Package 'MetNet'

June 17, 2019

Type Package

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

Version 1.3.0

Date 2019-04-03

VignetteBuilder knitr

Depends R (>= 3.5), stats (>= 3.5)

Imports bnlearn (>= 4.3), BiocParallel (>= 1.12.0), methods (>= 3.5), mpmi (>= 0.42), parmigene (>= 1.0.2), ppcor (>= 1.1), rfPermute (>= 2.1.5), sna (>= 2.4), stabs (>= 0.6), WGCNA (>= 1.61)

Suggests BiocGenerics (>= 0.24.0), BiocStyle (>= 2.6.1), igraph (>= 1.1.2), knitr (>= 1.11)

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-2

RoxygenNote 6.0.1

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

git_branch master

git_last_commit 2fdf457

git_last_commit_date 2019-05-02

Date/Publication 2019-06-16

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 <- createStructuralAdjacency(x_test, functional_groups, ppm = 5)
stat_adj <- createStatisticalAdjacency(x_test,
    model = c("pearson", "spearman","bayes"),
    adjust_correlation = "bonferroni")
cons_adj <- combineStructuralStatistical(struct_adj[[1]], stat_adj)
```

---

addToList *Add adjacency matrix to list*

Description

This helper function used in the function `createStatisticalAdjacencyList` adds a adjacency matrix to a list of adjacency matrices.

Usage

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

Arguments

`l`	list of adjacency matrices
`name`	character, name of newly created entry
`object`	matrix containing the adjacency matrix to be added

Details

Used internally in `createStatisticalAdjacencyList`

Value

list containing the existing adjacency matrices and the added adjacency matrix

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
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

`.information` infers an adjacency matrix using the algorithm for the reconstruction of accurate cellular networks using the `aracne.a` function from the `parmigene` package. The presence/absence is based on if the returned value exceeds a user-defined threshold value. `aracne` will return the adjacency matrix containing the presence/absence value.

Usage

```
aracne(mi, eps=0.05, aracne_threshold=0)
```

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
`aracne_threshold`	numeric, if the aracne value exceeds the threshold (aracne$_i,j$ > threshold, where aracne$_i, j$ is the aracne value of the ith row feature and of the jth column feature), the connection is defined as present, if the aracne value is lower than the threshold value (aracne$_i,j$ <= threshold) there is no statistical connection reported.

Details

For more details on the `aracne.a` function, refer to ?parmigene::aracne.a.

Value

matrix, matrix with edges inferred from Reconstruction of accurate cellular networks algorithm `aracne`

Author(s)

Thomas Naake, <thomasnaake @googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
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, aracne_threshold=0)
```

---

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

Description

bayes infers an adjacency matrix using constraint-based structure learning algorithm fast.iamb from the bnlearn package. bayes extracts then the reported connections from running the fast.iamb function and assigns the arcs of the discrete Bayesian connections to binary values. The adjacency matrix is returned by bayes.

Usage

```
bayes(x, ...)
```

Arguments

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

Details

For use of the parameters used in the fast.iamb function, refer to ?bnlearn::fast.iamb.

Value

matrix, matrix with edges inferred from constraint-based structure learning algorithm fast.iamb

Author(s)

Thomas Naake, <thomasnaake @googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
x <- as.matrix(x)
bayes(x)
```

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. The presence/absence is based on if the returned value exceeds a user-defined threshold value. clr will return the adjacency matrix containing the presence/absence value.

Usage

```
clr(mi, clr_threshold=0)
```

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.
clr_threshold	numeric, if the clr value exceeds the threshold (clr$_i,j$ > threshold, where clr$_i, j$ is the clr value of the ith row feature and of the jth column feature), the connection is defined as present, if the clr value is lower than the threshold value (clr$_i,j$ $\leq$ threshold) there is no statistical connection reported.

Details

For more details on the clr function, refer to ?parmigene::clr.

Value

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

Author(s)

Thomas Naake, <thomasnaake @googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
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, clr_threshold=0)
```

combineStructuralStatistical *Combine structural and statistical adjacency matrix*

Description

The function `combineStructuralStatistical` 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`. `combineStructuralStatistical` returns this consensus matrix supported by the structural and statistical adjacency matrices.

Usage

```
combineStructuralStatistical(structure, statistical, threshold=1)
```

Arguments

`structure`	matrix containing structural adjacency matrix
`statistical`	matrix containing 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

a matrix containing the consensus adjacency matrix as described above harbouring connections reported by the structual and statistcal adjacency matrices.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

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 <- createStructuralAdjacency(x_test, functional_groups, ppm=5)
```

```
stat_adj <- createStatisticalAdjacency(x_test,
    model=c("pearson", "spearman","bayes"),
    correlation_adjust="bonferroni")
combineStructuralStatistical(struct_adj[[1]], stat_adj)
```

---

consensusAdjacency *Create a consensus adjacency matrix of statistical adjacency matrices*

Description

The function takes a list of parameters (l) as input and creates a consensus adjacency matrix from these adjacency matrices by calling the function consensus from the sna package. Depending on the chosen method in consensus, the threshold of the consensus adjacency matrix should be chosen accordingly to report a connection by different statistical methods.

Usage

```
consensusAdjacency(l, threshold=1, ...)
```

Arguments

l	list, each entry of the list contains an adjacency matrix
threshold	numeric, when combining the adjacency matrices the threshold parameter defines if an edge is reported or not. For method="central.graph" threshold is set to 1 by default. For other values of method, the value should be carefully defined by the user. If threshold is set to NULL (default), it will be set to 1 internally.
...	parameters passed to the function consensus in the sna package

Details

consensusAdjacency is a wrapper function of the consensus function of the sna package. For use of the parameters used in the consensus function, refer to ?sna::consensus.

Value

matrix, consensus matrix from adjacency matrices

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
x <- as.matrix(x)
stat_adj_l <- createStatisticalAdjacencyList(x, c("pearson", "spearman"))
consensusAdjacency(stat_adj_l)
```

correlation *Create an adjacency matrix based on correlation*

Description

correlation infers an adjacency matrix using correlation using the corAndPvalue function (from the WGCNA package), pcor (from ppcor) or spcor (from ppcor). correlation extracts the reported p-values from the function corAndPvalue, pcor or spcor that can be adjusted for multiple testing (correlation_adjust parameter) and will return an unweighted adjacency matrix containing edges if the (adjusted) p-value is below the value defined by correlation_threshold.

Usage

```
correlation(x, correlation_adjust="none", type="pearson",
                                        correlation_threshold=0.05, ...)
```

Arguments

x	matrix, where columns are the samples and the rows are features (metabolites), cell entries are intensity values
correlation_adjust	character
type	character, either "pearson", "spearman", "pearson_partial", "spearman_partial", "pearson_semipartial" or "spearman_semipartial". type will be passed to argument method in corAndPvalue (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)
correlation_threshold	numeric, significance level α (default: 0.05), if the (adjusted) p-values exceed this value, there is no statistical connection between features
...	parameters passed to corAndPvalue (argument adjust will be ignored)

Details

If "pearson" or "spearman" is used as a method the function corAndPvalue from WGCNA 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. For use of the parameters used in the corAndPvalue function, refer to ?WGCNA::corAndPvalue.

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[, 3:dim(x_test)[2]]
x <- as.matrix(x)
correlation(x, correlation_adjust="bonferroni", type="pearson")
```

`createStatisticalAdjacency` *Create statistical adjacency matrix*

Description

`createStatisticalAdjacency` creates a consensus adjacency matrix given the models to use.

Usage

```
createStatisticalAdjacency(x, model, threshold=1, ...)
```

Arguments

`x`	matrix that contains intensity values of features/metabolites (rows) per sample (columns).
`model,`	character, vector containing the model that will be used ("lasso", "randomForest", "clr", "aracne", "pearson", "pearson_partial", "pearson_semipartial","spearman", "spearman_partial", "spearman_semipartial", "bayes")
`threshold`	numeric, when combining the adjacency matrices the threshold parameter defines if an edge is reported or not. For `method="central.graph"` threshold is set to 1 by default. For other values of method, the value should be carefully defined by the user. If threshold is set to NULL (default), it will be set to 1 internally.
`...`	parameters passed to the functions `lasso`, `randomForest`, `clr`, `aracne`, `correlation`, `bayes` and/or `consensusAdjacency`

Details

`createStatisticalAdjacency` is a wrapper function for the functions `createStatisticalAdjacencyList` and `consensusAdjacency`. See `?createStatisticalAdjacencyList` and `?consensusAdjacency` for further details. The function `createStatisticalAdjacencyList` includes functionality to caluclate 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).

Value

matrix, containing binary values if a connection is present or not

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
x <- as.matrix(x)
createStatisticalAdjacency(x, c("pearson", "spearman"))
```

`createStatisticalAdjacencyList`
Create a list of statistical adjacency matrices

Description

The function infers adjacency matrix topologies from statistical methods and returns matrices of these networks in a list. The function includes functionality to caluclate 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). The function returns a list of adjacency matrices that are defined by `model`.

Usage

```
createStatisticalAdjacencyList(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

`createStatisticalAdjacencyList` 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 unweighted adjacency matrix (if `model` is of length 1) or append these unweighted adjacency matrices to a list (if `model` is of length > 1). 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[, 3:dim(x_test)[2]]
x <- as.matrix(x)
createStatisticalAdjacencyList(x, c("pearson", "spearman"))
```

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

Description

The function `createStructuralAdjacency` infers an adjacency matrix using differences in m/z values that are matched against a `data.frame` of theoretically calculated differences of loss/addition of functional groups. `createStructuralAdjacency` returns the unweighted adjacency matrix together with a character matrix with the type of loss/addition as a list at the specific positions.

Usage

```
createStructuralAdjacency(x, transformation, ppm=5)
```

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)

Details

`createStructuralAdjacency` accesses the column `'mz'` of `x` to infer structural topologies based on the functional groups supplied 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, in the first list entry the matrix with edges inferred mass differences is stored, in the second list entry the 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("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"))
transformation <- data.frame(group=transformation[,1],
                             formula=transformation[,2],
                             mass=as.numeric(transformation[,3]))
struct_adj <- createStructuralAdjacency(x_test, transformation, ppm=5)
```

---

lasso *Create a 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 presence/absence of this connection to a matrix that is returned.

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[, 3:dim(x_test)[2]]
x <- as.matrix(x)
x_z <- t(apply(x, 1, function(y) (y - mean(y)) / sd(y)))
## Not run: lasso(x_z, PFER=0.75, 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.

Usage

```
mat_test
```

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.

Usage

```
mat_test_z
```

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).

Usage

peaklist

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 a adjacency matrix based on random forest*

Description

randomForest infers an adjacency matrix using random forest using the rfPermute function from the rfPermute package. randomForest extracts the p-values by the function rp.importance and writes the presence/absence based on the significance value ($\alpha \leq 0.05$) of this connection to a matrix. The adjacency matrix is returned.

Usage

```
randomForest(x, parallel=FALSE, randomForest_adjust="none", ...)
```

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.
randomForest_adjust	character, correction method for p-values from rp.importance, randomForest_adjust will be passed to the p.adjust function and should be one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
...	parameters passed to rfPermute.default

Details

For use of the parameters used in the rfPermute function, refer to ?rfPermute::rfPermute.default.

Value

matrix, matrix with edges inferred from random forest algorithm rfPermute and rp.importance

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
data("x_test", package="MetNet")
x <- x_test[, 3:dim(x_test)[2]]
x <- as.matrix(x)
## Not run: 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(struct_adj, x, transformation)
```

Arguments

`struct_adj`	list returned by the function `createStructuralAdjacency`, in the first list entry the matrix with edges inferred mass differences is stored, in the second list entry the matrix with the type (corresponding to the `'group'` column in `transformation`) is stored
`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 adjacency matrix returned by `createStructuralAdjacency` 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 `struct_adj` 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, in the first list entry the matrix with edges inferred mass differences corrected by retention time shifts is stored, in the second list entry the 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("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", "-"))
transformation <- data.frame(group=transformation[,1],
                             formula=transformation[,2],
                             mass=as.numeric(transformation[,3]),
                             rt=transformation[,4])
struct_adj <- createStructuralAdjacency(x_test, transformation, ppm=5)
struct_adj_rt <- rtCorrection(struct_adj, x_test, transformation)
```

---

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 remove duplicated arguments. `threeDotsCall` will call the function `fun` with the filtered arguments and will return the result.

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`, `correlation`, `bayes`, `consensusAdjacency`

Value

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)
```

---

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.

Usage

```
x_test
```

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