Package 'mosbi'

April 12, 2022

Title Molecular Signature identification using Biclustering

Version 1.0.3

Description This package is a implementation of biclustering ensemble method MoSBi (Molecular signature Identification from Biclustering). MoSBi provides standardized interfaces for biclustering results and can combine their
results with a multi-algorithm ensemble approach to compute robust ensemble biclusters on molecular omics data.
This is done by computing similarity networks of biclusters and filtering for overlaps using a custom error model.
After that, the louvain modularity it used to extract bicluster communities from the similarity network, which can then be converted to ensemble biclusters.
Additionally, MoSBi includes several network visualization methods to give an intuitive and scalable overview of the results.
MoSBi comes with several biclustering algorithms, but can be easily extended to new biclustering algorithms.

License AGPL-3 + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Depends R (>= 4.1)

SystemRequirements C++17, GNU make

LinkingTo Rcpp, BH, RcppParallel

Imports Rcpp, BH, xml2, methods, igraph, fabia, RcppParallel, biclust, isa2, QUBIC, akmbiclust, RColorBrewer

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

Collate 'RcppExports.R' 'bicluster.R' 'bicluster_net_methods.R' 'ensemble_bicluster.R' 'extract_BicARE.R' 'extract_akmbiclust.R' 'extract_biclust.R' 'extract_biclustpy.R' 'extract_fabia.R' 'extract_isa.R' 'feature_louvain_overlap.R' 'filter_biclusters.R' 'get_biclusters.R' 'misc.R' 'mosbi-package.R' 'pipeline.R' 'run_algorithms.R'

VignetteBuilder knitr

biocViews Software, StatisticalMethod, Clustering, Network

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/mosbi

git_branch RELEASE_3_14

git_last_commit fa15c49

git_last_commit_date 2022-02-01

Date/Publication 2022-04-12

Author Tim Daniel Rose [cre, aut],
Josch Konstantin Pauling [aut]

Maintainer Tim Daniel Rose <tim.rose@wzw.tum.de>

R topics documented:

`algohistogram` *Get list the list of algorithms from a list of bicluster objects.*

Description

Can be used for .g. histograms.

Usage

```
algohistogram(bic)
```

Arguments

`bic`	A list of bicluster objects.

Value

A character vector with the extracted biclustering algorithms used for each bicluster of the input list.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# algohistogram(bics)
```

apply_threshold *Apply a threshold to a bicluster similarity adjacency matrix or a co-occurrence adjacency matrix.*

Description

All values lower than the threshold will be replaced by a 0.

Usage

```
apply_threshold(bic_net)
```

Arguments

bic_net	An object of class bicluster_net or cooccurrence_net.

Value

An adjacency matrix with the applied threshold.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# apply_threshold(bn)
```

apply_threshold,bicluster_net-method *Apply a threshold to a bicluster similarity adjacency matrix.*

Description

All values lower than the threshold will be replaced by a 0.

Usage

```
## S4 method for signature 'bicluster_net'
apply_threshold(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

An adjacency matrix with the applied threshold.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# apply_threshold(bn)
```

---

apply_threshold,cooccurrence_net-method

Apply a threshold to a co-occurrence adjacency matrix.

Description

All values lower than the threshold will be replaced by a 0.

Usage

```
## S4 method for signature 'cooccurrence_net'
apply_threshold(bic_net)
```

Arguments

bic_net	An object of class cooccurrence_net.

Value

An adjacency matrix with the applied threshold.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# apply_threshold(fn)
```

attributeConnector *Extract the class-wise degree of an adjacency matrix.*

Description

For a adjacency matrix as computed by `full_graph`, the function computes how many row-column interactions connect rows (columns) to columns (rows) of a specific class/category.

Usage

```
attributeConnector(mat, otherclasses, useOther = FALSE)
```

Arguments

`mat`	A adjacency matrix with bipartite interactions as computed by `full_graph` or `attribute_graph` (with parameter `bipartite=TRUE`).
`otherclasses`	A logical vector indicating two classes of elements in rows (columns).
`useOther`	Logical indicating if the attributes, that are classified appear first in the matrix (`True`) or the attributes that connect classified attributes (`False`).

Value

A DataFrame that holds the total degree of every attribute (row/column) and the fraction of the degree that connects only to elements of class `True` (from parameter `otherclasses`).

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# attributeConnector(apply_threshold(fn),
#     otherclasses=c(rep(FALSE, 100), rep(TRUE, 100)))
```

attribute_graph *Generate attribute specific co-occurance networks.*

Description

The function generates co-occurance networks for all the attributes. E.g. if `MARGIN="column"`, for each column, a oc-occurance matrix of rows is generated, which includes all biclusters, where the column element is present.

Usage

```
attribute_graph(bics, m, MARGIN = "column")
```

Arguments

bics	A list of bicluster.
m	The matrix used for biclustering.
MARGIN	"row" or "row", Indicating if a list of row- or column-specific networks is generated

Value

A list of numeric matrices. If MARGIN="column" ("row"), the list has a length of ncol(m) (nrow(m)) and each matrix the dimensions of c(nrow(m),nrow(m)) (c(ncol(m),ncol(m)))

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# attribute_graph(bics, m)
```

---

attr_overlap *Count how often row/column elements occur in biclusters.*

Description

Given a list of bicluster objects (bicluster), the function counts the occurance of all elements in the biclusters.

Usage

```
attr_overlap(bics, named)
```

Arguments

bics	A list of bicluster objects.
named	Locigal, indicating, if all bicluster objects have names.

Value

A Data Frame with the counts oof all elements.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# attr_overlap(bics, named=FALSE)
```

---

bicluster-class *A S4 class to store biclusters.*

Description

A S4 class to store biclusters.

Slots

`row` A vector of row.

`column` A vector of columns.

`rowname` A vector of names for the rows in `row`.

`colname` A vector of names for the columns in `column`.

`algorithm` Algorithm that predicted this bicluster.

Examples

```
bicluster(row=c(1,2), column=c(1,2),
    rowname=c("a", "b"), colname=c("e", "f"))
```

---

bicluster_heatmap *Plot a heatmap of a bicluster*

Description

Uses the `stats::heatmap` function.

Usage

```
bicluster_heatmap(bic, m, ...)
```

Arguments

`bic`	A bicluster object.
`m`	The matrix, that was used for the biclustering. (Works only if matrix has row-/colnames.)
`...`	Arguments forwarded to `stats::heatmap`.

Value

A plot object

Examples

```
m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
rownames(m) <- c("c1", "c2")
bicluster_heatmap(bicluster(row=c(1,2), column=c(1,2)), m)
```

`bicluster_heatmap,bicluster,matrix-method`

Plot a heatmap of a bicluster

Description

Uses the `stats::heatmap` function.

Usage

```
## S4 method for signature 'bicluster,matrix'
bicluster_heatmap(bic, m, ...)
```

Arguments

`bic`	A bicluster object.
`m`	The matrix, that was used for the biclustering. (Works only if matrix has row-/colnames.)
`...`	Arguments forwarded to `stats::heatmap`.

Value

A plot object

Examples

```
m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
rownames(m) <- c("c1", "c2")
bicluster_heatmap(bicluster(row=c(1,2), column=c(1,2)), m)
```

bicluster_net-class *A S4 class to store bicluster networks.*

Description

Object that is returned e.g. be the function `bicluster_network`.

Slots

`adjacency_matrix` Adjacency matrix of bicluster similarities.

`threshold` Estimated threshold for the bicluster similarity adjacency matrix. All values lower than that in the matrix should be discarded. (Note that the indicated threshold is not applied to the `adjacency_matrix`)

`algorithms` List of algorithms that contributed to this bicluster network.

Examples

```
bicluster_net(adjacency_matrix=matrix(seq(1:16), nrow=4),
    threshold=4)
```

bicluster_network *Generate a bicluster network*

Description

The function computes a bicluster network based on a selected similarity metric. A similarity cut-off is calculated using randomized biclusters (the bicluster size distribution is kept).

Usage

```
bicluster_network(
  bics,
  mat,
  n_randomizations = 5,
  MARGIN = "both",
  metric = 4,
  n_steps = 100,
  plot_edge_dist = TRUE,
  sn_ratio = TRUE,
  error_threshold = 0.05,
  return_plot_data = FALSE,
  prob_scale = FALSE,
  prl = FALSE
)
```

Arguments

bics	A list of bicluster objects.
mat	The matrix used for biclustering.
n_randomizations	The number of randomizations for cut-off estimation. (The mean of all randomizations is used).
MARGIN	Margin over which the similarity is computed. Can be "row", "column", "mean" (In this case the mean of row and column similarity is used) or "both" (In this case the similarity between all the datapoints of biclusters is used).
metric	The similarity metric same as in similarity_matrix.
n_steps	Number of points where the difference between randomizations and the real data is evaluated.
plot_edge_dist	Show the plots for cut-off estimation with the error model.
sn_ratio	If TRUE, the signal to noise ratio is computed, otherwise the error_threshold is used to to estimate the cut-off at which only error_threshold*100 percent of the edges are estimated to be random overlaps.
error_threshold	If sn_ratio==FALSE this threshold is used to estimate the threshold at which only error_threshold*100 percent of the edges are estimated to be random overlaps.
return_plot_data	Please do not use outside of the package.
prob_scale	Scale similarity by the probability of an overlap equal of higher to the observed one. The scaling is done by multiplying the similarity with (1 -(1 / (1 -log(overlap_probability,base=100)))). The probability is computed using the function p_overlap_2d_higher for MARGIN =="both" and p_overlap_higher otherwise. Can be helpful for big imbalances of bicluster sizes.
prl	Compute the similarity matrix using multiple cores (works only for MARGIN="both"). The number of core can be defined by executing: RcppParallel::setThreadOptions(numThreads = 4) before running this function.

Value

An object of class bicluster_net.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bicluster_network(bics, m)
```

bicluster_net_to_igraph

Convert Bicluster network to an igraph graph object

Description

The function converts a bicluster_net object into an igraph graph object. The threshold is used as a cutoff for the edges of the network.

Usage

```
bicluster_net_to_igraph(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

An igraph::graph object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# bicluster_net_to_igraph(bn)
```

bicluster_net_to_igraph,bicluster_net-method

Convert Bicluster network to an igraph graph object

Description

The function converts a bicluster_net object into an igraph graph object. The threshold is used as a cutoff for the edges of the network.

Usage

```
## S4 method for signature 'bicluster_net'
bicluster_net_to_igraph(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

An igraph::graph object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# bicluster_net_to_igraph(bn)
```

---

bicluster_to_matrix *Convert a bicluster object to an acutal submatrix of the original matrix.*

Description

Convert a bicluster object to an acutal submatrix of the original matrix.

Usage

```
bicluster_to_matrix(m, bic)
```

Arguments

m	Matrix on which the bicluster was computed
bic	Bicluster object

Value

A matrix.

Examples

```
bicluster_to_matrix(matrix(seq(1:16), nrow=4),
    bicluster(row=c(1,2), column=c(1,2)))
```

bicluster_to_matrix,matrix,bicluster-method

Convert a bicluster object to an acutal submatrix of the original matrix.

Description

Convert a bicluster object to an acutal submatrix of the original matrix.

Usage

```
## S4 method for signature 'matrix,bicluster'
bicluster_to_matrix(m, bic)
```

Arguments

m	Matrix on which the bicluster was computed
bic	Bicluster object

Value

A matrix.

#' @examples bicluster_to_matrix(matrix(seq(1:16), nrow=4), bicluster(row=c(1,2), column=c(1,2)))

check_names

Throw an error, if a matrix has not both row- and colnames.

Description

Throw an error, if a matrix has not both row- and colnames.

Usage

```
check_names(m)
```

Arguments

m	A matrix.

Value

Throws error, if matrix has no row- and column names.

Examples

```
m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
colnames(m) <- c("c1", "c2")
check_names(m)
```

`clean_bicluster_list` *Clean a list of biclusters, by returning only the valid ones,*

Description

Clean a list of biclusters, by returning only the valid ones,

Usage

```
clean_bicluster_list(bics)
```

Arguments

`bics`	A list of bicluster objects.

Value

A lis tof bicluster objects

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
clean_bicluster_list(b)
```

`colhistogram` *Get the columnlengths for a list of bicluster objects.*

Description

Can be used for e.g. histograms.

Usage

```
colhistogram(bic)
```

Arguments

`bic`	A list of bicluster objects.

Value

A vector with the lenghts of the columns in every bicluster object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# colhistogram(bics)
```

---

cooccurrence_net-class

A S4 class to store co-occurence networks.

Description

Object that is returned e.g. be the function `feature_network`.

Slots

`adjacency_matrix` Adjacency matrix of row- and column-element co-occurences.

`threshold` Estimated threshold for the co-occurence adjacency matrix. All values lower than that in the matrix should be discarded. (Note that the indicated threshold is not applied to the `adjacency_matrix`)

Examples

```
cooccurrence_net(adjacency_matrix=matrix(seq(1:16), nrow=4),
    threshold=4)
```

---

cooccurrence_net_to_igraph

Convert a co-occurence network to an igraph graph object

Description

The function converts a `cooccurrence_net` object into an igraph graph object. The `threshold` is used as a cutoff for the edges of the network.

Usage

```
cooccurrence_net_to_igraph(occ_net)
```

Arguments

occ_net	An object of class cooccurrence_net.

Value

An igraph::graph object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# cooccurrence_net_to_igraph(fn)
```

---

cooccurrence_net_to_igraph,cooccurrence_net-method

Convert a co-occurrence to an igraph graph object

Description

The function converts a cooccurrence_net object into an igraph graph object. The threshold is used as a cutoff for the edges of the network.

Usage

```
## S4 method for signature 'cooccurrence_net'
cooccurrence_net_to_igraph(occ_net)
```

Arguments

occ_net	An object of class cooccurrence_net.

Value

An igraph::graph object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# cooccurrence_net_to_igraph(fn)
```

cpp_matrix_subsetting *Subsetting of R matrices within c++.*

Description

Subsetting of R matrices within c++.

Usage

```
cpp_matrix_subsetting(m, bic)
```

Arguments

m	A numeric matrix
bic	A bicluster object.

Value

Matrix subset.

Examples

```
cpp_matrix_subsetting(matrix(seq(1:16), nrow=4),
    bicluster(row=c(1,2), column=c(1,2)))
```

detect_elements *Detect the number of elements in a list of biclusters.*

Description

Finds the highest element in a list of bicluster objects.

Usage

```
detect_elements(bics, MARGIN = "row")
```

Arguments

bics	A list of bicluster objects.
MARGIN	Choose if the distance is computed over "row" or "column".

Value

Return highest row or column index from a list of biclusters.

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
detect_elements(b)
```

`dim,bicluster-method` *Get the dimensions of a bicluster.*

Description

Get the dimensions of a bicluster.

Usage

```
## S4 method for signature 'bicluster'
dim(x)
```

Arguments

`x`	A bicluster object.

Value

A numeric vector with the lengths of the rows and columns of the bicluster.

Examples

```
dim(bicluster(row=c(1,2), column=c(1,2)))
```

`distance_matrix` *Compute distances between biclusters*

Description

This function computes a distance matrix between biclusters using different dissimilarity metrics.

Usage

```
distance_matrix(bics, MARGIN = "row", metric = 1L)
```

Arguments

`bics`	A list of bicluster objects.
`MARGIN`	Choose if the distance is computed over `"row"` or `"column"`.
`metric`	Integer indicating which metric is used. 1: Bray-Curtis dissimilarity (default), 2: Jaccard distance, 3: 1-overlap coefficient 4: 1 - Fowlkes–Mallows index.

Value

A numeric matrix of the dissimilarities between all given biclusters.

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
distance_matrix(b)
```

---

ensemble_biclusters *Convert communities into ensemble biclusters*

Description

After calculation of communities with the get_louvain_communities function, the result can be converted into a list of bicluster objects with this function. Only biclusters are returned which have a minimum dimension of 2x2.

Usage

```
ensemble_biclusters(
  coms,
  bics,
  mat,
  row_threshold = 0.1,
  col_threshold = 0.1,
  threshold_sorted = FALSE
)
```

Arguments

coms	A list of communities (bicluster_nets) as outputted by get_louvain_communities.
bics	The list biclusters that was used for calculation with bicluster_network.
mat	The numeric matrix, that was used for biclustering.
row_threshold	Minimum fraction of biclusters of a community in which a row needs to occur so that it will be part of the outputted ensemble bicluster.
col_threshold	Minimum fraction of biclusters of a community in which a column needs to occur so that it will be part of the outputted ensemble bicluster.
threshold_sorted	Return the rows and columns in sorted by decreasing fraction.

Value

A list of bicluster objects.

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
    bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
# m <- matrix(runif(100), nrow=10)
# tm = matrix(c(0,1,1,0), nrow=2)
# bn <- list(bicluster_net(adjacency_matrix=tm, threshold=.5))
# ensemble_biclusters(bn, b, m)
```

`feature_louvain_overlap` *Overlap of features/samples in different louvain communities*

Description

The function calculates how often features or samples occur across all calculated louvain communities

Usage

```
feature_louvain_overlap(overlap_tables, mat)
```

Arguments

`overlap_tables`	List of tables as returned by `attr_overlap`.
`mat`	The data matrix used as input for the biclustering algorithms.

Value

List of tables as returned by `attr_overlap`, extended by a column showing how often elements occur across all tables.

Examples

```
# a = data.frame(type=c("row", "row", "row", "column", "column", "column"),
#   ID=c(1,2,3,1,2,3), Fraction=c(1,1,1,.5, .5, .5))
# b = data.frame(type=c("row", "row", "row", "column", "column", "column"),
#   ID=c(3,2,4,1,5,3), Fraction=c(1,1,1,.5, .5, .5))
# inl <- list(a, b)
# feature_louvain_overlap(outl, matrix(1:100, nrow=10))
```

feature_network *Generate a co-occurence network*

Description

The function computes a co-occurence network, based on the function full_graph. A similarity threshold is calculated using randomized biclusters (the bicluster size distribution is kept).

Usage

```
feature_network(
  bics,
  mat,
  n_randomizations = 5,
  n_steps = 100,
  plot_edge_dist = TRUE,
  sn_ratio = 1,
  error_threshold = 0.05,
  return_plot_data = FALSE,
  rr = 1,
  rc = 1,
  cc = 1,
  w = 0
)
```

Arguments

bics	A list of bicluster objects.
mat	The matrix used for biclustering.
n_randomizations	The number of randomizations for cut-off estimation. (The mean of all randomizations is used).
n_steps	Number of points where the difference between randomizations and the real data is evaluated.
plot_edge_dist	Show the plots for threshold estimation.
sn_ratio	If TRUE, the signal to noise ratio is computed, otherwise the error_threshold is used to to estimate the threshold at which only error_threshold*100 percent of the edges are estimated to be random overlaps.
error_threshold	If sn_ratio==FALSE this cut-off is used to estimate the cut-off at which only error_threshold*100 percent of the edges are estimated to be random overlaps.
return_plot_data	Please do not use outside of the package.
rr	See full_graph.

rc	See full_graph.
cc	See full_graph.
w	See parameter weighting of full_graph.

Value

An object of class cooccurrence_net.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# feature_network(bics, m)
```

---

filter_biclusters *Filter biclusters based on a user defined filter function.*

Description

If the function returns True, the bicluster is added to the output list of biclusters. Every bicluster is validated, before forwarding to the filter-function.

Usage

```
filter_biclusters(bics, mat, filterfun, ...)
```

Arguments

bics	A list of valid bicluster objects.
mat	Original matrix, that was used for biclustering.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A filtered list ob bicluster objects with length(returned_list)<=length(bics).

Examples

```
# m <- matrix(runif(100), nrow=10)
b <- list(bicluster(row=c(3,4), column=c(3,4)),
    bicluster(row=c(3,4,5,6), column=c(3,4,5,6)),
    bicluster(row=c(3,4,5,6), column=c(3,6)))
# filter_biclusters(b, m, function(x) sum(x) < 0)
```

filter_bicluster_size *Filter a list of bicluster objects, by erasing all biclusters, that do not fulfill the minimum number of rows and columns. Utilizes the function* validate_bicluster.

Description

Filter a list of bicluster objects, by erasing all biclusters, that do not fulfill the minimum number of rows and columns. Utilizes the function validate_bicluster.

Usage

```
filter_bicluster_size(bics, minRow, minCol)
```

Arguments

bics	List of bicluster objects.
minRow	Minimum number of rows.
minCol	Minimum number of columns.

Value

A filtered list of bicluster objects.

Examples

```
b <- list(bicluster(row=c(1,2), column=c(1,2,3,4)),
       bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
filter_bicluster_size(b, 3, 3)
```

filter_matrix *Filter a matrix*

Description

All values below the threshold will be replaces by 0.

Usage

```
filter_matrix(mat, threshold = 1)
```

Arguments

mat	A Numeric matrix.
threshold	All values below will be replaces by 0.

Value

A filtered numeric matrix.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# filter_matrix(m, threshold=1)
```

`filter_subsets` *Remove all biclusters from a list, that are identical or perfect subsets from each other. Additionally all invalid biclusters are removed (See* `validate_bicluster`*).*

Description

Remove all biclusters from a list, that are identical or perfect subsets from each other. Additionally all invalid biclusters are removed (See `validate_bicluster`).

Usage

```
filter_subsets(bics)
```

Arguments

`bics`	A list of bicluster objects

Value

A list of bicluster objects, where perfects subsets or identical biclusters are deleted.

Examples

```
filter_subsets(list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
    bicluster(row=c(1,2,3,4), column=c(1,2,3,4))))
```

full_graph *Generate a similarity network for a list of biclusters*

Description

The function computes a adjacency matrix for rows and columns of biclusters. The matrix values show, how often two rows or two columns or a row and a column occur together in biclusters. In the resulting adjacency matrix, rows are listed first, followed by columns. They have the same order as the the rows and columns of the input matrix.

Usage

```
full_graph(
  bics,
  m,
  rr_weight = 1L,
  rc_weight = 1L,
  cc_weight = 1L,
  weighting = 0L
)
```

Arguments

bics	A list of biclusters.
m	The matrix, that was used to calculated the biclusters.
rr_weight	Weight row-row interactions.
rc_weight	Weight row-col interactions.
cc_weight	Weight col-col interactions.
weighting	Weight interactions by bicluster size. 0 - no weighting, 1 - multiply by bicluster size, 2 - divide by bicluster size.

Details

In case the given biclusters have overall more or less columns than rows, the interactions can be weighted to visualize the result properly.

Value

An adjacency matrix.

Examples

```
m <- matrix(seq(1:16), nrow=4)
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
# full_graph(b, m)
```

getAkmbiclustClusters *Extract a list of bicluster objects from an akmbiclust biclustering object.*

Description

Extract a list of bicluster objects from an akmbiclust biclustering object.

Usage

```
getAkmbiclustClusters(bics, mat, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

bics	A result object from akmbiclust.
mat	Original matrix, that was used for biclustering.
transposed	True, if the bicluster calculation was performed on a tranposed matrix.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which have to be valid (See validate_bicluster.

Examples

```
# Function called in
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# Not run: run_akmbiclust(m, k=10)
```

getallBFClusters *Get all biclusters from a Bi-Force output file.*

Description

Get all biclusters from a Bi-Force output file.

Usage

```
getallBFClusters(filename)
```

Arguments

filename	Name of the Bi-Force output file.

Value

List of biclusters in the form of getBFCluster

Examples

```
a <- "PathToBiForceOutput.txt"
# getallBFClusters(a)
```

---

getBFCluster *Get a bicluster a Bi-Force output file*

Description

Get a bicluster a Bi-Force output file

Usage

```
getBFCluster(filename, cluster)
```

Arguments

filename	Name of the Bi-Force output file.
cluster	Number of the bicluster that should be extracted.

Value

Bicluster as list with rownames in attribute "row" and colnames in attribute "column".

Examples

```
a <- "PathToBiForceOutput.txt"
# getBFCluster(a, cluster=1)
```

getBicAREbiclusters *Extract a list of bicluster objects from an BicARE biclustering object.*

Description

Extract a list of bicluster objects from an BicARE biclustering object.

Usage

```
getBicAREbiclusters(bics, mat, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

bics	A BicARE bicluster object.
mat	Original matrix, that was used for biclustering.
transposed	True, if the bicluster calculation was performed on a tranposed matrix.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which have to be valid (See validate_bicluster.

Examples

```
# Note that BicARE packackage is not included in the mosbi package
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# res <- BicARE::FLOC(m)
# getBicAREbiclusters(res, m)
```

getBiclustClusters *Extract a list of bicluster objects from a biclust object.*

Description

Extract a list of bicluster objects from a biclust object.

Usage

```
getBiclustClusters(
  bics,
  mat,
  method = "biclust",
  transposed = FALSE,
  filterfun = NULL,
  ...
)
```

Arguments

bics	A biclust object.
mat	Original matrix, that was used for biclustering.
method	Name of the used biclustering algorithm. Should be one of the following: "biclust", "biclust-bimax", "biclust-cc", "biclust-plaid", "biclust-quest", "biclust-spectral", "biclust-xmotifs" or "biclust-qubic", "biclust-unibic".
transposed	True, if the bicluster calculation was performed on a tranposed matrix.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which have to be valid (See validate_bicluster.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# res <- biclust::biclust(m, method = biclust::BCBimax())
# getBiclustClusters(res, m)
```

---

getBiclustpyClusters *Extract a list of bicluster objects from an biclustpy output file.*

Description

Extract a list of bicluster objects from an biclustpy output file.

Usage

```
getBiclustpyClusters(bics, mat, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

bics	A biclust object.
mat	Original matrix, that was used for biclustering.
transposed	True, if the bicluster calculation was performed on a tranposed matrix.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which have to be valid (See validate_bicluster.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# Not run: getBiclustpyClusters("PathToFileOfBiclustpyResults", m)
```

---

getFabiaClusters *Extract a list of bicluster objects from an fabia biclustering object.*

Description

Extract a list of bicluster objects from an fabia biclustering object.

Usage

```
getFabiaClusters(bics, mat, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

bics	Extracted fabia biclusters.
mat	Original matrix, that was used for biclustering.
transposed	True, if the bicluster calculation was performed on a tranposed matrix.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which have to be valid (See validate_bicluster.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# res <- fabia::extractBic(fabia::fabia(m, p=5))
# getFabiaClusters(res, m)
```

`getIsaClusters` *Extract a list of bicluster objects from an isa2 biclustering object.*

Description

Extract a list of bicluster objects from an isa2 biclustering object.

Usage

```
getIsaClusters(bics, mat, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

`bics`	A biclust object.
`mat`	Original matrix, that was used for biclustering.
`transposed`	`True`, if the bicluster calculation was performed on a tranposed matrix.
`filterfun`	A function to filter biclusters. Only if the function returns `True`, the bicluster is added to the returned list. The function has to accept a the bicluster (given as submatrix of `mat`) `filterfun(bicluster_matrix,...)`.
`...`	Other parameters forwarded to the `filterfun`.

Value

A list of `bicluster` objects, which have to be valid (See `validate_bicluster`.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# Function part of:
# m <- matrix(rnorm(10000), nrow=100)
# Not run: run_isa(m)
```

getQUBIC2biclusters *Extract QUBIC2 biclusters*

Description

Extract biclusters from a QUBIC2 "*.blocks" file. Row and column names are not added to the bicluster objects.

Usage

```
getQUBIC2biclusters(filename, transposed = FALSE)
```

Arguments

filename	Path to the QUBIC2 results file.
transposed	Set to TRUE, if the biclustering was performed on a tranposed matrix.

Value

A list of validated bicluster objects (See validate_bicluster).

Examples

```
a <- "PathToQUBIC2output.txt"
# Not run: getQUBIC2biclusters(a)
```

get_adjacency *Get Adjacency matrix*

Description

Return Adjacency matrix from bicluster network

Usage

```
get_adjacency(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

Raw unfiltered adjacency matrix.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_adjacency(bn)
```

---

get_adjacency,bicluster_net-method

Get Adjacency matrix

Description

Return Adjacency matrix from bicluster network

Usage

```
## S4 method for signature 'bicluster_net'
get_adjacency(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

Raw unfiltered adjacency matrix.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_adjacency(bn)
```

get_algorithms *Get Algorithms*

Description

Get a unique vector of algorithms from a list of bicluster objects.

Usage

```
get_algorithms(bics)
```

Arguments

bics	a list of bicluster objects.

Value

A character vector with algorithm names

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4), algorithm="isa"),
    bicluster(row=c(3,4,5,6), column=c(3,4,5,6), algorithm="QUBIC"))
```

get_biclusters *Extract biclusters from different algorithms/packages*

Description

Converts biclusters output of different algorithms/packages in to lists of bicluster objects. Many algoritms can be directly executed using the run_... methods from this package. This directly returns the converted results. Not all algorithms are shipped with this package, like Bi-Force, which is running in Java as a standalone tool or BicARE, which required an full import using library(BicARE) in order to run. But their results can be converted using this function.

Usage

```
get_biclusters(bics, mat, method, transposed = FALSE, filterfun = NULL, ...)
```

Arguments

bics	A resulting object from a biclustering algorithm (extracted biclusters for fabia) or filename for stored biclustering results.
mat	Original matrix, that was used for biclustering.
method	Used biclustering package. One of "biclust" (can be further specified as "biclust-bimax", "biclust-cc", "biclust-plaid", "biclust-quest", "biclust-qubic", "biclust-spectral", "biclust-xmotifs", "biclust-unibic"), "BicARE", "isa", "fabia", "biforce", "biclustpy", "qubic2" or "akmbiclust".
transposed	Indicate, whether a transposed version of the matrix is used for biclustering. The matrix should not be transposed, when this argument is set to True.
filterfun	A function to filter biclusters. Only if the function returns True, the bicluster is added to the returned list. The function has to accept a bicluster (given as submatrix of mat) filterfun(bicluster_matrix,...).
...	Other parameters forwarded to the filterfun.

Value

A list of bicluster objects, which are valid (See validate_bicluster).

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# res <- isa2::isa(m)
# get_biclusters(res, m, "isa")
```

---

get_bic_net_algorithms

Get bicluster network algorithms

Description

Return algorithms from bicluster network

Usage

```
get_bic_net_algorithms(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

Algorithm names as characters.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_bic_net_algorithms(bn)
```

---

get_bic_net_algorithms,bicluster_net-method

Get bicluster network algorithms

Description

Return algorithms from bicluster network

Usage

```
## S4 method for signature 'bicluster_net'
get_bic_net_algorithms(bic_net)
```

Arguments

bic_net	An object of class bicluster_net.

Value

Algorithm names as characters.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_bic_net_algorithms(bn)
```

get_louvain_communities

Get louvain communities from a bicluster network

Description

Extracts the louvain communities from a bicluster_net or cooccurrence_net object using the louvain modularity optimization from the igraph package (cluster_louvain).

Usage

```
get_louvain_communities(bic_net, min_size = 2, bics = NULL)
```

Arguments

bic_net	A bicluster_net or cooccurrence_netobject.
min_size	Minimum size of a louvain community to be returned (minimum value is 2).
bics	Optional. Is only use for the class bicluster_net. The respective list of biclusters to identify, from which algorithms a community originates.

Value

A list of bicluster_net or cooccurrence_net objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_louvain_communities(bn)
```

get_louvain_communities,bicluster_net-method

Get louvain communities from a bicluster network

Description

Extracts the louvain communities from a bicluster_net object using the louvain modularity optimization from the igraph package (cluster_louvain).

Usage

```
## S4 method for signature 'bicluster_net'
get_louvain_communities(bic_net, min_size = 2, bics = NULL)
```

Arguments

bic_net	A bicluster_net object.
min_size	Minimum size of a louvain community to be returned.
bics	Optional. The respective list of biclusters to identify, from which algorithms a community originates.

Value

A list of bicluster_net objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# get_louvain_communities(bn)
```

---

get_louvain_communities,cooccurrence_net-method

Get louvain communities from a co-occurrence network

Description

Extracts the louvain communities from a cooccurrence_net object using the louvain modularity optimization from the igraph package (cluster_louvain).

Usage

```
## S4 method for signature 'cooccurrence_net'
get_louvain_communities(bic_net, min_size = 2, bics = NULL)
```

Arguments

bic_net	A cooccurrence_net object.
min_size	Minimum size of a louvain community to be returned (minimum value is 2).
bics	Not used.

Value

A list of cooccurrence_net objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# get_louvain_communities(fn)
```

---

has_names *Check, whether a matrix has row- and colnames.*

Description

Check, whether a matrix has row- and colnames.

Usage

```
has_names(m)
```

Arguments

m	A matrix

Value

Logical indicating existence of row- and colnames.

Examples

```
has_names(matrix(c(1,2,3,4), nrow=2))

m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
rownames(m) <- c("c1", "c2")
has_names(m)
```

is_subset_or_identical

Check if a bicluster is a subset (in rows AND columns) of identical to another bicluster.

Description

Check if a bicluster is a subset (in rows AND columns) of identical to another bicluster.

Usage

```
is_subset_or_identical(bic1, bic2)
```

Arguments

bic1	A bicluster.
bic2	A bicluster.

Value

1 if bic1 is a subset of bic2, 2 if bic 1 is identical to bic2, 0 else.

Examples

```
is_subset_or_identical(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
    bicluster(row=c(1,2,3,4), column=c(1,2,3,4)))
```

network_edge_strength

Count edges in an adjacency matrix using different cut-off thresholds.

Description

Computes the how many edges remain in a network if edges with a weight lower than a certain threshold are removed. The number of remaining edges between 1 and max(adjm) are calculated. It is assumend that the matrix is symmetric and therefore the number of edges divided by two. Uses the function replace_values.

Usage

```
network_edge_strength(adjm)
```

Arguments

adjm	A symmetrix numeric matrix.

Value

A numeirc matrix of `dim(max(adjm),2)`. The first column indicated the applied threshold, the second column the remaining edges.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# network_edge_strength(apply_threshold(fn))
```

`network_edge_strength_float`

Count edges in an adjacency matrix using different cut-off thresholds.

Description

Same as `network_edge_strength`, but for (positive) non-integer matrices.

Usage

```
network_edge_strength_float(adjm, steps = 100L, maximum = 0)
```

Arguments

`adjm`	A symmetrix numeric matrix.
`steps`	Number of steps for which the edge count is evaluated.
`maximum`	Highest value until which the edge weight is evaluated. If maximum=0, the max value of `adjm` is used.

Details

Computes the how many edges remain in a network if edges with a weight lower than a certain threshold are removed. The number of remaining edges between 1 and max(adjm) are calculated. It is assumend that the matrix is symmetric and therefore the number of edges divided by two. Uses the function `replace_values_float`.

Value

A numeirc matrix of `dim(max(adjm),2)`. The first column indicated the applied threshold, the second column the remaining edges.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# network_edge_strength_float(apply_threshold(bn))
```

NoBFBiclusters *Get the number of biclusters, generated by the Bi-Force algorithm.*

Description

Get the number of biclusters, generated by the Bi-Force algorithm.

Usage

```
NoBFBiclusters(filename)
```

Arguments

filename	Name of the Bi-Force output file.

Value

Number of biclusters.

Examples

```
a <- "PathToBiForceOutput.txt"
# NoBFBiclusters(a)
```

---

node_size *Node sizes for plotting bicluster networks.*

Description

When plotting bicluster networks, node sizes adapted to bicluster sizes can improve visual inspection. Node sizes are computed using the following formula: `(atan( (x -min(x)) / (max(x) -min(x)) + offset ) * base_size)`. With x being defined a vector of bicluster sizes defined by the `MARGIN` parameter.

Usage

```
node_size(bics, base_size = 10, offset = 0.2, MARGIN = "column")
```

Arguments

bics	A list of bicluster objects.
base_size	Is multiplied with the atan result for the node size
offset	Offset for the atan calculation. Has to be > 0. Smaller values result in higher differences of node sizes.
MARGIN	"column", "row" or "both" are taken into account for the size of a bicluster bicluster

Value

Vector of node sizes as floats.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# nz <- node_size(bics)
# plot_algo_network(bn, bics, vertex.size=nz)
# plot(bn, vertex.size=node_size(bics, offset=.1, base_size=15))
```

---

occurance_matrix *Occurance matrix of data points in a list of biclusters*

Description

The function computes a matrix with the same dimensions as the input matrix and fills the matrix elements with the frequence of occurance of the data points in the input list of biclusters.

Usage

```
occurance_matrix(bics, mat)
```

Arguments

bics	A list of bicluster objects.
mat	The data matrix used for biclustering.

Value

A numeric matrix with the dimensions of the input matrix. The values represent the frequency of occurance of this point in the list of biclusters.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# occurance_matrix(bics, m)
```

---

occurance_table *Occurance table of data points in a list of biclusters*

Description

The function uses the occurance_matrix function and returns all values higher than the threshold as a DataFrame.

Usage

```
occurance_table(bics, mat, threshold = 0)
```

Arguments

bics	A list of bicluster objects.
mat	The data matrix used for biclustering.
threshold	Only data points higher than this threshold are returned.

Value

A DataFrame with the frequencies of occurance for values higher than a threshold.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# occurance_table(bics, m, threshold=.1)
```

plot,bicluster_net,missing-method
Plot a bicluster network

Description

Converts the object into a graph and uses its plot function.

Usage

```
## S4 method for signature 'bicluster_net,missing'
plot(x, y, ...)
```

Arguments

x	An object of class bicluster_net.
y	Not used.
...	Plot parameters forwarded to igraph::plot.igraph

Value

An graph plot.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# plot(bn)
```

plot,cooccurrence_net,missing-method
Plot a co-occurrence network

Description

Converts the object into a graph and uses its plot function.

Usage

```
## S4 method for signature 'cooccurrence_net,missing'
plot(x, y, ...)
```

Arguments

x	An object of class cooccurrence_net.
y	Not used.
...	Plot parameters forwarded to igraph::plot.igraph

Value

An graph plot.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# fn <- feature_network(bics, m)
# plot(fn)
```

---

plot_algo_network *Plot a bicluster network colored by algorithms.*

Description

In the plot each bicluster is colored by the algorithm, that generated it.

Usage

```
plot_algo_network(bic_net, bics, new_layout = TRUE, ...)
```

Arguments

bic_net	A bicluster_net object.
bics	The corresponding list of biclusters from bic_net.
new_layout	If FALSE, the plot accepts a network layout as a parameter, other wise a new layout is computed.
...	Plot parameters forwarded to igraph::plot.igraph After calculating communities with get_louvain_communities it is necessary to get the subset of biclusters using select_biclusters_from_bicluster_network.

Value

If new_layout, a new network layout is returned that can be used for other plots.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# plot_algo_network(bn, bics)
```

`plot_piechart_bicluster_network` *Plot a bicluster network with piecharts as nodes.*

Description

Plot a bicluster network with piecharts as nodes.

Usage

```
plot_piechart_bicluster_network(
  bic_net,
  bics,
  class_vector,
  colors,
  named = TRUE,
  MARGIN = "column",
  new_layout = TRUE,
  ...
)
```

Arguments

`bic_net`	A `bicluster_net` object.
`bics`	The corresponding list of biclusters from `bic_net`. After calculating communities with `get_louvain_communities` it is necessary to get the subset of biclusters using `select_biclusters_from_bicluster_network`.
`class_vector`	A (named) vector with class affinities. Every occuring element in the biclustes must have a non NA value in this list.
`colors`	Colors used for the classes. Must be a vector with colors in the order of sort(unique(class_vector)).
`named`	Indicates if `rowname`/`colname` of the bicluster objects should be used instead of the indizes.
`MARGIN`	Must be "row" or "column". Indicates which dimension of the bicluster should be used for coloring.
`new_layout`	If `FALSE`, the plot accepts a network layout as a parameter, other wise a new layout is computed.
`...`	Additional parameters forwarded to `plot.igraph`.

Value

If new_layout, a new network layout is returned that can be used for other plots.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# groups <- ifelse(runif(100)< 0.5, "group1", "group2")
# cols <- c("group1"="blue", "group2"="grey")
# plot_piechart_bicluster_network(bn, bics, groups, cols, named=FALSE)
```

---

p_overlap *Probability for an overlap of two samples.*

Description

The probability is computed using the forumla $\frac{\binom{y}{x} \times \binom{n-y}{k-x}}{\binom{n}{k}}$.

Usage

```
p_overlap(x, y, k, n)
```

Arguments

x	Overlap.
y	Size of sample 1.
k	Size of Sample 2.
n	Number of all elements sampled from.

Value

Overlap probability.

Examples

```
p_overlap(10, 20, 30, 100)
```

p_overlap_2d *Probability for an overlap of two dimensional samples*

Description

Is computed by calculating the overlap probability for each dimension independently and multiplying them using the function p_overlap.

Usage

```
p_overlap_2d(ov_x, ov_y, s1x, s1y, s2x, s2y, mat_x, mat_y)
```

Arguments

ov_x	Overlap in the first dimension.
ov_y	Overlap in the second dimension.
s1x	First sample of the first dimension.
s1y	First sample of the second dimension.
s2x	Second sample of first dimension.
s2y	Second sample of the second dimension.
mat_x	Number of all elements from the first dimension sampled from.
mat_y	Number of all elements from the second dimension sampled from.

Value

Overlap probability.

Examples

```
p_overlap_2d(10, 10, 20, 20, 30, 30, 100, 100)
```

p_overlap_2d_higher *Probability for an overlap higher or equal to the observed one of two dimensional samples*

Description

Is computed by adding up probabilities for all combinations of the observed or higher overlaps using the function p_overlap_2d.

Usage

```
p_overlap_2d_higher(ov_x, ov_y, s1x, s1y, s2x, s2y, mat_x, mat_y)
```

Arguments

ov_x	Overlap in the first dimension.
ov_y	Overlap in the second dimension.
s1x	First sample of the first dimension.
s1y	First sample of the second dimension.
s2x	Second sample of first dimension.
s2y	Second sample of the second dimension.
mat_x	Number of all elements from the first dimension sampled from.
mat_y	Number of all elements from the second dimension sampled from.

Value

Overlap probability

Examples

```
p_overlap_2d_higher(10, 10, 20, 20, 30, 30, 100, 100)
```

---

p_overlap_higher *Probability for an overlap higher or equal to the observed one of two samples*

Description

Is computed by adding up probabilities for all possible overlaps equal or higher to the observed one using the function p_overlap.

Usage

```
p_overlap_higher(x, y, k, n)
```

Arguments

x	Overlap.
y	Size of sample 1.
k	Size of Sample 2.
n	Number of all elements sampled from.

Value

Overlap probability.

Examples

```
p_overlap_higher(10, 20, 30, 100)
```

randomize_matrix *Randomize a matrix*

Description

Randomize a matrix bu shuffling all rows and columns.

Usage

```
randomize_matrix(m)
```

Arguments

m	A matrix.

Value

A randomized version of the input matrix.

Examples

```
m <- matrix(c(1,2,3,4), nrow=2)
randomize_matrix(m)
```

replace_threshold *Replace elements of an integer matrix.*

Description

This function replaces all elements of an integer matrix, which are under a certain threshold (<) with zero.

Usage

```
replace_threshold(m, threshold)
```

Arguments

m	A numeric matrix.
threshold	A numeric threshold under which all elements in the matrix are replaced by zero.

Value

An integer matrix.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# replace_threshold(m, 1)
```

`replace_values` *Replace values in an integer adjacency matrix.*

Description

Replace values in an integer matrix, that are lower than a certain threshold.

Usage

```
replace_values(mat, threshold, replace_higher = TRUE)
```

Arguments

`mat`	An integer matrix
`threshold`	All values in the matrix lower than this values are replaced by 0.
`replace_higher`	If set to true, all values >= `threshold` are replaced by 1.

Value

An integer matrix with (partially) replaced values.

Examples

```
replace_values(matrix(seq(1, 16), nrow=4), threshold=4)
```

`replace_values_float` *Replace values in a adjacency matrix.*

Description

Same as `replace_values`, but for (positive) non-integer matrices.

Usage

```
replace_values_float(mat, threshold, replace_higher = TRUE)
```

Arguments

`mat`	A numeric matrix
`threshold`	All values in the matrix lower than this values are replaced by 0.
`replace_higher`	If set to true, all values >= `threshold` are replaced by 1.

Details

Replace values in a numeric matrix, that are lower than a certain threshold.

Value

A numeric matrix with (partially) replaced values.

Examples

```
replace_values(matrix(rnorm(100), nrow=10), threshold=1)
```

---

rowhistogram *Get the rowlengths for a list of bicluster objects.*

Description

Can be used for e.g. histograms.

Usage

```
rowhistogram(bic)
```

Arguments

`bic`	A list of bicluster objects.

Value

A vector with the lenghts of the rows in every bicluster object.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# rowhistogram(bics)
```

run_akmbiclust *Run the akmbiclust biclustering algorithm*

Description

The function executes the akmbiclust biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_akmbiclust(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the akmbiclust function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# set.seed(10)
# m <- matrix(rnorm(10000), nrow=100)
# Not run: run_akmbiclust(m, k=10)
```

run_bimax *Run the Bimax biclustering algorithm*

Description

The function executes the BCBimax biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_bimax(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCBimax function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_bimax(m)
```

---

run_cc *Run the CC biclustering algorithm*

Description

The function executes the BCCC biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_cc(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCCC function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_cc(m)
```

`run_fabia` *Run the fabia biclustering algorithm*

Description

The function executes the fabia biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_fabia(
  data_matrix,
  minRow = 2,
  minCol = 2,
  thresZ = 0.5,
  thresL = NULL,
  ...
)
```

Arguments

`data_matrix`	A numeric matrix.
`minRow`	Same parameters as in filter_bicluster_size.
`minCol`	Same parameters as in filter_bicluster_size.
`thresZ`	See parameter from the extractBic function.
`thresL`	See parameter from the extractBic function.
`...`	Other parameters forwarded to the fabia function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(1000), nrow=10)
# run_fabia(m, p=5)
```

run_isa *Run the isa biclustering algorithm*

Description

The function executes the isa biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_isa(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the isa function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# Not run: run_isa(m)
```

run_plaid *Run the Plaid biclustering algorithm*

Description

The function executes the BCPlaid biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_plaid(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCPlaid function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_plaid(m)
```

---

run_qubic *Run the QUBIC biclustering algorithm*

Description

The function executes the BCQU biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_qubic(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCQU function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_qubic(m)
```

run_quest *Run the Quest biclustering algorithm*

Description

The function executes the BCQuest biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_quest(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCQuest function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_quest(m)
```

run_spectral *Run the spectral biclustering algorithm*

Description

The function executes the BCSpectral biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_spectral(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCSpectral function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_spectral(m)
```

---

run_unibic *Run the UniBic biclustering algorithm*

Description

The function executes the runibic::BCUnibic biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_unibic(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the runibic::BCUnibic function. function.

Value

a list of bicluster objects.

Function as a string, which can be executed.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_unibic(m, nbic=10)
```

run_xmotifs *Run the Xmotifs biclustering algorithm*

Description

The function executes the BCXmotifs biclustering algorithm, returning a list of biclusters converted into bicluster objects compatible with this package. If the algorithm fails to run, an empty list is returned.

Usage

```
run_xmotifs(data_matrix, minRow = 2, minCol = 2, ...)
```

Arguments

data_matrix	A numeric matrix.
minRow	Same parameters as in filter_bicluster_size.
minCol	Same parameters as in filter_bicluster_size.
...	Other parameters forwarded to the BCXmotifs function.

Value

a list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# run_xmotifs(m)
```

sample_biclusters *Sample a list of biclusters.*

Description

The function generates a list of biclusters given an input list of biclusters, where each bicluster has the same number or rows and columns, but with sampled entries from a uniform distribution of all rows and columns is the matrix.

Usage

```
sample_biclusters(bics, mat)
```

Arguments

bics	A list of validated bicluster objects.
mat	The numeric matrix, that was used to generate the biclusters.

Value

A list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# sample_biclusters(bics, m)
```

---

select_biclusters_from_bicluster_network

Create a subset of biclusters based on a bicluster network

Description

The function returns an adapted bicluster list based on a bicluster_net object. This might be necessary e.g. after get_louvain_communities was used a community consists only of a subset of the biclusters.

Usage

```
select_biclusters_from_bicluster_network(bic_net, bics)
```

Arguments

bic_net	A bicluster_net.
bics	A list of bicluster objects as returned by get_biclusters.

Value

A subsetted list of bicluster objects

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# lc <- get_louvain_communities(bn)
# select_biclusters_from_bicluster_network(lc[[1]], bics)
```

select_biclusters_from_bicluster_network,bicluster_net,list-method

Create a subset of biclusters based on a bicluster network

Description

The function returns an adapted bicluster list based on a bicluster_net object. This might be necessary e.g. after get_louvain_communities was used and a community consists only of a subset of the biclusters.

Usage

```
## S4 method for signature 'bicluster_net,list'
select_biclusters_from_bicluster_network(bic_net, bics)
```

Arguments

bic_net	A bicluster_net.
bics	A list of bicluster objects as returned by get_biclusters.

Value

A subsetted list of bicluster objects.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# lc <- get_louvain_communities(bn)
# select_biclusters_from_bicluster_network(lc[[1]], bics)
```

set_bicluster_names *Add row-/colnames to a bicluster object.*

Description

Add row-/colnames to a bicluster object.

Usage

```
set_bicluster_names(bic, m)
```

Arguments

bic	A bicluster object.
m	The matrix, that was used for the biclustering. (Works only if matrix has row-/colnames.)

Value

The updated bicluster object.

Examples

```
m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
rownames(m) <- c("c1", "c2")
set_bicluster_names(bicluster(row=c(1,2), column=c(1,2)), m)
```

---

set_bicluster_names,bicluster,matrix-method

Add row-/colnames to a bicluster object.

Description

Add row-/colnames to a bicluster object.

Usage

```
## S4 method for signature 'bicluster,matrix'
set_bicluster_names(bic, m)
```

Arguments

bic	A bicluster object.
m	The matrix, that was used for the biclustering. (Works only if matrix has row-/colnames.)

Value

The updated bicluster object.

#' @examples m <- matrix(c(1,2,3,4), nrow=2) rownames(m) <- c("r1", "r2") rownames(m) <- c("c1", "c2") set_bicluster_names(bicluster(row=c(1,2), column=c(1,2)), m)

`similarity_matrix` *Compute similarities between biclusters*

Description

This function computes a similarity matrix between biclusters using different similarity metrics.

Usage

```
similarity_matrix(
  bics,
  MARGIN = "both",
  metric = 1L,
  prob_scale = FALSE,
  mat_row = 0L,
  mat_col = 0L,
  prl = FALSE
)
```

Arguments

`bics`	A list of bicluster objects.
`MARGIN`	Choose if the distance is computed over `"row"` , `"column"` or `"both"`.
`metric`	Integer indicating which metric is used. 1: Bray-Curtis similarity (default), 2: Jaccard index, 3: overlap coefficient, 4: Fowlkes–Mallows index.
`prob_scale`	Scale similarity by the probability of an overlap equal of higher to the observed one. The scaling is done by multiplying the similarity with `(1 -(1 / (1 -log(overlap_probability,base=100))))`. The probability is comupted using the function `p_overlap_2d_higher` for `MARGIN ="both"` and `p_overlap_higher` otherwise. Can be helpful for big imbalances of bicluster sizes.
`mat_row`	If `prob_scale == TRUE`, the number of rows of the input matrix for biclustering must be given.
`mat_col`	If `prob_scale == TRUE`, the number of columns of the input matrix for biclustering must be given.
`prl`	Compute the similarity matrix using multiple cores (works only for `MARGIN="both"`). The number of core can be defined by executing: `RcppParallel::setThreadOptions(numThreads = 4)` before running this function.

Value

A numeric matrix of the similarities between all given biclusters.

Examples

```
b <- list(bicluster(row=c(1,2,3,4), column=c(1,2,3,4)),
        bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
similarity_matrix(b)
```

`transpose_bicluster` *Transpose a bicluster. Row and column slots will be changed.*

Description

Transpose a bicluster. Row and column slots will be changed.

Usage

```
transpose_bicluster(bic)
```

Arguments

`bic`	A bicluster object.

Value

A transposed bicluster object,

Examples

```
transpose_bicluster(bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
```

`validate_bicluster` *Indicates, whether a bicluster is valid. That means it needs at least one row and one column.*

Description

Indicates, whether a bicluster is valid. That means it needs at least one row and one column.

Usage

```
validate_bicluster(bic, minRow = 1L, minCol = 1L)
```

Arguments

`bic`	A bicluster object
`minRow`	Minimum number of required rows (Min=1).
`minCol`	Minimum number of required columns (Min=1).

Value

Logical indicating a valid bicluster object.

Examples

```
validate_bicluster(bicluster(row=c(3,4,5,6), column=c(3,4,5,6)))
```

`write_graphml` *Save adjacency matrix as GraphML file*

Description

Save and adjacency matrix as returned by `full_graph` or 1 - `distance_matrix` as a GraphML file.

Usage

```
write_graphml(m, filename, cols)
```

Arguments

`m`	A symmetric numeric matrix (Adjacency matrix). Rownames are considered as node names.
`filename`	Name of the resulting GraphML file (should end with ".gml").
`cols`	Node colors.

Value

0 if successful.

Examples

```
m <- matrix(seq(1:16), nrow=4)
# m <- matrix(rnorm(10000), nrow=100)
# bics <- c(run_fabia(m), run_isa(m), run_plaid(m))
# bn <- bicluster_network(bics, m)
# write_graphml(apply_threshold(bn), "testfile.txt")
```

write_matrix *Write an R matrix to a file (In a Bi-Force or QUBIC2 readable format).*

Description

Write an R matrix to a file (In a Bi-Force or QUBIC2 readable format).

Usage

```
write_matrix(m, filename, qubic2_format = FALSE)
```

Arguments

m	A Numeric matrix.
filename	Name of the output file.
qubic2_format	Write the matrix in a format QUBIC2 is able to read. This means adding a row- and column names to the file.

Value

0 if file was written successfully.

Examples

```
write_matrix(matrix(c(1,2,3,4), nrow=2), "testfile.txt")
```

zero_subsetting *Make a vector of R indices compatible with c++ by substracting every element by one.*

Description

Make a vector of R indices compatible with c++ by substracting every element by one.

Usage

```
zero_subsetting(v)
```

Arguments

v	A numeric vector.

Value

A numeric vector with every element decremented by one.

Examples

```
zero_subsetting(c(1,2,3,4,5))
```

Index