

# Package ‘mosbi’

May 20, 2024

**Title** Molecular Signature identification using Biclustering

**Version** 1.10.0

**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' 'mouse\_data.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\_19  
**git\_last\_commit** b968052  
**git\_last\_commit\_date** 2024-04-30  
**Repository** Bioconductor 3.19  
**Date/Publication** 2024-05-19  
**Author** Tim Daniel Rose [cre, aut],  
 Josch Konstantin Pauling [aut],  
 Nikolai Koehler [aut]  
**Maintainer** Tim Daniel Rose <[tim.rose@wzw.tum.de](mailto:tim.rose@wzw.tum.de)>

## Contents

alghistogram . . . . .	4
apply_threshold . . . . .	5
apply_threshold,bicluster_net-method . . . . .	5
apply_threshold,cooccurrence_net-method . . . . .	6
attributeConnector . . . . .	7
attribute_graph . . . . .	7
attr_overlap . . . . .	8
bicluster-class . . . . .	9
bicluster_heatmap . . . . .	9
bicluster_heatmap,bicluster,matrix-method . . . . .	10
bicluster_net-class . . . . .	11
bicluster_network . . . . .	11
bicluster_net_to_igraph . . . . .	13
bicluster_net_to_igraph,bicluster_net-method . . . . .	13
bicluster_to_matrix . . . . .	14
bicluster_to_matrix,matrix,bicluster-method . . . . .	15
check_names . . . . .	15
clean_bicluster_list . . . . .	16
colhistogram . . . . .	16
cooccurrence_net-class . . . . .	17
cooccurrence_net_to_igraph . . . . .	17
cooccurrence_net_to_igraph,cooccurrence_net-method . . . . .	18
cpp_matrix_subsetting . . . . .	19
detect_elements . . . . .	19
dim,bicluster-method . . . . .	20
distance_matrix . . . . .	20
ensemble_biclusters . . . . .	21
feature_louvain_overlap . . . . .	22

feature_network . . . . .	23
filter_biclusters . . . . .	24
filter_bicluster_size . . . . .	25
filter_matrix . . . . .	25
filter_subsets . . . . .	26
full_graph . . . . .	27
getAkmbiclustClusters . . . . .	28
getAllBFClusters . . . . .	28
getBFCluster . . . . .	29
getBicAREbiclusters . . . . .	30
getBiclustClusters . . . . .	30
getBiclustpyClusters . . . . .	31
getFabiaClusters . . . . .	32
getIsaClusters . . . . .	33
getQUBIC2biclusters . . . . .	34
get_adjacency . . . . .	34
get_adjacency,bicluster_net-method . . . . .	35
get_algorithms . . . . .	36
get_biclusters . . . . .	36
get_bic_net_algorithms . . . . .	37
get_bic_net_algorithms,bicluster_net-method . . . . .	38
get_louvain_communities . . . . .	39
get_louvain_communities,bicluster_net-method . . . . .	39
get_louvain_communities,cooccurrence_net-method . . . . .	40
has_names . . . . .	41
is_subset_or_identical . . . . .	42
mouse_data . . . . .	42
network_edge_strength . . . . .	43
network_edge_strength_float . . . . .	43
NoBFBiclusters . . . . .	44
node_size . . . . .	45
occurance_matrix . . . . .	46
occurance_table . . . . .	46
plot,bicluster_net,missing-method . . . . .	47
plot,cooccurrence_net,missing-method . . . . .	48
plot_algo_network . . . . .	48
plot_piechart_bicluster_network . . . . .	49
p_overlap . . . . .	50
p_overlap_2d . . . . .	51
p_overlap_2d_higher . . . . .	52
p_overlap_higher . . . . .	52
randomize_matrix . . . . .	53
replace_threshold . . . . .	54
replace_values . . . . .	54
replace_values_float . . . . .	55
rowhistogram . . . . .	56
run_akmbiclust . . . . .	56
run_bimax . . . . .	57

run_cc . . . . .	58
run_fabia . . . . .	58
run_isa . . . . .	59
run_plaid . . . . .	60
run_qubic . . . . .	60
run_quest . . . . .	61
run_spectral . . . . .	62
run_unibic . . . . .	62
run_xmotifs . . . . .	63
sample_biclusters . . . . .	64
select_biclusters_from_bicluster_network . . . . .	64
select_biclusters_from_bicluster_network,bicluster_net,list-method . . . . .	65
set_bicluster_names . . . . .	66
set_bicluster_names,bicluster,matrix-method . . . . .	67
similarity_matrix . . . . .	67
transpose_bicluster . . . . .	68
validate_bicluster . . . . .	69
write_graphml . . . . .	70
write_matrix . . . . .	70
zero_subsetting . . . . .	71

## Index 72

---

alghistogram	<i>Get list the list of algorithms from a list of bicluster objects.</i>
--------------	--

---

### Description

Can be used for .g. histograms.

### Usage

```
alghistogram(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))
# alghistogram(bics)
```

---

apply_threshold	<i>Apply a threshold to a bicluster similarity adjacency matrix or a co-occurrence adjacency matrix.</i>
-----------------	--

---

**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 an 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-occurrence networks.*

---

### Description

The function generates co-occurrence networks for all the attributes. E.g. if `MARGIN="column"`, for each column, a co-occurrence 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 <b>biclusters</b> .
m	The matrix used for biclustering.
MARGIN	"row" or "column", 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 occurrence of all elements in the biclusters.

**Usage**

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

**Arguments**

bics	A list of <b>bicluster</b> objects.
named	Boolean, indicating, if all bicluster objects have names.

**Value**

A Data Frame with the counts of 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	<i>A S4 class to store biclusters.</i>
-----------------	--

---

**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	<i>Plot a heatmap of a bicluster</i>
-------------------	--------------------------------------

---

**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")
colnames(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: <a href="#">heatmap</a> .

**Value**

A plot object

**Examples**

```
m <- matrix(c(1,2,3,4), nrow=2)
rownames(m) <- c("r1", "r2")
colnames(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. by 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 <a href="#">similarity_matrix</a> .
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 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 $(1 - (1 / (1 - \log(\text{overlap\_probability}, \text{base}=100))))$ . The probability is computed using the f unction <a href="#">p_overlap_2d_higher</a> for MARGIN=="both" and <a href="#">p_overlap_higher</a> otherwise. Can be helpful for big imbalances of bicluster sizes.
pr1	Compute the similarity matrix using multiple cores (works only for MARGIN=="both"). The number of core can be defined by executing: <code>RcppParallel::setThreadOptions(numThreads = 4)</code> 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 actual submatrix of the original matrix.*

---

**Description**

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

**Usage**

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

**Arguments**

<code>m</code>	Matrix on which the bicluster was computed
<code>bic</code>	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 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)))
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 lengths 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-occurrence networks.*

---

**Description**

Object that is returned e.g. by the function [feature\\_network](#).

**Slots**

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

`threshold` Estimated threshold for the co-occurrence 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-occurrence 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 ( <a href="#">bicluster_nets</a> ) as outputted by <a href="#">get_louvain_communities</a> .
bics	The list biclusters that was used for calculation with <a href="#">bicluster_network</a> .
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	<i>Generate a co-occurrence network</i>
-----------------	---

---

### Description

The function computes a co-occurrence 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 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 <a href="#">full_graph</a> .

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 of 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

<code>bics</code>	List of bicluster objects.
<code>minRow</code>	Minimum number of rows.
<code>minCol</code>	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 replaced by 0.

### Usage

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

### Arguments

<code>mat</code>	A Numeric matrix.
<code>threshold</code>	All values below will be replaced 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	<i>Remove all biclusters from a list, that are identical or perfect subsets from each other. Additionally all invalid biclusters are removed (See <a href="#">validate_bicluster</a>).</i>
----------------	--

---

**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 perfect 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	<i>Generate a similarity network for a list of biclusters</i>
------------	---

---

### 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

<code>bics</code>	A result object from akmbiclust.
<code>mat</code>	Original matrix, that was used for biclustering.
<code>transposed</code>	True, if the bicluster calculation was performed on a transposed matrix.
<code>filterfun</code>	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) <code>filterfun(bicluster_matrix, ...)</code> .
<code>...</code>	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)
```

---

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

---

### Description

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

### Usage

```
getBicAREbicclusters(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)
# getBicAREbicclusters(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**

<code>bics</code>	A biclust object.
<code>mat</code>	Original matrix, that was used for biclustering.
<code>method</code>	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".
<code>transposed</code>	True, if the bicluster calculation was performed on a tranposed matrix.
<code>filterfun</code>	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 <code>mat</code> ) <code>filterfun(bicluster_matrix, ...)</code> .
<code>...</code>	Other parameters forwarded to the <code>filterfun</code> .

**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	<i>Extract a list of bicluster objects from an fabia biclustering object.</i>
------------------	---

---

**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	<i>Extract a list of bicluster objects from an isa2 biclustering object.</i>
----------------	--

---

**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 transposed 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	<i>Get Algorithms</i>
----------------	-----------------------

---

**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	<i>Extract biclusters from different algorithms/packages</i>
----------------	--

---

**Description**

Converts biclusters output of different algorithms/packages in to lists of `bicluster` objects. Many algorithms 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", "bi-force", "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 <code>bicluster_net</code> .
---------	---

**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**

<code>bic_net</code>	A <code>bicluster_net</code> or <code>cooccurrence_net</code> object.
<code>min_size</code>	Minimum size of a louvain community to be returned (minimum value is 2).
<code>bics</code>	Optional. Is only use for the class <code>bicluster_net</code> . 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 <a href="#">bicluster_net</a> 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 <a href="#">cooccurrence_net</a> 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	<i>Check, whether a matrix has row- and colnames.</i>
-----------	---

---

**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")
colnames(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 `bic1` 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)))
```

---

`mouse_data`

*Mouse brain lipidomics data*

---

**Description**

Development of the mice brain lipidome over several weeks.

**Usage**

```
data(mouse_data)
```

**Format**

A data frame with 245 rows and 61 columns

**Source**

<https://www.ebi.ac.uk/metabolights/MTBLS562>

**Examples**

```
# data(mouse_data)
# View(mouse_data)
```

---

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  $\max(\text{adjm})$  are calculated. It is assumed 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 symmetric numeric matrix.

**Value**

A numeric matrix of  $\dim(\max(\text{adjm}), 2)$ . The first column indicates 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 symmetric 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 assumed that the matrix is symmetric and therefore the number of edges divided by two. Uses the function [replace\\_values\\_float](#).

**Value**

A numeric matrix of dim(max(adjm), 2). The first column indicates 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	<i>Node sizes for plotting bicluster networks.</i>
-----------	--

---

**Description**

When plotting bicluster networks, node sizes adapted to bicluster sizes can improve visual inspection. Node sizes are computed using the following formula:  $(\text{atan}((x - \min(x)) / (\max(x) - \min(x)) + \text{offset})) * \text{base\_size})$ . 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 <code>bicluster</code> 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))
```

---

occurrence_matrix	<i>Occurance matrix of data points in a list of biclusters</i>
-------------------	--

---

### Description

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

### Usage

```
occurrence_matrix(bics, mat)
```

### Arguments

bics	A list of <code>bicluster</code> 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 occurrence 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))
# occurrence_matrix(bics, m)
```

---

occurrence_table	<i>Occurance table of data points in a list of biclusters</i>
------------------	---

---

### Description

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

### Usage

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

### Arguments

bics	A list of <code>bicluster</code> 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 occurrence 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))
# occurrence_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 <a href="#">bicluster_net</a> .
y	Not used.
...	Plot parameters forwarded to <code>igraph::plot.igraph</code>

**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 <a href="#">cooccurrence_net</a> .
y	Not used.
...	Plot parameters forwarded to <a href="#">igraph::plot.igraph</a>

### 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**

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

**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 <code>bicluster_net</code> object.
bics	The corresponding list of biclusters from <code>bic_net</code> . After calculating communities with <code>get_louvain_communities</code> it is necessary to get the subset of biclusters using <code>select_biclusters_from_bicluster_network</code> .
class_vector	A (named) vector with class affinities. Every occurring element in the biclusters 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 <code>sort(unique(class_vector))</code> .
named	Indicates if rowname/colname of the bicluster objects should be used instead of the indices.
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 <code>plot.igraph</code> .

**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 formula  $\frac{\binom{y}{x} \times \binom{n-y}{k-x}}{\binom{n}{k}}$ .

**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)
```

---

<code>p_overlap_2d</code>	<i>Probability for an overlap of two dimensional samples</i>
---------------------------	--

---

**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	<i>Randomize a matrix</i>
------------------	---------------------------

---

**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  $\geq$  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 $\geq$ 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	<i>Get the rowlengths for a list of bicluster objects.</i>
--------------	--

---

**Description**

Can be used for e.g. histograms.

**Usage**

```
rowhistogram(bic)
```

**Arguments**

`bic`                    A list of bicluster objects.

**Value**

A vector with the lengths 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	<i>Run the akmbiclust biclustering algorithm</i>
----------------	--

---

**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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCBimax</a> 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	<i>Run the CC biclustering algorithm</i>
--------	--

---

### 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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCCC</a> 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	<i>Run the fabia biclustering algorithm</i>
-----------	---

---

### 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**

<code>data_matrix</code>	A numeric matrix.
<code>minRow</code>	Same parameters as in <a href="#">filter_bicluster_size</a> .
<code>minCol</code>	Same parameters as in <a href="#">filter_bicluster_size</a> .
<code>thresZ</code>	See parameter from the <a href="#">extractBic</a> function.
<code>thresL</code>	See parameter from the <a href="#">extractBic</a> function.
<code>...</code>	Other parameters forwarded to the <a href="#">fabia</a> 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)
```

---

<code>run_isa</code>	<i>Run the isa biclustering algorithm</i>
----------------------	---

---

**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**

<code>data_matrix</code>	A numeric matrix.
<code>minRow</code>	Same parameters as in <a href="#">filter_bicluster_size</a> .
<code>minCol</code>	Same parameters as in <a href="#">filter_bicluster_size</a> .
<code>...</code>	Other parameters forwarded to the <a href="#">isa</a> 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	<i>Run the Plaid biclustering algorithm</i>
-----------	---

---

**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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCPlaid</a> 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	<i>Run the QUBIC biclustering algorithm</i>
-----------	---

---

**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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCQU</a> 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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCQuest</a> 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	<i>Run the spectral biclustering algorithm</i>
--------------	--

---

### 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 <a href="#">filter_bicluster_size</a> .
minCol	Same parameters as in <a href="#">filter_bicluster_size</a> .
...	Other parameters forwarded to the <a href="#">BCSpectral</a> 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	<i>Run the UniBic biclustering algorithm</i>
------------	--

---

### 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.

**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)

```

---

<code>run_xmotifs</code>	<i>Run the Xmotifs biclustering algorithm</i>
--------------------------	---

---

**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	<i>Sample a list of biclusters.</i>
-------------------	-------------------------------------

---

### Description

The function generates a list of biclusters given an input list of biclusters, where each bicluster has the same number of rows and columns, but with sampled entries from a uniform distribution of all rows and columns in 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	<i>Create a subset of biclusters based on a bicluster network</i>
--	---

---

### 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 or higher to the observed one. The scaling is done by multiplying the similarity with $(1 - (1 / (1 - \log(\text{overlap\_probability}, \text{base}=100))))$ . The probability is computed using the function <code>p_overlap_2d_higher</code> for MARGIN=="both" and <code>p_overlap_higher</code> 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.
pr1	Compute the similarity matrix using multiple cores (works only for MARGIN="both"). The number of core can be defined by executing: <code>RcppParallel::setThreadOptions(numThreads = 4)</code> 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	<i>Indicates, whether a bicluster is valid. That means it needs at least one row and one column.</i>
--------------------	--

---

**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	<i>Save adjacency matrix as GraphML file</i>
---------------	--

---

### 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

<code>m</code>	A symmetric numeric matrix (Adjacency matrix). Rownames are considered as node names.
<code>filename</code>	Name of the resulting GraphML file (should end with ".gml").
<code>cols</code>	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	<i>Write an R matrix to a file (In a Bi-Force or QUBIC2 readable format).</i>
--------------	---

---

### 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	<i>Make a vector of R indices compatible with c++ by subtracting every element by one.</i>
-----------------	--

---

**Description**

Make a vector of R indices compatible with c++ by subtracting 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

## \* datasets

- mouse\_data, 42
  
- akmbiclust, 56
- alghistogram, 4
- apply\_threshold, 5
- apply\_threshold, bicluster\_net-method, 5
- apply\_threshold, cooccurrence\_net-method, 6
- attr\_overlap, 8, 22
- attribute\_graph, 7, 7
- attributeConnector, 7
  
- BCBimax, 57
- BCCC, 58
- BCPlaid, 60
- BCQU, 60, 61
- BCQuest, 61
- BCSpectral, 62
- BCXmotifs, 63
- bicluster, 8, 21, 28, 30–33, 36, 37, 45, 46, 57–65
- bicluster (bicluster-class), 9
- bicluster-class, 9
- bicluster\_heatmap, 9
- bicluster\_heatmap, bicluster, matrix-method, 10
- bicluster\_net, 12, 13, 21, 39, 40, 47, 49, 50, 64, 65
- bicluster\_net (bicluster\_net-class), 11
- bicluster\_net-class, 11
- bicluster\_net\_to\_igraph, 13
- bicluster\_net\_to\_igraph, bicluster\_net-method, 13
- bicluster\_network, 11, 11, 21
- bicluster\_to\_matrix, 14
- bicluster\_to\_matrix, matrix, bicluster-method, 15
  
- check\_names, 15
- clean\_bicluster\_list, 16
- cluster\_louvain, 39, 40
- colhistogram, 16
- cooccurrence\_net, 17, 18, 24, 39, 40, 48
- cooccurrence\_net (cooccurrence\_net-class), 17
- cooccurrence\_net-class, 17
- cooccurrence\_net\_to\_igraph, 17
- cooccurrence\_net\_to\_igraph, cooccurrence\_net-method, 18
- cpp\_matrix\_subsetting, 19
  
- detect\_elements, 19
- dim, bicluster-method, 20
- distance\_matrix, 20, 70
  
- ensemble\_biclusters, 21
- extractBic, 59
  
- fabia, 58, 59
- feature\_louvain\_overlap, 22
- feature\_network, 17, 23
- filter\_bicluster\_size, 25, 56–63
- filter\_biclusters, 24
- filter\_matrix, 25
- filter\_subsets, 26
- full\_graph, 7, 23, 24, 27, 70
  
- get\_adjacency, 34
- get\_adjacency, bicluster\_net-method, 35
- get\_algorithms, 36
- get\_bic\_net\_algorithms, 37
- get\_bic\_net\_algorithms, bicluster\_net-method, 38
- get\_biclusters, 36, 65
- get\_louvain\_communities, 21, 39, 49, 50, 64, 65
- get\_louvain\_communities, bicluster\_net-method, 39



get\_louvain\_communities, cooccurrence\_net-method, [plaid](#), [60](#)  
     [40](#)  
 getAkmbiclustClusters, [28](#)  
 getAllBFClusters, [28](#)  
 getBFCluster, [29](#), [29](#)  
 getBicAREbicclusters, [30](#)  
 getBiclustClusters, [30](#)  
 getBiclustpyClusters, [31](#)  
 getFabiaClusters, [32](#)  
 getIsaClusters, [33](#)  
 getQUBIC2bicclusters, [34](#)  
 graph, [13](#), [14](#), [18](#), [47](#), [48](#)  
  
 has\_names, [41](#)  
 heatmap, [9](#), [10](#)  
  
 is\_subset\_or\_identical, [42](#)  
 isa, [59](#)  
  
 mouse\_data, [42](#)  
  
 network\_edge\_strength, [43](#), [43](#)  
 network\_edge\_strength\_float, [43](#)  
 NoBFBicclusters, [44](#)  
 node\_size, [45](#)  
  
 occurrence\_matrix, [46](#), [46](#)  
 occurrence\_table, [46](#)  
  
 p\_overlap, [50](#), [51](#), [52](#)  
 p\_overlap\_2d, [51](#), [52](#)  
 p\_overlap\_2d\_higher, [12](#), [52](#), [68](#)  
 p\_overlap\_higher, [12](#), [52](#), [68](#)  
 plot, bicluster\_net, missing-method, [47](#)  
 plot, cooccurrence\_net, missing-method,  
     [48](#)  
 plot.igraph, [47–50](#)  
 plot\_algo\_network, [48](#)  
 plot\_piechart\_bicluster\_network, [49](#)  
  
 randomize\_matrix, [53](#)  
 replace\_threshold, [54](#)  
 replace\_values, [43](#), [54](#), [55](#)  
 replace\_values\_float, [44](#), [55](#)  
 rowhistogram, [56](#)  
 run\_akmbiclust, [56](#)  
 run\_bimax, [57](#)  
 run\_cc, [58](#)  
 run\_fabia, [58](#)  
 run\_isa, [59](#)  
  
 run\_qubic, [60](#)  
 run\_quest, [61](#)  
 run\_spectral, [62](#)  
 run\_unibic, [62](#)  
 run\_xmotifs, [63](#)  
  
 sample\_biclust, [64](#)  
 select\_biclusters\_from\_bicluster\_network,  
     [49](#), [50](#), [64](#)  
 select\_biclusters\_from\_bicluster\_network, bicluster\_net, lis  
     [65](#)  
 set\_bicluster\_names, [66](#)  
 set\_bicluster\_names, bicluster, matrix-method,  
     [67](#)  
 similarity\_matrix, [12](#), [67](#)  
  
 transpose\_bicluster, [68](#)  
  
 validate\_bicluster, [25](#), [26](#), [28](#), [30–34](#), [37](#),  
     [69](#)  
  
 write\_graphml, [70](#)  
 write\_matrix, [70](#)  
  
 zero\_subsetting, [71](#)