Package 'MetNet'

November 12, 2026

Type Package

Title Inferring metabolic networks from untargeted high-resolution mass spectrometry data

Version 1.30.0

Date 2025-10-24

VignetteBuilder knitr

Depends R (>= 4.1), S4Vectors (>= 0.28.1), SummarizedExperiment (>= 1.20.0)

Imports bnlearn (>= 4.3), BiocParallel (>= 1.12.0), corpcor (>= 1.6.10), dplyr (>= 1.0.3), ggplot2 (>= 3.3.3), GeneNet (>= 1.2.15), GENIE3 (>= 1.7.0), methods (>= 4.1), parmigene (>= 1.0.2), psych (>= 2.1.6), rlang (>= 0.4.10), stabs (>= 0.6), stats (>= 4.1), tibble (>= 3.0.5), tidyr (>= 1.1.2)

Suggests BiocGenerics (>= 0.24.0), BiocStyle (>= 2.6.1), glmnet (>= 4.1-1), igraph (>= 1.1.2), knitr (>= 1.11), rmarkdown (>= 1.15), testthat (>= 2.2.1), Spectra (>= 1.4.1), MsCoreUtils (>= 1.6.0)

biocViews ImmunoOncology, Metabolomics, MassSpectrometry, Network, Regression

Description MetNet contains functionality to infer metabolic network topologies from quantitative data and high-resolution mass/charge information. Using statistical models (including correlation, mutual information, regression and Bayes statistics) and quantitative data (intensity values of features) adjacency matrices are inferred that can be combined to a consensus matrix. Mass differences calculated between mass/charge values of features will be matched against a data frame of supplied mass/charge differences referring to transformations of enzymatic activities. In a third step, the two levels of information are combined to form a adjacency matrix inferred from both quantitative and structure information.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

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

git_branch RELEASE_3_23

git_last_commit 65a9cd0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Thomas Naake [aut, cre],
Liesa Salzer [ctb],
Elva Maria Novoa-del-Toro [ctb] (ORCID:
<https://orcid.org/0000-0002-6135-5839>)

Maintainer Thomas Naake <thomasnaake@googlemail.com>

Contents

MetNet-package . . . 3
.AdjacencyMatrix . . . 4
.assays_have_identical_colnames_rownames . . . 4
.assays_have_identical_dimnames . . . 5
addSpectralSimilarity . . . 6
addToList . . . 7
AdjacencyMatrix . . . 8
AdjacencyMatrix-class . . . 9
AllGenerics . . . 11
aracne . . . 11
bayes . . . 12
clr . . . 13
combine . . . 14
correlation . . . 16
getLinks . . . 17
lasso . . . 18
mat_test . . . 19
mat_test_z . . . 19
ms2_test . . . 20
mz_summary . . . 20
mz_vis . . . 22
partialCorrelation . . . 23
peaklist . . . 24
randomForest . . . 24
rtCorrection . . . 25
spectra_matrix . . . 27
statistical . . . 28
structural . . . 29
threshold . . . 30
topKnet . . . 33
x_annotation . . . 34
x_test . . . 35

Index **38**

`MetNet-package` *Inferring metabolic networks from untargeted high-resolution mass spectrometry data*

Description

Inferring metabolic networks from untargeted high-resolution mass spectrometry data.

Details

The package infers network topologies from quantitative data (intensity values) and structural data (m/z values of mass features). `MetNet` combines these two data sources to a consensus matrix.

Author(s)

Author: Thomas Naake [aut, cre], Liesa Salzer [ctb], Elva Maria Novoa-del-Toro [ctb] (ORCID: <https://orcid.org/0000-0002-6135-5839>) Maintainer: Thomas Naake <thomasnaake@googlemail.com>

References

Breitling, R. et al. Ab initio prediction of metabolic networks using Fourier transform mass spectrometry data. 2006. Metabolomics 2: 155–164. 10.1007/s11306-006-0029-z

Examples

```
data("x_test", package = "MetNet")
x_test <- as.matrix(x_test)
functional_groups <- rbind(
    c("Hydroxylation (-H)", "O", "15.9949146221"),
    c("Malonyl group (-H2O)", "C3H2O3", "86.0003939305"),
    c("C6H10O6", "C6H10O6", "178.0477380536"),
    c("D-ribose (-H2O) (ribosylation)", "C5H8O4", "132.0422587452"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Glucuronic acid (-H2O)", "C6H8O6", "176.0320879894"),
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
functional_groups <- data.frame(group = functional_groups[,1],
    formula = functional_groups[,2],
    mass = as.numeric(functional_groups[,3]))
struct_adj <- structural(x_test, functional_groups, ppm = 5)

stat_adj_l <- statistical(x_test,
    model = c("pearson", "spearman","bayes"))
args_top1 <- list(n = 10)
stat_adj <- threshold(stat_adj_l, type = "top2", args = args_top1)
cons_adj <- combine(struct_adj, stat_adj)
```

`.AdjacencyMatrix` *Create S4 class AdjacencyMatrix*

Description

The class ‘AdjacencyMatrix‘ extends the ‘SummarizedExperiment‘ class. It will add the slots ‘type‘, ‘directed‘, and ‘thresholded‘.

Details

The slot ‘type‘ is of type ‘"character"‘, storing the type of the ‘"AdjacencyMatrix"‘, i.e. ‘"structural"‘, ‘"statistical"‘, or ‘"combined"‘. The slot ‘directed‘ is of type ‘"logical"‘, storing if the adjacency matrix is directed or not. The slot ‘thresholded‘ is of type ‘"logical"‘, storing if the adjacency matrix was thresholded, e.g. if the functions ‘rtCorrection‘ or ‘threshold‘ were applied on the ‘structural‘ or ‘statistical‘ ‘AdjacencyMatrix‘ objects.

If any of the ‘AdjacencyMatrix‘ objects passed to the ‘combine‘ function was ‘directed = TRUE‘ or ‘thresholded = TRUEs‘ the ‘combine‘ ‘AdjacencyMatrix‘ object will be ‘directed = TRUE‘ or ‘thresholded = TRUE‘.

Value

class generator function for class ‘AdjacencyMatrix‘

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

`.assays_have_identical_colnames_rownames`
Check if all the assays in the ‘AdjacencyMatrix‘ object have identical colnames and rownames

Description

The function will check if all the assays in the ‘AdjacencyMatrix‘ object have identical colnames and rownames.

Usage

```
.assays_have_identical_colnames_rownames(object)
```

Arguments

`object`	‘AdjacencyMatrix‘ object

Details

Helper function for validity check of ‘AdjacencyMatrix‘ objects.

Value

‘logical‘ of length 1

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

`.assays_have_identical_dimnames` *Check if the assays in the ‘AdjacencyMatrix‘ object have identical dimnames*

Description

The function will check if the assays in the ‘AdjacencyMatrix‘ object have identical dimnames.

Usage

```
.assays_have_identical_dimnames(object)
```

Arguments

`object`	‘AdjacencyMatrix‘ object

Details

Helper function for validity check of ‘AdjacencyMatrix‘ objects.

Value

‘logical‘ of length 1

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

addSpectralSimilarity *Adding a spectral similarity matrix to the "structural" 'AdjacencyMatrix'*

Description

The function 'addSpectralSimilarity' adds adjacency matrices from spectral similarity into the "structural" 'AdjacencyMatrix' object. One or multiple spectral similarity matrices can be added to the "structural" 'AdjacencyMatrix' object.

Usage

```
addSpectralSimilarity(am_structural, ms2_similarity = list())
```

Arguments

am_structural	'AdjacencyMatrix' of type "structural" that was created using matching MS1 data of the same data set. The respective spectral similarity matrices will be added into 'am_structural'
ms2_similarity	'list' containing spectral similarity adjacency matrices with matching row-/colnames of the structural 'AdjacencyMatrix'. The name of the list entries should reference to the similarity calcululation method (e.g. "ndotproduct")

Details

The function 'addSpectralSimilarity' includes functionality to add spectral adjacency matrices e.g. that were created by functionality from the 'RforMassSpectrometry' infrastructure. 'addSpectralSimilarity' iterates through a 'list' with named spectral similarity matrices and adds them to the "structural" 'AdjacencyMatrix'. Matching between spectral similarity and "structural" 'AdjacencyMatrix' is performed via rownames/colnames. Thus, it is important that the spectral similarity matrices have row/colnames matching to the respective MS1 data. 'addSpectralSimilarity' will add the adjacency matrices and will return the "structural" 'AdjacencyMatrix' containing the added weighted adjacency matrices in the 'assays' slot.

Value

'AdjacencyMatrix' of type "structural" containing the respective adjacency matrices in the 'assay' slot as specified by 'methods'

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

Examples

```
data("x_test", package = "MetNet")
transformation <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
transformation <- data.frame(group = transformation[, 1],
    formula = transformation[, 2],
    mass = as.numeric(transformation[, 3]))
am_struct <- structural(x_test, transformation, var = c("group", "mass"),
    ppm = 10, directed = TRUE)

## load the file containing MS2 similarities
f <- system.file("spectra_matrix/spectra_matrix.RDS", package = "MetNet")
adj_spec <- readRDS(f)

## run the addSpectralSimilarity function
spect_adj <- addSpectralSimilarity(am_structural = am_struct,
   ms2_similarity = list("ndotproduct" = adj_spec))
```

`addToList` *Add adjacency matrix to list*

Description

This helper function used in the function ‘statistical‘ adds an adjacency matrix to a ‘list‘ of adjacency matrices.

Usage

```
addToList(l, name, object)
```

Arguments

`l`	‘list‘ of adjacency matrices
`name`	‘character‘, name of added entry
`object`	‘matrix‘ that will be added

Details

The function ‘addToList‘ is a helper function used internally in ‘statistical‘.

Value

‘list‘ containing the existing matrices and the added matrix

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
cor_pearson <- correlation(x, method = "pearson")
cor_spearman <- correlation(x, method = "spearman")
l <- list(pearson = cor_pearson)
MetNet:::addToList(l, "spearman_coef", cor_spearman$r)
```

`AdjacencyMatrix` *Wrapper to create an instance of S4 class AdjacencyMatrix*

Description

The function 'AdjacencyMatrix' will create an object of type 'AdjacencyMatrix'.

Usage

```
AdjacencyMatrix(
  adj_l,
  rowData,
  type = c("structural", "statistical", "combine"),
  directed = c(TRUE, FALSE),
  thresholded = c(TRUE, FALSE)
)
```

Arguments

`adj_l`	'list' of adjacency matrices
`rowData`	'data.frame', containing information on the features
`type`	'character', either '"structural"', '"statistical"', or '"combine"'
`directed`	'logical', if the adjacency matrix underlying the graph is directed or undirected
`thresholded`	'logical', if the functions 'rtCorrection' or 'threshold' were applied on the 'structural' or 'statistical' 'AdjacencyMatrix' objects

Details

'adj_l' is a list of adjacency matrices. The adjacency matrices have identical dimensions and 'dimnames' and each adjacency matrix has the same number of columns and rows and identical 'rownames' and 'colnames'. 'rowData' will be also used for the 'colData' slot (since the 'rownames' and 'colnames' are identical).

Value

object of S4 class ‘AdjacencyMatrix‘

Accessors

- The ‘AdjacencyMatrix‘ class extends the [SummarizedExperiment::SummarizedExperiment] class and inherits all its accessors and replacement methods.

- The ‘type‘ accessor returns the ‘type‘ (‘"structural"‘, ‘"statistical"‘, ‘"combine"‘) slot.

- The ‘directed‘ accessor returns the ‘directed‘ (‘logical‘ of length 1) slot.

- The ‘thresholded‘ accessor returns the ‘thresholded‘ (‘logical‘ of length 1) slot.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
binary <- matrix(0, ncol = 10, nrow = 10)
transformation <- matrix("", ncol = 10, nrow = 10)
mass_difference <- matrix("", ncol = 10, nrow = 10)

rownames(binary) <- rownames(transformation) <- rownames(mass_difference) <- paste("feature", 1:10)
colnames(binary) <- rownames(transformation) <- rownames(mass_difference) <- paste("feature", 1:10)

binary[5, 4] <- 1
transformation[5, 4] <- "glucose addition"
mass_difference[5, 4] <- "162"

## create adj_l and rowData
adj_l <- list(binary = binary, transformation = transformation,
    mass_difference = mass_difference)
rowData <- DataFrame(features = rownames(binary),
    row.names = rownames(binary))

AdjacencyMatrix(adj_l = adj_l, rowData = rowData, type = "structural",
    directed = TRUE, thresholded = FALSE)
```

AdjacencyMatrix-class *Methods for ‘AdjacencyMatrix‘ objects*

Description

‘length‘ returns the length of an ‘AdjacencyMatrix‘ object (number of rows of an assay). ‘length‘ returns a ‘numeric‘ of length 1.

‘dim‘ returns the length of an ‘AdjacencyMatrix‘ object (number of rows of an assay, number of cols of an assay). ‘dim‘ returns a ‘numeric‘ of length 2.

'type' will return the type of an 'AdjacencyMatrix' ('statistical', 'structural' or 'combine'). 'type' returns a 'character' of length 1

'directed' returns the information on directed of an 'AdjacencyMatrix', i.e. if the underlying graph is directed or undirected. 'directed' returns 'logical' of length 1.

'thresholded' returns the information if the adjacency matrix is thresholded, i.e. if the function 'rtCorrection' or 'threshold' was applied to the 'AdjacencyMatrix' object. 'thresholded' returns a 'logical' of length 1.

'show' prints summary information on an object of class 'AdjacencyMatrix'.

'as.data.frame' returns the adjacency matrices (stored in the 'assays' slot) and returns information on the nodes and the associated information on edges as a data frame. 'as.data.frame' returns a 'data.frame'.

Usage

```
## S4 method for signature 'AdjacencyMatrix'
length(x)

## S4 method for signature 'AdjacencyMatrix'
dim(x)

## S4 method for signature 'AdjacencyMatrix'
type(x)

## S4 method for signature 'AdjacencyMatrix'
directed(object)

## S4 method for signature 'AdjacencyMatrix'
thresholded(object)

## S4 method for signature 'AdjacencyMatrix'
show(object)

## S4 method for signature 'AdjacencyMatrix'
as.data.frame(x)
```

Arguments

x	instance of class 'AdjacencyMatrix'
object	instance of class 'AdjacencyMatrix'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

AllGenerics — *Placeholder for generics functions documentation*

Description

Placeholder for generics functions documentation

aracne — *Create an adjacency matrix based on algorithm for the reconstruction of accurate cellular networks*

Description

'aracne' infers an adjacency matrix using the algorithm for the reconstruction of accurate cellular networks using the 'aracne.a' function from the 'parmigene' package. The function 'aracne' will return the weighted adjacency matrix of the inferred network after applying 'aracne.a'.

Usage

```
aracne(mi, eps = 0.05, ...)
```

Arguments

mi	matrix, where columns are the samples and the rows are features (metabolites), cell entries are mutual information values between the features. As input, the mutual information (e.g. raw MI estimates) from the 'knnmi.all' function of the 'parmigene' package can be used.
eps	numeric, used to remove the weakest edge of each triple of nodes
...	not used here

Details

For more details on the 'aracne.a' function, refer to '?parmigene::aracne.a'. 'aracne.a' considers each triple of edges independently and removes the weakest one if $MI(i,j) < MI(j,k) - eps$ and $MI(i,j) < MI(i,k) - eps$. See Margolin et al. (2006) for further information.

Value

matrix, matrix with edges inferred from Reconstruction of accurate cellular networks algorithm 'aracne.a'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

References

Margolin et al. (2006): ARACNE : An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics, S7, doi: [10.1186/1471-2105-7-S1-S7](https://doi.org/10.1186/1471-2105-7-S1-S7)

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
x_z <- apply(x, 1, function(y) (y - mean(y)) / sd(y))
mi_x_z <- parmigene::knnmi.all(x_z)
aracne(mi_x_z, eps = 0.05)
```

`bayes` *Create an adjacency matrix based on score-based structure learning algorithm*

Description

'bayes' infers an adjacency matrix using score-based structure learning algorithm 'boot.strength' from the 'bnlearn' package. 'bayes' extracts then the reported connections from running the 'boot.strength' function and assigns the strengths of the arcs of the Bayesian connections to an adjacency matrix. 'bayes' returns this weighted adjacency matrix.

Usage

```
bayes(x, algorithm = "tabu", R = 100, ...)
```

Arguments

`x`	'matrix' where columns are the samples and the rows are features (metabolites), cell entries are intensity values
`algorithm`	'character', structure learning to be applied to the bootstrap replicates (default is '"tabu"')
`R`	'numeric', number of bootstrap replicates
`...`	parameters passed to 'boot.strength'

Details

'boot.strength' measures the strength of the probabilistic relationships by the arcs of a Bayesian network, as learned from bootstrapped data. By default 'bayes' uses the Tabu greedy search.

For use of the parameters used in the 'boot.strength' function, refer to '?bnlearn::boot.strength'. For further information see also Friedman et al. (1999) and Scutari and Nagarajan (2001).

Value

'matrix' with edges inferred from score-based structure learning algorithm 'boot.strength'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

References

Friedman et al. (1999): Data Analysis with Bayesian Networks: A Bootstrap Approach. Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence, 196-201.

Scutari and Nagarajan (2011): On Identifying Significant Edges in Graphical Models. Proceedings of the Workshop Probabilistic Problem Solving in Biomedicine of the 13th Artificial Intelligence in Medicine Conference, 15-27.

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
bayes(x, algorithm = "tabu", R = 100)
```

---

clr *Create an adjacency matrix based on context likelihood or relatedness network*

Description

'clr' infers an adjacency matrix using context likelihood/relatedness network using the 'clr' function from the 'parmigene' package. 'clr' will return the adjacency matrix containing the Context Likelihood of Relatedness Network-adjusted scores of Mutual Information values.

Usage

```
clr(mi, ...)
```

Arguments

mi	matrix, where columns are samples and the rows are features (metabolites), cell entries are mutual information values between the features. As input, the mutual information (e.g. raw MI estimates) from the 'knnmi.all' function of the 'parmigene' package can be used.
...	not used here

Details

For more details on the ‘clr‘ function, refer to ‘?parmigene::clr‘. CLR computes the score $sqrt(z_i^2 + z_j^2)$ for each pair of variables i, j, where $z_i = max(0, (I(X_i, X_j) - mean(X_i))/sd(X_i))$. $mean(X_i)$ and $sd(X_i)$ are the mean and standard deviation of the mutual information values $I(X_i, X_k)$ for all $k = 1, ..., n$. For more information on the CLR algorithm see Faith et al. (2007).

Value

matrix, matrix with edges inferred from Context Likelihood of Relatedness Network algorithm ‘clr‘

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

References

Faith et al. (2007): Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles. PLoS Biology, e8, doi: [10.1371/journal.pbio.0050008](https://doi.org/10.1371/journal.pbio.0050008)

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
x_z <- apply(x, 1, function(y) (y - mean(y)) / sd(y))
mi_x_z <- parmigene::knnmi.all(x_z)
clr(mi_x_z)
```

---

combine *Combine structural and statistical ‘AdjacencyMatrix‘ objects*

Description

The function ‘combine‘ takes as input the structural and statistical ‘AdjacencyMatrix‘ objects, created in former steps. It will access the assays ‘binary‘ and ‘consensus‘, adds them together and will report a connection between metabolites if the edge is present in both matrices.

‘combine‘ returns an ‘AdjacencyMatrix‘ containing this consensus matrix supported by the structural and statistical adjacency matrices (assay ‘combine_binary‘). The ‘AdjacencyMatrix‘ object furthermore contains the assays from the statistical ‘AdjacencyMatrix‘ and the combined assays from the structural ‘AdjacencyMatrix‘, e.g. if the structural ‘AdjacencyMatrix‘ has the assays ‘group‘ and ‘mass‘, the combine ‘AdjacencyMatrix‘ object will contain the assays ‘combine_group‘ and ‘combine_mass‘ that have support from the structural and statistical ‘AdjacencyMatrix‘ object.

Usage

```
combine(am_structural, am_statistical)
```

Arguments

am_structural	'AdjacencyMatrix' containing the 'numeric' structural adjacency matrix (assay 'binary') and other 'character' or 'numeric' structural and spectral similarity adjacency matrices (e.g. 'group', 'mass' or spectral similarity as 'ndotprodcut').
am_statistical	'AdjacencyMatrix' containing the assay 'consensus' and other 'numeric' adjacency matrices depending on the chosen statistical models

Details

The matrices from the assays 'binary' and 'consensus' will be added and an unweighted connection will be reported when the edges are respectively present in both 'binary' and 'consensus'.

Value

'AdjacencyMatrix' object containing the assays 'combine_binary' ('numeric' adjacency matrix), and the combined matrices derived from the structural 'AdjacencyMatrix' ('character' adjacency matrices).

The 'AdjacencyMatrix' object will also contain all other assays contained in 'am_structural' and 'am_statistical'.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x_test <- as.matrix(x_test)
transformation <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
transformation <- data.frame(group = transformation[, 1],
    formula = transformation[, 2],
    mass = as.numeric(transformation[, 3]))

## create AdjacencyMatrix object of type structural
am_struct <- structural(x_test, transformation, var = c("group", "mass"),
    ppm = 10)

## create AdjacencyMatrix object of type statistical
x_test_cut <- as.matrix(x_test[, -c(1:2)])
am_stat <- statistical(x_test_cut, model = c("pearson", "spearman"),
    adjust = "bonferroni")
am_stat <- threshold(am_stat, type = "top2", args = list(n = 10))

## combine
combine(am_structural = am_struct, am_statistical = am_stat)
```

`correlation` *Create an adjacency matrix based on correlation*

Description

‘correlation‘ infers an adjacency matrix using correlation using the ‘corr.test‘ function (from the ‘psych‘ package) or ‘partialCorrelation‘. ‘correlation‘ extracts the reported pair-wise correlation coefficients from the function ‘corr.test‘ and ‘partialCorrelation‘ and will return the weighted adjacency matrix of the correlation coefficients, together with the associated p-values.

Usage

```
correlation(x, method = "pearson", p.adjust = "none", ...)
```

Arguments

`x`	‘matrix‘, where columns are the samples and the rows are features (metabolites), cell entries are intensity values
`method`	‘character‘, either "pearson", "spearman", "pearson_partial", "spearman_partial", or "ggm".
`p.adjust`	‘character‘, method of p-value adjustment passed to ‘p.adjust‘
`...`	additional arguments passed to ‘corr.test‘ or ‘partialCorrelation‘

Details

If ‘"pearson"‘ or ‘"spearman"‘ is used as a ‘method‘, the function ‘corr.test‘ from ‘psych‘ will be employed.

If ‘"ggm"‘ is used as a ‘method‘, the function ‘ggm.estimate.pcor‘ from ‘GeneNet‘ will be employed.

If ‘"pearson_partial"‘ or ‘"spearman_partial"‘ is used as a ‘method‘ the function ‘partialCorrelation‘ will be employed.

‘method‘ will be passed to argument ‘method‘ in ‘corr.test‘ (in the case of ‘"pearson"‘ or ‘"spearman"‘) or to ‘method‘ in ‘partialCorrelation‘ (‘"pearson"‘ and ‘"spearman"‘ for ‘"pearson_partial"‘ and ‘"spearman_partial"‘, respectively).

Value

‘list‘ containing two matrices, the first matrix contains correlation coefficients and the second matrix contains the corresponding p-values as obtained from the correlation algorithms ‘corr.test‘ or ‘partialCorrelation‘ (depending on the chosen ‘method‘) and optionally the adjusted p.values (argument ‘p.adjust‘)

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>, Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
correlation(x, method = "pearson")
```

---

getLinks *Write an adjacency matrix to a 'data.frame'*

Description

'getLinks' vectorizes a numerical square 'matrix' and writes the values and their corresponding ranks to a 'data.frame'.

Usage

```
getLinks(mat, exclude = "== 1", decreasing = TRUE)
```

Arguments

mat	matrix containing the values of confidence for a link
exclude	'character', logical statement as 'character' to set 'TRUE' values to NaN in 'mat', will be omitted if 'exclude = NULL'
decreasing	'logical', if 'TRUE', the highest confidence value will get the first rank, if 'FALSE', the lowest confidence value will get the first rank

Details

'getLinks' is a helper function used in the function 'threshold'.

Value

'data.frame' with entries 'row', 'col', 'confidence' and 'rank'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
mat <- matrix(0:8, ncol = 3, nrow = 3)
MetNet:::getLinks(mat, exclude = "== 0", decreasing = TRUE)
```

lasso *Create an adjacency matrix based on LASSO*

Description

'lasso' infers a adjacency matrix using LASSO using the 'stabsel.matrix' function from the 'stabs' package. 'lasso' extracts the predictors from the function 'stabsel.matrix' and writes the coefficients to an adjacency matrix.

Usage

```
lasso(x, parallel = FALSE, ...)
```

Arguments

x	matrix, where columns are the samples and the rows are features (metabolites), cell entries are intensity values
parallel	logical, should computation be parallelized? If 'parallel = TRUE' the 'bplapply' will be applied if 'parallel = FALSE' the 'lapply' function will be applied.
...	parameters passed to 'stabsel.matrix'

Details

For use of the parameters used in the 'stabsel.matrix' function, refer to '?stabs::stabsel.matrix'.

Value

matrix, matrix with edges inferred from LASSO algorithm 'stabsel.matrix'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
x_z <- t(apply(x, 1, function(y) (y - mean(y)) / sd(y)))
## Not run: lasso(x = x_z, PFER = 0.95, cutoff = 0.95)
```

mat_test *Example data for* MetNet*: unit tests*

Description

mat_test contains 7 toy features that were derived from rnorm. It will be used as an example data set in unit tests.

Format

matrix

Value

matrix

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Source

set.seed(1) random_numbers <- rnorm(140, mean = 10, sd = 2) mat_test <- matrix(random_numbers, nrow = 7) mat_test[1:3,] <- t(apply(mat_test[1:3,], 1, sort)) mat_test[5:7,] <- t(apply(mat_test[5:7,], 1, sort, decreasing = TRUE)) rownames(mat_test) <- paste("x", 1:7, sep = "")

mat_test_z *Example data for* MetNet*: unit tests*

Description

mat_test_z contains 7 toy features that were derived from rnorm. It will be used as an example data set in unit tests.

Format

matrix

Value

matrix

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Source

set.seed(1) random_numbers <- rnorm(140, mean = 10, sd = 2) mat_test <- matrix(random_numbers, nrow = 7) mat_test[1:3,] <- t(apply(mat_test[1:3,], 1, sort)) mat_test[5:7,] <- t(apply(mat_test[5:7,], 1, sort, decreasing = TRUE)) rownames(mat_test) <- paste("x", 1:7, sep = "") mat_test_z <- apply(mat_test, 1, function(x) (x - mean(x, na.rm = TRUE))/sd(x, na.rm = TRUE))

`ms2_test` *Spectra data to test addSpectralSimilarity*

Description

`ms2_test` contains a subset of a `Spectra` object. It will be used as an example annotation in the vignette to show the functionality of the package.

Format

`Spectra`

Value

`Spectra`

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

`mz_summary` *Create a summary from adjacency list containing mass differences*

Description

The function ‘mz_summary‘ creates a summary from the ‘AdjacencyMatrix‘, containing mass differences. Individual mass differences are counted over all features. The input may be an ‘AdjacencyMatrix‘ object originating from the function ‘structural‘, or ‘combine‘. The parameter ‘filter‘ will define if data will be filtered above a certain threshold or not.

Usage

```
mz_summary(am, var = c("group", "formula"), filter = 0)
```

Arguments

am	'AdjacencyMatrix', a formal class of 'AdjacencyMatrix' containing the mass differences, that have previously been generated by the function 'structural'or 'combine'
var	'character' vector corresponding to 'assayNames(am)', the counts will be grouped according to 'var'
filter	'numeric', leave empty or set to '0' if unfiltered data are required. Select a 'numeric' as a threshold on counts of mz differences. May be useful to visualize big data.

Details

Summarizes the adjacency matrices containing mass difference values, i.e. either adjacency list from 'structural'or 'combine' may be used. The default is filter = F, so the unfiltered summary will be returned. If filter is set to a 'number', e.g. 1000 only mz differences above this threshold will be displayed.

The function can be applied for adjacency lists from 'structural' and 'combine'.

Value

'data.frame' containing the numbers of present mz differences and corresponding name.

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de> and Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
transformation <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
transformation <- data.frame(group = transformation[, 1],
                             formula = transformation[, 2],
                             mass = as.numeric(transformation[, 3]))
am_struct <- structural(x_test, transformation, ppm = 5,
    var = c("group", "mass", "formula"), directed = TRUE)
# unfiltered mz difference counts
mz_summary(am_struct)
# filtered mz difference counts
mz_summary(am_struct, filter = 2)
```

mz_vis *Visualize mass difference distribution*

Description

The function ‘mz_vis‘ visualizes the mass difference distribution, which has been summarized by ‘mz_summary‘.

Usage

```
mz_vis(df, var = "group")
```

Arguments

df	‘data.frame‘, previously generated by ‘mz_summary‘. Needs to contain the columns "transformation", "mass_difference" and "counts".
var	‘character(1)‘, the column in ‘df‘ to visualize on the y-axis

Details

Plots the mass difference distribution, summarized by ‘mz_summary‘. Visualization is performed using ggplot2

Value

‘ggplot‘ object and corresponding barplot for visualizations

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de> and Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
transformation <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
transformation <- data.frame(group = transformation[, 1],
                                formula = transformation[, 2],
                                mass = as.numeric(transformation[, 3]))
am_struct <- structural(x_test, transformation,
    var = c("group", "formula", "mass"), ppm = 5, directed = TRUE)
mz_sum <- mz_summary(am_struct, var = "group")
mz_vis(mz_sum)
```

`partialCorrelation` *Calculate the partial correlation and p-values*

Description

'partialCorrelation' infers an adjacency matrix of partial correlation values and associated p-values using using the 'cor2pcor' function (from the 'corpcor' package). 'partialCorrelation' calculates the p-values from the number of samples ('n') and the number of controlling variables ('g'). The function will return a list containing the weighted adjacency matrix of the correlation values, together with the associated p-values.

Usage

```
partialCorrelation(x, method = "pearson", ...)
```

Arguments

`x`	'matrix', where columns are the features (metabolites) and the rows are samples, cell entries are intensity values
`method`	'character', either "pearson", "spearman"
`...`	further arguments passed to 'cor' from 'base' or 'cor2pcor' from 'corpcor'

Details

The correlation coefficients $r_{ij|S}$ are obtained from 'cor2pcor' ('corpcor' package).

The t-values are calculated via

$t_{ij|S} = r_{ij|S} \cdot \sqrt{\frac{n-2-g}{1-r^2_{ij|S}}}$, where n are the number of samples and g the number of controlling variables (number of features - 2).

The p-values are calculated as follows $p_{ij|S} = 2 \cdot pt(-abs(t_{ij|S}), df = n - 2 - g)$

Value

'list' containing two matrices, the first matrix contains correlation coefficients and the second matrix contains the corresponding p-values

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[, 3:ncol(x_test)]
x <- as.matrix(x)
x <- t(x)
partialCorrelation(x, use = "pairwise", method = "pearson")
```

peaklist *Example data for* MetNet*: data input*

Description

The object peaklist is a data.frame, where rows are features and the columns are samples (starting with X001-180).

Format

data.frame

Value

data.frame

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Source

Internal peaklist from metabolite profiling of Nicotiana species after W+OS and MeJA treatment. The data was processed by xcms and CAMERA scripts. All unncessary information is removed, keeping only the columns "mz", "rt" and the respective columns containing the intensity values. All row entries with retention time < 103 s and > 440 s were removed. Entries with m/z values < 250 and > 1200 were removed as well as entries with m/z values between 510 and 600 to reduce the file size.

randomForest *Create an adjacency matrix based on random forest*

Description

'randomForest' infers an adjacency matrix using random forest using the 'GENIE3' function from the 'GENIE3' package. 'randomForest' returns the importance of the link between features in the form of an adjacency matrix.

Usage

```
randomForest(x, ...)
```

Arguments

x	matrix, where columns are the samples and the rows are features (metabolites), cell entries are intensity values
...	parameters passed to 'GENIE3'

Details

For use of the parameters used in the 'GENIE3' function, refer to '?GENIE3::GENIE3'. The arguments 'regulators' and 'targets' are set to 'NULL'. Element $w_{i,j}$ (row i, column j) gives the importance of the link from i to j.

Value

matrix, matrix with the importance of the links inferred from random forest algorithm implemented by 'GENIE3'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
randomForest(x)
```

---

rtCorrection *Correct connections in the structural adjacency matrices by retention time*

Description

The function 'rtCorrection' corrects the adjacency matrix infered from structural data based on shifts in the retention time. For known chemical modifications (e.g. addition of glycosyl groups) molecules with the moiety should elute at a different time (in the case of glycosyl groups the metabolite should elute earlier in a reverse-phase liquid chromatography system). If the connection for the metabolite does not fit the expected behaviour, the connection will be removed (otherwise sustained).

Usage

```
rtCorrection(am, x, transformation, var = "group")
```

Arguments

am	'AdjacencyMatrix' object returned by the function 'structural'. The object contains the assays '"binary"' and additional assays with 'character' matrices (only the '"binary"' assay is required). The assay '"binary"' stores the 'numeric' matrix with edges inferred by mass differences.

x	‘matrix‘, where columns are the samples and the rows are features (metabolites), cell entries are intensity values, ‘x‘ contains the columns "‘mz‘" and ‘"rt"‘ that has the m/z and rt information (numerical values) for the correction of retention time shifts between features that have a putative connection assigned based on m/z value difference.
transformation	‘data.frame‘, containing the columns ‘var‘, and ‘"rt"‘ that will be used for correction of transformation of (functional) groups based on retention time shifts derived from ‘x‘
var	‘character(1)‘, the key that is used for matching between the column ‘var‘ in ‘transformation‘ and the assay ‘var‘ in ‘am‘

Details

‘rtCorrection‘ is used to correct the (unweighted) adjacency matrices returned by ‘structural‘ when information is available about the retention time and shifts when certain transformation occur (it is meant to filter out connections that were created by m/z differences that have by chance the same m/z difference but different/unexpected retention time behaviour).

‘rtCorrection‘ accesses the assay ‘transformation‘ of ‘am‘ and matches the elements in the ‘var‘ column against the character matrix. In case of matches, ‘rtCorrection‘ accesses the ‘"mz"‘ and ‘"rt"‘ columns of ‘x‘ and calculates the retention time difference between the features. ‘rtCorrection‘ then checks if the observed retention time difference matches the expected behaviour (indicated by ‘"+"‘ for a higher retention time of the feature with the putative group, ‘"-"‘ for a lower retention time of the feature with the putative group or ‘"?"‘ when there is no information available or features with that group should not be checked).

In case several transformation were assigned to a feature/feature pair, the connection will be removed if there is an inconsistency with any of the given transformations.

Value

‘AdjacencyMatrix‘ containing the slots ‘binary‘ and additional ‘character‘ adjacency matrices. The slot ‘directed‘ is inherited from ‘am‘.

The assay ‘binary‘ stores the ‘numeric‘ ‘matrix‘ with edges inferred mass differences corrected by retention time shifts.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
rownames(x_test) <- paste(round(x_test[, "mz"], 2),
    round(x_test[, "rt"]), sep = "_")
transformation <- rbind(
    c("Hydroxylation (-H)", "O", 15.9949146221, "+"),
    c("Malonyl group (-H2O)", "C3H2O3", 86.0003939305, "+"),
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315", "-"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851", "-"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945", "-"))
```

```
transformation <- data.frame(group = transformation[, 1],
         formula = transformation[, 2],
         mass = as.numeric(transformation[, 3]),
         rt = transformation[, 4])
am_struct <- structural(x = x_test, transformation = transformation,
    var = c("group", "mass"), ppm = 10, directed = FALSE)
am_struct_rt <- rtCorrection(am = am_struct, x = x_test,
     transformation = transformation)

## visualize the adjacency matrices
library(igraph)
g <- graph_from_adjacency_matrix(assay(am_struct, "binary"),
    mode = "undirected")
g_rt <- graph_from_adjacency_matrix(assay(am_struct_rt, "binary"),
    mode = "undirected")

plot(g, edge.width = 2, edge.arrow.size = 0.5, vertex.label.cex = 0.7)
plot(g_rt, edge.width = 2, edge.arrow.size = 0.5, vertex.label.cex = 0.7)
```

`spectra_matrix` *Spectra data to test addSpectralSimilarity*

Description

`spectra_matrix` contains one selected putative annotation of `x_test`. Missing annotations are filled with 'NA''s. It will be used as an example annotation in the vignette to show the functionality of the package.

Format

`matrix`

Value

`matrix`

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

Source

library(MsCoreUtils) library(Spectra)

f <- system.file("spectra_matrix/ms2_test.RDS", package = "MetNet") sps_sub <- readRDS(f)

adj_spec <- Spectra::compareSpectra(sps_sub, FUN = ndotproduct) colnames(adj_spec) <- sps_sub$id rownames(adj_spec) <- sps_sub$id

`statistical` *Create an 'AdjacencyMatrix' object containing assays of adjacency matrices from statistical methods*

Description

The function 'statitical' infers adjacency matrix topologies from statistical methods and returns matrices of these networks in an 'AdjacencyMatrix' object. The function includes functionality to calculate adjacency matrices based on LASSO (L1 norm)-regression, random forests, context likelihood of relatedness (CLR), the algorithm for the reconstruction of accurate cellular networks (ARACNE), Pearson correlation (also partial), Spearman correlation (also partial) and score-based structure learning (Bayes). The function returns an 'AdjacencyMatrix' object of adjacency matrices that are defined by 'model'.

Usage

```
statistical(x, model, ...)
```

Arguments

`x`	'matrix' that contains intensity values of features/metabolites (rows) per sample (columns).
`model`	'character' vector containing the methods that will be used ('"lasso"', '"randomForest"', '"clr"', '"aracne"', '"pearson"', '"pearson_partial"', '"spearman"', '"spearman_partial"', 'ggm', '"bayes"')
`...`	parameters passed to the functions 'lasso', 'randomForest', 'clr', 'aracne', 'correlation' and/or 'bayes'

Details

The function 'statistical' includes functionality to calculate adjacency matrices based on LASSO (L1 norm)-regression, random forests, context likelihood of relatedness (CLR), the algorithm for the reconstruction of accurate cellular networks (ARACNE), Pearson correlation (also partial), Spearman correlation (also partial) and Constraint-based structure learning (Bayes).

'statistical' calls the function 'lasso', 'randomForest', 'clr', 'aracne', 'correlation' (for '"pearson"', '"pearson_partial"', '"spearman"', '"spearman_partial"', '"ggm"') and/or 'bayes' as specified by 'model'. It will create adjacency matrices using the specified methods and will return an 'AdjacencyMatrix' containing the weighted adjacency matrices in the 'assays' slot.

Internally 'x' will be z-scaled and the z-scaled object will be used in 'lasso', 'clr' and/or 'aracne'.

The slot 'type' is set to 'statistical'. The slot 'directed' is set to 'TRUE' if the methods '"lasso"', '"randomForest"', or '"bayes"' were used, otherwise 'directed' is set to 'FALSE'. The slot 'threshold' is set to 'FALSE'.

Value

'AdjacencyMatrix' containing the respective adjacency matrices in the 'assay' slot as specified by 'model'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
statistical(x = x, model = c("pearson", "spearman"))
statistical(x = x, model = c("pearson", "spearman"), p.adjust = "BH")
```

---

structural *Create adjacency matrix based on m/z (molecular weight) difference*

Description

The function ‘structural‘ infers an unweighted adjacency matrix using differences in m/z values that are matched against a ‘data.frame‘ (‘transformation‘ of calculated theoretical differences of loss/addition of functional groups. ‘structural‘ returns an ‘AdjacencyMatrix‘ object containing the unweighted ‘numeric‘ ‘matrix‘ (assay ‘binary‘), together with one or multiple ‘character‘ matrices containing e.g. the type of loss/addition or the mass differences. The creation of the additional ‘character‘ matrices is controlled by the ‘var‘ argument that specifies the column in ‘transformation‘ as the data source for the adjacency matrices.

Usage

```
structural(x, transformation, var = character(), ppm = 5, directed = FALSE)
```

Arguments

x	‘matrix‘ or ‘data.frame‘, where columns are the samples and the rows are features (metabolites), cell entries are intensity values. ‘x‘ contains the column ‘"mz"‘ that has the m/z information (numerical values) for the calculation of mass differences between features
transformation	‘data.frame‘, containing the columns ‘"group"‘, and ‘"mass"‘ that will be used for detection of transformation of (functional) groups
var	‘character‘ corresponding to column names in ‘transformation‘
ppm	‘numeric(1)‘, mass accuracy of m/z features in parts per million (ppm)
directed	‘logical(1)‘, if ‘TRUE‘, absolute values of m/z differences will be taken to query against ‘transformation‘ (irrespective the sign of ‘mass‘) and undirected adjacency matrices will be returned as the respective assays. This means, if there is a negative mass in ‘transformation[, "mass"]‘, this negative mass will not be reported. If ‘FALSE‘, directed adjacency matrices will be returned with links reported that match the transformations defined in ‘transformation‘ (respecting the sign of ‘mass‘). The ‘directed‘ slot of the returned ‘AdjacencyMatrix‘ object will contain the information on ‘directed‘.

Details

‘structural‘ accesses the column ‘"mz"‘ of ‘x‘ to infer structural topologies based on the functional groups defined by ‘transformation‘. To account for the mass accuracy of the dataset ‘x‘, the user can specify the accuracy of m/z features in parts per million (ppm) by the ‘ppm‘ argument. The m/z values in the ‘"mz"‘ column of ‘x‘" will be converted to m/z ranges according to the ‘ppm‘ argument (default ‘ppm = 5‘).

The returned ‘AdjacencyMatrix‘ object contains the assays ‘binary‘ and additional adjacency matrices depending on the ‘var‘ parameter. The assay ‘binary‘ stores the ‘numeric‘ ‘matrix‘ with binary edges inferred from mass differences. The ‘var‘ parameter has to be set according to the column names in ‘transformation‘. E.g. if the ‘transformation‘ object contains a ‘"group"‘ column that stores the name of the transformation, setting ‘var = "group"‘ will create an assay ‘"group"‘ that contains the adjacency matrices where the entries correspond to the ‘"group"‘ information for the respective feature pairs.

The ‘type‘ slot is set to ‘structural‘. The ‘directed‘ slot is set accordingly to the ‘directed‘ argument of the function ‘structural‘. The ‘thresholded‘ slot is set to ‘FALSE‘.

Value

‘AdjacencyMatrix‘ object. The object will store the adjacency matrix/matrices in the assay slot/slots. The numerical (unweighted) adjacency matrix inferred from mass differences is stored as the assay ‘"binary"‘. Depending on the ‘var‘ argument, there are additional adjacency matrices stored in the assay slot.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com> and Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

Examples

```
data("x_test", package = "MetNet")
transformation <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"))
transformation <- data.frame(group = transformation[, 1],
                             formula = transformation[, 2],
                             mass = as.numeric(transformation[, 3]))
am_struct <- structural(x_test, transformation, var = c("group", "mass"),
    ppm = 10, directed = TRUE)
```

`threshold` *Threshold the statistical adjacency or spectral similarity matrices*

Description

The function ‘threshold‘ takes as input an ‘AdjacencyMatrix‘ object containing adjacency matrices as returned from the function ‘statistical‘ OR the ‘AdjacencyMatrix‘ object of the type "structural" containing spectral similarity adjacency matrices, that were added by ‘addSpectSimil()‘.Depending on the ‘type‘ argument, ‘threshold‘ will identify the strongest link that are lower or higher a certain threshold (‘type = "threshold"‘) or identify the top ‘n‘ links (‘type‘ either ‘"top1"‘, ‘"top2"‘ or ‘"mean"‘). It will return this kind of information as a binary matrix in the form of an ‘AdjacencyMatrix‘ object.

Usage

```
threshold(
  am,
  type = c("threshold", "top1", "top2", "mean"),
  args,
  values = c("all", "min", "max"),
  na.rm = TRUE
)
```

Arguments

am	‘AdjacencyMatrix‘ object of ‘type‘ ‘"statistical"‘ as created from the function ‘statistical‘ OR ‘AdjacencyMatrix‘ object of the type "structural" containing spectral similarity adjacency matrices, that were added by ‘addSpectSimil()‘. The object will contain the adjacency matrices in the ‘assay‘ slot.
type	‘character‘, either ‘"threshold"‘, ‘"top1"‘, ‘"top2"‘ or ‘"mean"‘
args	‘list‘. Depending on the ‘type‘ arguments the list element will be different. In the case of ‘type == "threshold"‘, ‘args‘ has the entry ‘filter‘ (‘character‘ of length 1). The character vector will specify the kind of filtering applied to the adjacency matrices. Elements in ‘filter‘ will refer to the ‘assayNames‘, e.g. ‘list(filter = "pearson_coef > 0.8")‘ will retain all edges with Pearson correlation coefficients > 0.8. ‘list(filter = "pearson_coef > 0.8 & spearman_coef > 0.5")‘ will retain all edges with Pearson correlation coefficients > 0.8 AND Spearman correlation coefficients > 0.5. ‘list(filter = "abs(pearson_coef) > 0.8 & spearman_coef > 0.5")‘ will retain all edges with Pearson correlation coefficients > 0.8 or < -0.8 AND Spearman correlation coefficients > 0.5. In the case of ‘type == "top1"‘, ‘type == "top2"‘, or ‘type == "mean"‘, ‘args‘ has the entry ‘n‘ (‘numeric‘ of length 1), that denotes the number of top ranks written to the consensus matrix. Optionally, ‘args‘ has the entry ‘abs‘ which will take absolute values of the coefficients (will default to ‘FALSE‘ if ‘args$abs‘ is not specified).
values	‘character‘, take from the adjacency matrix all values ("all"), the minimum of the pairs ("min"), $a^*_{ij} = min(a_{ij}, a_{ji})$, or the maximum ("max"), $a^*_{ij} = max(a_{ij}, a_{ji})$
na.rm	‘logical‘, if set to ‘TRUE‘, the ‘NA‘s in the assay slots will not be taken into account when creating the ‘"consensus"‘ assay. If set to ‘FALSE‘, the ‘NA‘s will be taken into account and might be passed to the ‘"consensus"‘ assay (or

‘"binary"‘ if input was type "structural"). If ‘FALSE‘ the user can set the filter e.g. to ‘(ggm_coef > 0 | is.na(ggm_coef))‘, when there are ‘NA‘s in ‘ggm_coef‘ to disregard ‘NA‘s.

Details

‘values‘ has to be set carefully depending on if the ‘AdjacencyMatrix‘ object ‘am‘ is ‘directed‘ or not.

In the case of ‘type == "threshold"‘, ‘args‘ has the entry ‘filter‘ (‘character‘ of length 1). The character vector will specify the kind of filtering applied to the adjacency matrices. Elements in ‘filter‘ will refer to the ‘assayNames‘, e.g. ‘list(filter = "pearson_coef > 0.8")‘ will retain all edges with Pearson correlation coefficients > 0.8. ‘list(filter = "pearson_coef > 0.8 & spearman_coef > 0.5")‘ will retain all edges with Pearson correlation coefficients > 0.8 AND Spearman correlation coefficients > 0.5. ‘list(filter = "abs(pearson_coef) > 0.8 & spearman_coef > 0.5")‘ will retain all edges with Pearson correlation coefficients > 0.8 and < -0.8.

If ‘type‘ is equal to ‘"top1"‘, ‘"top2"‘ or ‘"mean"‘, then ‘args‘ has to contain a numeric vector of length 1 that gives the number of top ranks included in the returned adjacency matrix. In this case values that are 0 for the models ‘lasso‘, ‘randomForest‘ and ‘bayes‘ are set to ‘NaN‘; values from correlation (Pearson and Spearman, including for partial correlation) and ‘clr‘ and ‘aracne‘ are taken as they are.

For ‘type = "top1"‘, the best (i.e. lowest) rank in ‘am‘ is taken. For ‘type = "top2"‘, the second best (i.e. second lowest) rank in ‘am‘ is taken. For ‘type = "mean"‘, the average rank in ‘am‘ is taken. Subsequently the first ‘n‘ unique ranks are returned.

Value

‘AdjacencyMatrix‘ object containing a binary adjacency matrix given the links supported by the ‘type‘ and the ‘args‘ (in the slot ‘"consensus"‘ if the input was type "statistical" or in the slot ‘"binary"‘ if it was type "structural". The object will furthermore contain the supplied data input, i.e. all assays from ‘am‘. The slot ‘threshold‘ is set to ‘TRUE‘.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x <- x_test[1:10, 3:ncol(x_test)]
x <- as.matrix(x)
model <- c("pearson", "spearman")
args <- list()
am_stat <- statistical(x, model = model)

## type = "threshold"
args <- list(filter = "pearson_coef > 0.95 & spearman_coef > 0.95")
threshold(am = am_stat, type = "threshold", args = args)

## type = "top1"
```

```
args <- list(n = 10)
threshold(am = am_stat, type = "top1", args = args)

## type = "top2"
threshold(am = am_stat, type = "top2", args = args)

## type = "mean"
threshold(am = am_stat, type = "mean", args = args)
```

---

topKnet *Return consensus ranks from a matrix containing ranks*

Description

'topKnet' returns consensus ranks depending on the 'type' argument from 'ranks', a matrix containing the ranks per statistical 'model'.

Usage

```
topKnet(ranks, type, na.rm = TRUE)
```

Arguments

ranks	'matrix' containing the ranks per statistical model (in columns) and per feature pair (in rows)
type	'character', either '"top1"', '"top2"' or '"mean"'
na.rm	'logical', if set to 'TRUE', the 'NA's in the assay slots will not be taken into account when creating the '"top1"', '"top2"' or '"mean"' of ranks. If set to 'FALSE', the 'NA's will be taken into account when creating the '"top1"', '"top2"' or '"mean"' ranks. If 'FALSE' the resulting aggregations will be 'NA' if an 'NA' is present in the coeffients of one feature pair.

Details

See Hase et al. (2014) for further details.

Value

'numeric' 'vector" with consensus ranks

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

References

Hase et al. (2014): Harnessing Diversity towards the Reconstructing of Large Scale Gene Regulatory Networks. PLoS Computational Biology, 2013, e1003361, doi: [10.1371/journal.pcbi.1003361](https://journals.plos.org

Examples

```
## na.rm == TRUE
ranks <- matrix(c(c(1, 2, 3), c(2, 1, 3)), ncol = 2)

## type = "top1"
MetNet:::topKnet(ranks = ranks, type = "top1", na.rm = TRUE)

## type = "top2"
MetNet:::topKnet(ranks = ranks, type = "top2", na.rm = TRUE)

## type = "mean"
MetNet:::topKnet(ranks = ranks, type = "mean", na.rm = TRUE)

## na.rm == FALSE
ranks <- matrix(c(c(1, 2, 3), c(2, 1, 3)), ncol = 2)

## type = "top1"
MetNet:::topKnet(ranks = ranks, type = "top1", na.rm = FALSE)

## type = "top2"
MetNet:::topKnet(ranks = ranks, type = "top2", na.rm = FALSE)

## type = "mean"
MetNet:::topKnet(ranks = ranks, type = "mean", na.rm = FALSE)

## na.rm == FALSE
ranks <- matrix(c(c(1, 2, NA), c(2, 1, 3)), ncol = 2)

## type = "top1"
MetNet:::topKnet(ranks = ranks, type = "top1", na.rm = FALSE)

## type = "top2"
MetNet:::topKnet(ranks = ranks, type = "top2", na.rm = FALSE)

## type = "mean"
MetNet:::topKnet(ranks = ranks, type = "mean", na.rm = FALSE)
```

---

x_annotation *Example annotation for* MetNet*: data input*

Description

x_annotation contains one selected putative annotation of x_test. Missing annotations are filled with 'NA''s. It will be used as an example annotation in the vignette to show the functionality of the packages.

Format

matrix

Value

matrix

Author(s)

Liesa Salzer, <liesa.salzer@helmholtz-muenchen.de>

Source

data("x_test", package = "MetNet")

x_annotation <- x_test[,1:2]

x_annotation <- cbind(x_annotation,"database_mz" = NA, "database_identifier" = NA, "chemical_formula" = NA, "smiles" = NA, "inchi" = NA, "inchikey" = NA, "metabolite_identification" = NA, "fragmentations" = NA, "modifications" = NA, "charge" = NA, "database" = NA)

x1856 <- cbind(x_annotation["x1856", "mz"], x_annotation["x1856", "rt"], "database_mz" = 308.2, "database_identifier" = "N-caffeoylspermidine", "chemical_formula" = "C16H25N3O3", "smiles" = "C=1(C=C(C(=CC1)O)O)/C=C/C(NCCCNCCCCN)=O", "inchi" = "InChI=1S/C16H25N3O3/c17-8-1-2-9-18-10-3-11-19-16(22)7-5-13-4-6-14(20)15(21)12-13/h4-7,12,18,20-21H,1-3,8-11,17H2,(H,19,22)/b7-5+", "inchikey" = "AZSUJBAOTYNFDE-FNORWQNLSA-N", "metabolite_identification" = NA, "fragmentations" = NA, "modifications" = NA, "charge" = 1, "database" = NA)

x_annotation[rownames(x_annotation) == "x1856",] <- x1856 x_annotation <- x_annotation[,-c(1:2)]

---

x_test *Example data for* MetNet*: data input*

Description

x_test contains 36 selected metabolic features of peaklist. It will be used as an example data set in the vignette to show the functionality of the packages.

Format

matrix

Value

matrix

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Source

data("peaklist_example", package = "MetNet") peaklist[, 3:dim(peaklist)[2]] <- apply(peaklist[, 3:dim(peaklist)[2]], 2, function(x) x / quantile(x, 0.75)) peaklist[, 3:dim(peaklist)[2]] <- log2(peaklist[, 3:dim(peaklist)[2]] + 1)

function to add specific features of x (defined by m/z and retention ## time) to x_test addTo_x_test <- function(x_test, x, mz, rt) mzX <- x[, "mz"] rtX <- x[, "rt"] new <- x[mzX>(mz-0.01) & mzX<(mz+0.01) & rtX>(rt-0.01) & rtX<(rt+0.01),] x_test <- rbind(x_test, new) return(x_test)

Nicotianoside IX M+Na+ 739.3515 rt 426.1241 x_test <- peaklist[peaklist[, "mz"] > 739.35 & peaklist[, "mz"] < 739.36 & peaklist[, "rt"] > 426.18 & peaklist[, "rt"] < 426.2,] ## Lyciumoside I M+Na+ 653.3497 x_test <- addTo_x_test(x_test, peaklist, mz = 653.3497, rt = 417.46) ## LyciumosideII M+Na+ 815.4043 x_test <- addTo_x_test(x_test, peaklist, mz = 815.40, rt = 383.60) ## Nicotianoside X M+Na+ 825.3503 x_test <- addTo_x_test(x_test, peaklist, mz = 825.35, rt = 434.38) ## Nicotianoside XI M+Na+ 901.39913 x_test <- addTo_x_test(x_test, peaklist, mz = 901.40, rt = 391.15) ## NicotianosideXII M+Na+ 987.4037 x_test <- addTo_x_test(x_test, peaklist, mz = 987.40, rt = 398.46) ## NicotianosideXIII M+Na+ 1074.4042 x_test <- addTo_x_test(x_test, peaklist, mz = 1074.40, rt = 404.92) ## Lyciumoside IV M+Na+ 799.4091 x_test <- addTo_x_test(x_test, peaklist, mz = 799.40, rt = 411.23) ## Nicotianoside I M+Na+ 885.4084 x_test <- addTo_x_test(x_test, peaklist, mz = 885.41, rt = 420.12) ## Nicotianoside II M+Na+ 971.4074 x_test <- addTo_x_test(x_test, peaklist, mz = 971.41, rt = 428.81) ## Nicotianoside III M+Na+ 945.4653 x_test <- addTo_x_test(x_test, peaklist, mz = 945.46, rt = 402.75) ## Nicotianoside IV M+Na+ 1031.4645 x_test <- addTo_x_test(x_test, peaklist, mz = 1031.46, rt = 412.40) ## Nicotianoside V M+Na+ 1117.4681 x_test <- addTo_x_test(x_test, peaklist, mz = 1117.46, rt = 422.19) ## Attenoside (or DTG956) M+Na+ 961.4601 x_test <- addTo_x_test(x_test, peaklist, mz = 961.46, rt = 380.46) ## DTG1042/Nicotianoside VI M+Na+ 1047.4525 x_test <- addTo_x_test(x_test, peaklist, mz = 1047.46, rt = 387.28) ## Nicotianoside-VII M+Na+ 1133.4624 x_test <- addTo_x_test(x_test, peaklist, mz = 1133.46, rt = 394.70) ## NicotianosideVIII M+Na+ 1219.4619 x_test <- addTo_x_test(x_test, peaklist, mz = 1219.46, rt = 400.99) ## N-coumaroylputrescine [M+H+]+ 235.143 x_test <- addTo_x_test(x_test, peaklist, mz = 235.14, rt = 193.85) ## N',N''-coumaroyl,caffeoylspermidine [M+H+]+ 454.23 x_test <- addTo_x_test(x_test, peaklist, mz = 454.23, rt = 264.43) ## N-caffeoylputrescine isomer 1 [M+H+]+ 251.14 x_test <- addTo_x_test(x_test, peaklist, mz = 251.14, rt = 108.34) ## N-caffeoylputrescine isomer 2 [M+H+]+ 251.14 x_test <- addTo_x_test(x_test, peaklist, mz = 251.14, rt = 143.11) ## N-caffeoylspermidine [M+H+]+ 308.2 x_test <- addTo_x_test(x_test, peaklist, mz = 308.2, rt = 246.71) ## N-feruloylputrescine [M+H+]+ 265.153 x_test <- addTo_x_test(x_test, peaklist, mz = 265.15, rt = 191.55) ## N-feruloyl-spermidine iso1 [M+H+]+ 322.212 x_test <- addTo_x_test(x_test, peaklist, mz = 322.21, rt = 104.13) ## N-feruloyl-spermidine iso2 [M+H+]+ 322.212 x_test <- addTo_x_test(x_test, peaklist, mz = 322.21, rt = 147.98) ## N'-N''-dicaffeoyl -spermidine [M+H+]+ 470.23 x_test <- addTo_x_test(x_test, peaklist, mz = 470.23, rt = 247.15) ## N'-N''-diferuloyl-spermidine/ ##N#,N$-Coumaroyl,sinapoyl spermidine isomer [M+H+]+ 498.260/498.261 x_test <- addTo_x_test(x_test, peaklist, mz = 498.26, rt = 289.05) ## N'-N''-dihydrated-diferuloyl-spermidine [M+H+]+ 502.25 x_test <- addTo_x_test(x_test, peaklist, mz = 502.25, rt = 242.55) ## unknown conjugate [M+H+]+ 411.2012 x_test <- addTo_x_test(x_test, peaklist, mz = 411.20, rt = 211.67) ## N'-N''-caffeoyl,feruloyl spermidine iso1 [M+H+]+ 484.245 x_test <- addTo_x_test(x_test, peaklist, mz = 484.24, rt = 264.44) ## N'-N''-caffeoyl,feruloyl spermidine iso2 [M+H+]+ 484.245 x_test <- addTo_x_test(x_test, peaklist, mz = 484.24, rt = 270.65) ## O -Coumaroylquinic acid isomer 1 [M+H+]+ 339.109 x_test <- addTo_x_test(x_test, peaklist, mz = 339.11, rt = 248.79) ## O -Coumaroylquinic acid isomer 1 [M+H+]+ 339.109 x_test <- addTo_x_test(x_test, peaklist, mz = 339.11, rt = 268.97) ## O-caffeoylquinic acid isomer 1 [M+H+]+ 355.1014 x_test <- ad-

dTo_x_test(x_test, peaklist, mz = 355.10, rt = 175.75) ## O-caffeoylquinic acid isomer 2 [M+H+]+ 355.1014 x_test <- addTo_x_test(x_test, peaklist, mz = 355.10, rt = 215.85) ## O-caffeoylquinic acid isomer 3 [M+H+]+ 355.1014 x_test <- addTo_x_test(x_test, peaklist, mz = 355.10, rt = 241.04)

change rownames (that it is accepted by formulas) rownames(x_test) <- paste0("x", rownames(x_test))

Index

∗ **mass spectrometry, metabolomics**
MetNet-package, 3
.AdjacencyMatrix, 4
.assays_have_identical_colnames_rownames, 4
.assays_have_identical_dimnames, 5

addSpectralSimilarity, 6
addToList, 7
AdjacencyMatrix, 8
AdjacencyMatrix-class, 9
AllGenerics, 11
aracne, 11
as.data.frame *(*AdjacencyMatrix-class*)*, 9
as.data.frame,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9

bayes, 12

clr, 13
combine, 14
correlation, 16

dim *(*AdjacencyMatrix-class*)*, 9
dim,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9
directed *(*AdjacencyMatrix-class*)*, 9
directed,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9

getLinks, 17

lasso, 18
length,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9

mat_test, 19
mat_test_z, 19
MetNet *(*MetNet-package*)*, 3
MetNet-package, 3
ms2_test, 20
mz_summary, 20
mz_vis, 22

partialCorrelation, 23
peaklist, 24

randomForest, 24
rtCorrection, 25

show *(*AdjacencyMatrix-class*)*, 9
show,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9
spectra_matrix, 27
sps_sub *(*ms2_test*)*, 20
statistical, 28
structural, 29

threshold, 30
thresholded *(*AdjacencyMatrix-class*)*, 9
thresholded,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9
topKnet, 33
type *(*AdjacencyMatrix-class*)*, 9
type,AdjacencyMatrix-method *(*AdjacencyMatrix-class*)*, 9

x_annotation, 34
x_test, 35