Package 'MetNet'

March 2, 2021

Type Package

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

Version 1.8.0

Date 2020-03-07

VignetteBuilder knitr

Depends R (>= 3.6)

Imports bnlearn (>= 4.3), BiocParallel (>= 1.12.0), GENIE3 (>= 1.7.0), methods (>= 3.5), mpmi (>= 0.42), parmigene (>= 1.0.2), ppcor (>= 1.1), sna (>= 2.4), stabs (>= 0.6), stats (>= 3.6)

Suggests BiocGenerics (>= 0.24.0), BiocStyle (>= 2.6.1), glmnet (>= 2.0-18), igraph (>= 1.1.2), knitr (>= 1.11), rmarkdown (>= 1.15), testthat (>= 2.2.1)

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 matrices are combined to form a adjacency matrix inferred from both quantitative and structure information.

License GPL (>= 3)

RoxygenNote 6.1.1

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

git_branch RELEASE_3_12

git_last_commit d0b799b

git_last_commit_date 2020-10-27

Date/Publication 2021-03-01

Author Thomas Naake [aut, cre]

Maintainer Thomas Naake <thomasnaake@googlemail.com>

R topics documented:

`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: NA Maintainer: NA

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)
```

`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, type = "pearson")
cor_spearman <- correlation(x, type = "spearman")
l <- list(pearson = cor_pearson)
MetNet:::addToList(l, "spearman", cor_spearman)
```

---

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 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 or Jackknife bias corrected MI estimates) from the 'cmi' function of the 'mpmi' package can be used.
eps	numeric, used to remove the weakest edge of each triple of nodes

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 <- t(apply(x, 1, function(y) (y - mean(y)) / sd(y)))
mi_x_z <- mpmi::cmi(x_z)$bcmi
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 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 or Jackknife bias corrected MI estimates) from the 'cmi' function of the 'mpmi' package can be used.

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 <- t(apply(x, 1, function(y) (y - mean(y)) / sd(y)))
mi_x_z <- mpmi::cmi(x_z)$bcmi
clr(mi_x_z)
```

`combine` *Combine structural and statistical adjacency matrix*

Description

The function ‘combine‘ takes as input the structural and statistical adjacency matrix, created in former steps, adds them together and will report a connection between metabolites in the returned when the sum exceeds the ‘threshold‘. `combine` returns this consensus matrix supported by the structural and statistical adjacency matrices.

Usage

```
combine(structural, statistical, threshold = 1)
```

Arguments

`structural`	list containing ‘numeric‘ structural adjacency matrix in the first entry and ‘character‘ structural adjanceny matrix in the second entry
`statistical`	matrix containing ‘numeric‘ statistical adjacency matrix
`threshold`	numeric, threshold value to be applied to define a connection as present

Details

The matrices will be added and a unweighted connection will be reported when the value exceeds a certain value.

Value

‘list‘, in the first entry ‘matrix‘ of type ‘numeric‘containing the consensus adjacency matrix as described above harbouring connections reported by the structual and statistcal adjacency matrices. In the second entry a ‘matrix‘ of type ‘character‘ the corresonding type/putative link at this position.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package = "MetNet")
x_test <- as.matrix(x_test)
functional_groups <- rbind(
    c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
    c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
    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"),
    correlation_adjust = "bonferroni")
stat_adj <- threshold(stat_adj_l, type = "top2", args = list(n = 10))
combine(struct_adj, stat_adj)
```

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

Description

‘correlation‘ infers an adjacency matrix using correlation using the ‘cor‘ function (from the ‘stats‘ package), ‘pcor‘ (from ‘ppcor‘) or ‘spcor‘ (from ‘ppcor‘). ‘correlation‘ extracts the reported pairwise correlation coefficients from the function ‘corAndPvalue‘, ‘pcor‘ or ‘spcor‘ and will return the weighted adjacency matrix of the absolute correlation values.

Usage

```
correlation(x, type = "pearson", use = "pairwise.complete.obs")
```

Arguments

`x`	matrix, where columns are the samples and the rows are features (metabolites), cell entries are intensity values
`type`	‘character‘, either "pearson", "spearman", "pearson_partial", "spearman_partial", "pearson_semipartial" or "spearman_semipartial".
`use`	‘character‘ string giving a method for computing covariance in the presence of missing values, Only for ‘type = "pearson"‘ or ‘type = "spearman"‘. For further information see ‘?stats::cor‘

Details

If ‘"pearson"‘ or ‘"spearman"‘ is used as a ‘method‘, the function ‘corAndPvalue‘ from ‘stats‘ will be employed.

If ‘"pearson_partial"‘ or ‘"spearman_partial"¿ is used as a ‘method‘ the function ‘pcor‘ from ‘spcor‘ will be employed.

If ‘"pearson_semipartial"‘ or ‘"spearman_semipartial"‘ is used as a ‘method‘ the function ‘spcor‘ from ‘spcor‘ will be employed.

‘type‘ will be passed to argument ‘method‘ in ‘cor‘ (in the case of ‘"pearson"‘ or ‘"spearman"‘) or to ‘method‘ in ‘pcor‘ (‘"pearson"‘ and ‘"spearman"‘ for ‘"pearson_partial"‘ and ‘"spearman_partial"‘, respectively) or to ‘method‘ in ‘spcor‘ (‘"pearson"‘ or ‘"spearman"‘ for ‘"pearson_semipartial"‘ and ‘"spearman_semipartial"‘, respectively).

Value

matrix, matrix with edges inferred from correlation algorithm ‘corAndPvalue‘, ‘pcor‘ or ‘spcor‘ (depending on the chosen ‘method‘)

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)
correlation(x, type = "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")
```

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‘

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")
```

`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_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))

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 matrix 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 elue 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(structural, x, transformation)
```

Arguments

`structural`	‘list‘ returned by the function ‘structural‘. The first entry stores the ‘numeric‘ matrix with edges inferred by mass differences. The second entry stores the ‘character‘ matrix with the type (corresponding to the ‘"group"‘ column in ‘transformation‘).
`x`	‘matrix‘, where columns are the samples and the rows are features (metabolites), cell entries are intensity values, ‘x‘ contains the column ‘"rt"‘ that has the 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 ‘"group"‘, and ‘"rt"‘ that will be used for correction of transformation of (functional) groups based on retention time shifts derived from ‘x‘

Details

‘rtCorrection‘ is used to correct the unweighted adjacency matrix 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 second list element of ‘structural‘ and matches the elements in the ‘"group"‘ column against the character matrix. In case of matches, ‘rtCorrection‘ accesses the ‘"rt"‘ column 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 connections will always be removed if there is an inconsistency with any of the given transformation.

Value

‘list‘ containing two matrices. The first entry stores the ‘numeric‘ ‘matrix‘ with edges inferred mass differences corrected by retention time shifts. The second entry stores the ‘character‘ matrix with the type (corresponding to the ‘"group‘" column in ‘transformation“) is stored.

Author(s)

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 ]),
                             rt = transformation[, 4])
struct_adj <- structural(x_test, transformation, ppm = 5)
struct_adj_rt <- rtCorrection(struct_adj, x_test, transformation)
```

`statistical` *Create a list of adjacency matrices from statistical methods*

Description

The function ‘statitical‘ infers adjacency matrix topologies from statistical methods and returns matrices of these networks in a ‘list‘. 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 and semipartial), Spearman correlation (also partial and semipartial) and score-based structure learning (Bayes). The function returns a list 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"‘, ‘"pearson_semipartial"‘, ‘"spearman"‘, ‘"spearman_partial"‘, ‘"spearman_semipartial"‘, ‘"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 and semipartial), Spearman correlation (also partial and semipartial) and Constraint-based structure learning (Bayes).

‘statistical‘ calls the function ‘lasso‘, ‘randomForest‘, ‘clr‘, ‘aracne‘, ‘correlation‘ (for ‘"pearson"‘, ‘"pearson_partial"‘, ‘"pearson_semipartial"‘, ‘"spearman"‘, ‘"spearman_partial"‘, ‘"spearman_semipartial"‘) and/or ‘bayes‘ as specified by ‘model‘. It will create adjacency matrices using the specified methods and will return a ‘list‘ containing the weighted adjacency matrices.

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

Value

‘list‘ containing the respective adjacency matrices 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"))
```

---

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‘ of calculated theoretical differences of loss/addition of functional groups. ‘structural‘ returns the unweighted ‘numeric‘ ‘matrix‘ together with a ‘character‘ ‘matrix‘ with the type of loss/addition as a list at the specific positions.

Usage

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

Arguments

x	‘matrix‘, 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
ppm	‘numeric‘, mass accuracy of m/z features in parts per million (ppm)
directed	‘logical‘, if ‘TRUE‘ absolute values of m/z differences will be taken to query against ‘transformation‘ (irrespective the sign of ‘mass‘) and an undirected adjacency matrix will be returned, if ‘FALSE‘ a directed adjacency matrix will be returned with links reported that match the transformations defined in ‘transformation‘ (respecting the sign of ‘mass‘)

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‘).

Value

‘list‘ containing two matrices. The first entry stores the ‘numeric‘ ‘matrix‘ with edges inferred from mass differences. The second entry stores the ‘character‘ ‘matrix‘ with the type (corresponding to the ‘"group"‘ column in ‘transformation‘) is stored

Author(s)

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]))
struct_adj <- structural(x_test, transformation, ppm = 5, directed = TRUE)
```

threeDotsCall — *Check if passed arguments match the function's formal arguments and call the function with the checked arguments*

Description

The function 'threeDotsCall' gets the formal arguments of a function 'fun' and checks if the passed arguments '...' matches the formal arguments. 'threeDotsCall' will call the function 'fun' with the filtered arguments and will return the result of the function call and the given arguments.

Usage

```
threeDotsCall(fun, ...)
```

Arguments

fun	'function' to check for arguments and to call
...	arguments to be tested to be passed to 'fun'

Details

Used internally in 'lasso', 'randomForest', 'bayes', 'statistical' and 'threshold'.

'threeDotsCall' will not remove duplicated arguments and throw an error.

Value

Returned object given the function call with passed arguments

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
MetNet:::threeDotsCall(stats::sd, x = 1:10, y = 1:10)
## in contrast to the above example, the following example will result in an
## error
## Not run: stats::sd(x = 1:10, y = 1:10)
```

threshold — *Threshold the statistical adjacency matrices*

Description

The function 'threshold' takes as input a list of adjacency matrices as returned from the function 'statistical'. 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"').

Usage

```
threshold(statistical, type, args, values = c("all", "min", "max"), ...)
```

Arguments

statistical	‘list‘ containing adjacency matrices
type	‘character‘, either ‘"threshold"‘, ‘"top1‘, ‘"top2‘ or ‘"mean"‘
args	‘list‘ of arguments, has to contain thresholds for weighted adjacency matrices depending on the statistical model (a named list, where names are identical to ‘model‘s in ‘statistical‘) or a numerical vector of length 1 that denotes the number of top ranks written to the consensus matrix (a named list with entry ‘n‘)
values	‘character‘, take from the adjacency matrix all values ("all"), the minimum of the pairs ("min") or the maximum ("max") a^*_ij = min(a_ij, a_ji) a^*_ij = max(a_ij, a_ji)
...	parameters passed to the function ‘consensus‘ in the ‘sna‘ package (only for ‘type = "threshold"‘)

Details

The entries of ‘args‘ differ depending on the argument ‘type‘. If ‘type = "treshhold"‘, then ‘args‘ has to contain numeric vector of length 1 with names equal to ‘names(statistical)‘ for each ‘model‘ (‘names(statistical)‘) and the entry ‘threshold‘, a numerical ‘vector(1)‘ to threshold the consensus matrix after using the ‘consensus‘ function from the ‘sna‘ package. Depending on the chosen ‘method‘ in ‘consensus‘, the ‘threshold‘ value of the consensus adjacency matrix should be chosen accordingly to report a connection by different statistical modelss.

When combining the adjacency matrices the ‘threshold‘ value defines if an edge is reported or not. For ‘method = "central.graph"‘ threshold should be set to 1 by default. For other values of ‘method‘, the value should be carefully defined by the user.

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 and semipartial correlation) and ‘clr‘ and ‘aracne‘ are taken as they are.

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

Value

‘matrix‘, binary adjacency matrix given the links supported by the ‘type‘ and the ‘args‘

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("pearson" = 0.95, "spearman" = 0.95, n = 10)
l <- statistical(x, model = model)

## type = "threshold"
args <- list("pearson" = 0.95, "spearman" = 0.95, threshold = 1)
threshold(statistical = l, type = "threshold", args = args)

## type = "top1"
args <- list(n = 10)
threshold(statistical = l, type = "top1", args = args)

## type = "top2"
threshold(statistical = l, type = "top2", args = args)

## type = "mean"
threshold(statistical = l, 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)
```

Arguments

ranks	'matrix' containing the ranks per statistical model (in columns) and per feature pair (in rows)
type	'character', either '"top1"', '"top2"' or '"mean"'

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

Examples

```
ranks <- matrix(c(c(1, 2, 3), c(2, 1, 3)), ncol = 2)

## type = "top1"
MetNet:::topKnet(ranks = ranks, type = "top1")

## type = "top2"
MetNet:::topKnet(ranks = ranks, type = "top2")

## type = "mean"
MetNet:::topKnet(ranks = ranks, type = "mean")
```

---

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 <- addTo_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