedy_peeling_XY_all(), *2*, *8*, *9*
greedy_peeling_XY_one, 6
greedy_peeling_XY_one(), *5*

matlab_jet, 6

plot3in1_v1, 8
plot_all_result, 7

sort_result, 9
sort_result_XandY *(*sort_result*)*, 9
stats::cor.test(), *12*
SummarizedExperiment::SummarizedExperiment, *10*

triger, 10
triger(), *2*, *8*
triger_association, 11
triger_association(), *10*, *11*