Package ‘DAPAR’

November 20, 2016

Type Package
Title Tools for the Differential Analysis of Proteins Abundance with R
Version 1.6.0
Date 2016-06-27
Author Samuel Wieczorek [cre,aut],
      Florence Combes [aut],
      Thomas Burger [aut],
      Cosmin Lazar [ctb],
      Alexia Dorffer [ctb]
Maintainer Samuel Wieczorek <samuel.wieczorek@cea.fr>
Description This package contains a collection of functions for the
      visualisation and the statistical analysis of proteomic data.
License Artistic-2.0
VignetteBuilder knitr
Depends R (>= 3.3)
Suggests BiocGenerics, Biobase, testthat, BiocStyle, Prostar
Imports MSnbase, RColorBrewer,stats,preprocessCore,Cairo,png,
      lattice,reshape2,gplots, pcaMethods,ggplot2,
      limma,knitr,tmvn, norm, impute, imputeLCMD, doParallel,
      parallel, foreach, grDevices, graphics, openxlsx, utils, cp4p
      (>= 0.3.5), scales, Matrix, vioplot
biocViews Proteomics, Normalization, Preprocessing, MassSpectrometry,
      QualityControl, DataImport
NeedsCompilation no
RoxygenNote 5.0.1

R topics documented:

boxPlotD .................................................. 3
BuildAdjacencyMatrix .................................. 4
BuildColumnToProteinDataset ............................. 4
compareNormalizationD .................................. 5
corrMatrixD ............................................ 6
CountPep ............................................... 7
createMSnset .......................................... 7
R topics documented:

deleteLinesFromIndices ........................................... 8
densityPlotD .......................................................... 9
diffAna ................................................................. 10
diffAnaComputeFDR .................................................. 10
diffAnaGetSignificant ............................................... 11
diffAnaLimma ......................................................... 12
diffAnaSave .......................................................... 13
diffAnaVolcanoplot ................................................. 14
diffAnaWelch ........................................................ 15
getIndicesConditions ............................................... 15
getIndicesOfLinesToRemove ....................................... 16
getNumberOf .......................................................... 17
getNumberOfEmptyLines .......................................... 17
getPaletteForLabels ............................................... 18
getPaletteForReplicates ......................................... 19
getPourcentageOfMV ............................................... 19
getProcessingInfo ................................................ 20
getProteinsStats .................................................. 20
GraphPepProt ....................................................... 21
heatmap.DAPAR ..................................................... 22
heatmapD ............................................................. 23
limmaCompleteTest ............................................... 23
MeanPeptides ......................................................... 24
mvFilter .............................................................. 25
mvFilterFromIndices ............................................... 26
mvFilterGetIndices ................................................ 26
tmvHisto ............................................................. 27
tmvImage ............................................................. 28
tmvImputation ........................................................ 29
mvPerLinesHisto ................................................... 29
mvPerLinesHistoPerCondition ................................... 30
mvTypePlot ........................................................... 31
normalizeD .......................................................... 31
pepAggregate ........................................................ 32
proportionConRev .................................................. 33
removeLines ........................................................ 34
SumPeptides .......................................................... 34
test ................................................................. 35
testWithoutNA ....................................................... 35
TopnPeptides ......................................................... 36
USPep25 .............................................................. 36
varianceDistD ........................................................ 37
violinPlotD .......................................................... 38
wrapper.boxPlotD .................................................. 39
wrapper.compareNormalizationD .................................. 39
wrapper.corrMatrixD ............................................... 40
wrapper.densityPlotD ............................................. 41
wrapper.diffAnaLimma ............................................. 42
wrapper.diffAnaWelch ............................................. 42
wrapper.heatmapD ................................................ 43
wrapper.mvHisto .................................................... 44
wrapper.mvImage ................................................... 44
**boxPlotD**

Builds a boxplot from a dataframe

**Description**

Boxplot for quantitative proteomics data

**Usage**

```r
boxPlotD(qData, dataForXAxis = NULL, labels = NULL, group2Color = "Condition")
```

**Arguments**

- `qData`: A dataframe that contains quantitative data.
- `labels`: A vector of the conditions (labels) (one label per sample).
- `group2Color`: A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

**Value**

A boxplot

**Author(s)**

Florence Combes, Samuel Wieczorek

**See Also**

densityPlotD

**Examples**

```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
types <- c("Label","Analyt.Rep")
dataForXAxis <- Biobase::pData(UPSpep25)[,types]
labels <- Biobase::pData(UPSpep25)[,"Label"]
boxPlotD(qData, dataForXAxis, labels)
```
**BuildAdjacencyMatrix**

*Function matrix of appartenance group*

**Description**

Method to create a binary matrix with proteins in columns and peptides in lines on a MSnSet object (peptides)

**Usage**

```
BuildAdjacencyMatrix(obj.pep, protID, unique = TRUE)
```

**Arguments**

- `obj.pep`: An object (peptides) of class `MSnbase`.
- `protID`: The name of proteins ID column
- `unique`: A boolean to indicate whether only the unique peptides must be considered (TRUE) or if the shared peptides have to be integrated (FALSE).

**Value**

A binary matrix

**Author(s)**

Florence Combes, Samuel Wieczorek, Alexia Dorffer

**Examples**

```
data(UPSpep25)
BuildAdjacencyMatrix(UPSpep25, "Protein.group.IDs", TRUE)
```

---

**BuildColumnToProteinDataset**

*creates a column for the protein dataset after aggregation by using the previous peptide dataset.*

**Description**

This function creates a column for the protein dataset after aggregation by using the previous peptide dataset.

**Usage**

```
BuildColumnToProteinDataset(peptideData, matAdj, columnName)
```
compareNormalizationD

Arguments

peptideData A data.frame of meta data of peptides. It is the fData of the MSnset object.
matAdj The adjacency matrix used to aggregate the peptides data.
columnName The name of the column in fData(peptides_MSnset) that the user wants to keep in the new protein data.frame.

Value

A vector

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
data <- Biobase::fData(UPSpep25)
name <- "organism"
BuildColumnToProteinDataset(data, M, name )

compareNormalizationD Builds a plot from a dataframe

Description

Plot to compare the quantitative proteomics data before and after normalization

Usage

compareNormalizationD(qDataBefore, qDataAfter, labelsForLegend = NULL,
indData2Show = NULL, group2Color = "Condition")

Arguments

qDataBefore A dataframe that contains quantitative data before normalization.
qDataAfter A dataframe that contains quantitative data after normalization.
labelsForLegend A vector of the conditions (labels) (one label per sample).
indData2Show A vector of the indices of the columns to show in the plot. The indices are those of indices of the columns int the data.frame qDataBefore.
group2Color A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

Value

A plot
Author(s)
Samuel Wieczorek

Examples
data(UPSpep25)
qDataBefore <- Biobase::exprs(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
qDataAfter <- normalizeD(qDataBefore,labels,"Median Centering",
"within conditions")
compareNormalizationD(qDataBefore, qDataAfter, labels)

corrMatrixD

Description
Correlation matrix based on a MSnSet object

Usage
corrMatrixD(qData, samplesData, gradientRate = 5)

Arguments
qData A dataframe of quantitative data.
samplesData A dataframe where lines correspond to samples and columns to the meta-data
for those samples.
gradientRate The rate parameter to control the exponential law for the gradient of colors

Value
A colored correlation matrix

Author(s)
Florence Combes, Samuel Wieczorek

Examples
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
samplesData <- Biobase::pData(UPSpep25)
corrMatrixD(qData, samplesData)
CountPep

Compute the number of peptides used to aggregate proteins

**Description**

This function computes the number of peptides used to aggregate proteins.

**Usage**

CountPep(M)

**Arguments**

- **M**: A "valued" adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

**Value**

A vector of boolean which is the adjacency matrix but with NA values if they exist in the intensity matrix.

**Author(s)**

Alexia Dorffer

**Examples**

```r
data(UPSpep25)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
CountPep(M)
```

---

createMSnset

Creates an object of class **MSnSet** from text file

**Description**

Builds an object of class **MSnSet** from a single tabulated-like file for quantitative and meta-data and a dataframe for the samples description. It differs from the original **MSnSet** builder which requires three separated files tabulated-like quantitative proteomic data into a **MSnSet** object, including meta-data.

**Usage**

```r
createMSnset(file, metadata = NULL, indExpData, indFData, indiceID = NULL, logData = FALSE, replaceZeros = FALSE, pep_prot_data = NULL)
```
deleteLinesFromIndices

Arguments

**file**
The name of a tab-separated file that contains the data.

**metadata**
A dataframe describing the samples (in lines).

**indExpData**
A vector of string where each element is the name of a column in designTable that have to be integrated in the fData() table of the MSnSet object.

**indiceID**
The indice of the column containing the ID of entities (peptides or proteins)

**logData**
A boolean value to indicate if the data have to be log-transformed (Default is FALSE)

**replaceZeros**
A boolean value to indicate if the 0 and NaN values of intensity have to be replaced by NA (Default is FALSE)

**pep_prot_data**
A string that indicates whether the dataset is about peptides or proteins.

Value

An instance of class MSnSet.

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```r
exprsFile <- system.file("extdata", "UPSpep25.txt", package="DAPAR")
metadataFile <- system.file("extdata", "samples.txt", package="DAPAR")
metadata = read.table(metadataFile, header=TRUE, sep="\t", as.is=TRUE)
indExpData <- c(56:61)
indFData <- c(1:55,62:71)
indiceID <- 64
createMSnset(exprsFile, metadata, indExpData, indFData, indiceID, pep_prot_data = "peptide")
```

---

deleteLinesFromIndices

*Delete the lines in the matrix of intensities and the metadata table given their indice.*

Description

Delete the lines of exprs() table identified by their indice.

Usage

```r
deleteLinesFromIndices(obj, deleteThat = NULL, processText = NULL)
```

Arguments

**obj**
An object of class MSnSet containing quantitative data.

**deleteThat**
A vector of integers which are the indices of lines to delete.

**processText**
A string to be included in the MSnSet object for log.
densityPlotD

Value
An instance of class MSnSet that have been filtered.

Author(s)
Florence Combes, Samuel Wieczorek

Examples
data(UPSpep25)
mvFilter(UPSpep25, c(1:10))

densityPlotD Builds a density plot from a dataframe

Description
Densityplot of quantitative proteomics data over samples.

Usage
densityPlotD(qData, labelsForLegend = NULL, indData2Show = NULL, group2Color = "Condition")

Arguments
qData A dataframe that contains quantitative data.
labelsForLegend A vector of the conditions (labels) (one label per sample).
indData2Show A vector of indices to show in densityplot. If NULL, then all labels are displayed.
group2Color A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

Value
A density plot

Author(s)
Florence Combes, Samuel Wieczorek

See Also
boxPlotD, varianceDistD

Examples
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
labels <- lab2Show <- Biobase::pData(UPSpep25)[,"Label"]
densityPlotD(qData, labels)
**diffAna**

This function performs a differential analysis on an MSnSet object (adapted from limma)

### Description
Performing a differential analysis on an MSnSet object, based on limma functions.

### Usage
```r
diffAna(qData, design)
```

### Arguments
- **qData**
  A dataframe that contains quantitative data.
- **design**
  The design matrix as described in the limma package documentation.

### Value
A dataframe with the p-value and log(Fold Change) associated to each element (peptide/protein)

### Author(s)
Florence Combes, Samuel Wieczorek

### Examples
```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
design <- cbind(cond1=1, cond2 = rep(0,nrow(Biobase::pData(UPSpep25))))
rownames(design) <- rownames(Biobase::pData(UPSpep25))
labels <- Biobase::pData(UPSpep25)[,"Label"]
indices <- getIndicesConditions(labels, "25fmol", "10fmol")
design[indices[iCond2,2] <- 1
diffAna(qData, design)
```

---

**diffAnaComputeFDR**

Computes the FDR corresponding to the p-values of the differential analysis using

### Description
This function is a wrapper to the function adjust.p from the cp4p package. It returns the FDR corresponding to the p-values of the differential analysis. The FDR is computed with the function p.adjust(stats).

### Usage
```r
diffAnaComputeFDR(data, threshold_PVal = 0, threshold_LogFC = 0, pi0Method = 1)
```
DiffanaGetSignificant

Arguments

- `data`: The result of the differential analysis processed by `diffAna`.
- `threshold_PVal`: The threshold on p-value to distinguish between differential and non-differential data.
- `threshold_LogFC`: The threshold on log(Fold Change) to distinguish between differential and non-differential data.
- `pi0Method`: The parameter `pi0.method` of the method `adjust.p` in the package `cp4p`.

Value

The computed FDR value (floating number).

Author(s)

Samuel Wieczorek

Examples

```r
data(UPSpep25)
obj <- wrapper.mvImputation(UPSpep25, "QRLC")
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(obj)
samplesData <- Biobase::pData(obj)
labels <- Biobase::pData(obj)[,"Label"]
limma <- diffAnaLimma(qData,samplesData, labels, condition1, condition2)
diffAnaComputeFDR(limma)
```

`DiffanaGetSignificant` *Returns a MSnSet object with only proteins significant after differential analysis.*

Description

Returns a MSnSet object with only proteins significant after differential analysis.

Usage

```r
diffAnaGetSignificant(obj)
```

Arguments

- `obj`: An object of class `MSnSet`.

Value

A MSnSet

Author(s)

Alexia Dorffer
diffAnaLimma

**diffAnaLimma**

Performs differential analysis on an **MSnSet** object, calling the **limma** package functions

**Description**

Method to perform differential analysis on an **MSnSet** object (calls the **limma** package function).

**Usage**

`diffAnaLimma(qData, samplesData, labels, condition1, condition2)`

**Arguments**

- `qData` A dataframe that contains quantitative data.
- `samplesData` A dataframe where lines correspond to samples and columns to the meta-data for those samples.
- `labels` A vector of the conditions (labels) (one label per sample).
- `condition1` A vector that contains the names of the conditions considered as condition 1
- `condition2` A vector that contains the names of the conditions considered as condition 2

**Value**

A dataframe as returned by the **limma** package

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
condition1 <- "25fmol"
condition2 <- "10fmol"
resLimma <- wrapper.diffAnaLimma(UPSpep25, condition1, condition2)
obj <- diffAnaSave(UPSpep25, resLimma, "limma", condition1, condition2)
signif <- diffAnaGetSignificant(obj)
```
diffAnaSave

Returns a MSnSet object with the results of the differential analysis performed with limma package.

Description

This method returns a MSnSet object with the results of differential analysis.

Usage

diffAnaSave(obj, data, method = "limma", condition1, condition2, threshold_pVal = 1e-60, threshold_logFC = 0, fdr = 0, calibrationMethod = "pounds")

Arguments

obj
An object of class MSnSet.
data
The result of the differential analysis processed by diffAna
method
The method used for differential analysis. Available choices are: "limma", "Welch"
condition1
A vector containing the names (some values of the slot "Label" of pData() of the first condition.
condition2
A vector containing the names (some values of the slot "Label" of pData() of the second condition.
threshold_pVal
A float that indicates the threshold on p-value chosen to discriminate differential proteins.
threshold_logFC
A float that indicates the threshold on log(Fold Change) to discriminate differential proteins.
fdr
The FDR based on the values of threshold_pVal and threshold_logFC
calibrationMethod
The calibration method used to compute the calibration plot

Value

A MSnSet

Author(s)

Alexia Dorffer, Samuel Wieczorek

Examples

data(UPSpep25)
condition1 <- '25fmol'
condition2 <- '10fmol'
limma <- wrapper.diffAnaLimma(UPSpep25, condition1, condition2)
obj <- diffAnaSave(UPSpep25, limma, "limma", condition1, condition2)
**diffAnaVolcanoplot**  
*Volcanoplot of the differential analysis*

**Description**

Plots a volcanoplot after the differential analysis. Typically, the log of Fold Change is represented on the X-axis and the log10 of the p-value is drawn on the Y-axis. When the `threshold_pVal` and the `threshold_logFC` are set, two lines are drawn respectively on the y-axis and the X-axis to visually distinguish between differential and non differential data.

**Usage**

```r
diffAnaVolcanoplot(logFC = NULL, pVal = NULL, threshold_pVal = 1e-60, threshold_logFC = 0, conditions = NULL)
```

**Arguments**

- `logFC`: A vector of the log(fold change) values of the differential analysis.
- `pVal`: A vector of the p-value values returned by the differential analysis.
- `threshold_pVal`: A floating number which represents the p-value that separates differential and non-differential data.
- `threshold_logFC`: A floating number which represents the log of the Fold Change that separates differential and non-differential data.
- `conditions`: A list of the names of condition 1 and 2 used for the differential analysis.

**Value**

A volcanoplot

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
condition1 <- '25fmol'
condition2 <- '10fmol'
data <- wrapper.diffAnaLimma(UPSpep25, condition1, condition2)
diffAnaVolcanoplot(data$logFC, data$P.Value)
```
**diffAnaWelch**

Performs a differential analysis on a MSnSet object using the Welch t-test

**Description**

Computes differential analysis on an MSnSet object, using the Welch t-test \( t.test(\text{stats}) \).

**Usage**

```r
diffAnaWelch(qData, labels, condition1, condition2)
```

**Arguments**

- `qData` : A dataframe that contains quantitative data.
- `labels` : A vector of the conditions (labels) (one label per sample).
- `condition1` : A vector containing the names of the conditions qData as condition 1
- `condition2` : A vector containing the names of the conditions considered as condition 2

**Value**

A dataframe with two slots : P.Value (for the p-value) and logFC (the log of the Fold Change).

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
condition1 <- "25fmol"
condition2 <- "10fmol"
qData <- Biobase::exprs(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
diffAnaWelch(qData, labels, condition1, condition2)
```

---

**getIndicesConditions**

Gets the conditions indices.

**Description**

Returns a list for the two conditions where each slot is a vector of indices for the samples.

**Usage**

```r
getIndicesConditions(labels, cond1, cond2)
```
Arguments

labels A vector of strings containing the column "Label" of the pData().
cond1 A vector of Labels (a slot in the pData() table) for the condition 1.
cond2 A vector of Labels (a slot in the pData() table) for the condition 2.

Value

A list with two slots iCond1 and iCond2 containing respectively the indices of samples in the pData() table of the dataset.

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
getIndicesConditions(labels, "25fmol", "10fmol")

getIndicesOfLinesToRemove

Get the indices of the lines to delete, based on a prefix string

Description

This function returns the indice of the lines to delete, based on a prefix string

Usage

getIndicesOfLinesToRemove(obj, idLine2Delete = NULL, prefix = NULL)

Arguments

obj An object of class MSnSet.
idLine2Delete The name of the column that correspond to the data to filter
prefix A character string that is the prefix to find in the data

Value

A vector of integers.

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
getIndicesOfLinesToRemove(UPSpep25, "Potential.contaminant", prefix="+")
**getNumberOf**

*Number of lines with prefix*

**Description**

Returns the number of lines, in a given column, where content matches the prefix.

**Usage**

```r
getNumberOf(obj, name = NULL, prefix = NULL)
```

**Arguments**

- `obj` An object of class `MSnSet`.
- `name` The name of a column.
- `prefix` A string

**Value**

An integer

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
getNumberOf(UPSpep25, "Potential.contaminant", "+")
```

---

**getNumberOfEmptyLines**  
*Returns the number of empty lines in the data*

**Description**

Returns the number of empty lines in a matrix.

**Usage**

```r
getNumberOfEmptyLines(qData)
```

**Arguments**

- `qData` A matrix corresponding to the quantitative data.

**Value**

An integer
getPaletteForLabels

Author(s)

Samuel Wieczorek

Examples

data(UPSlep25)
qData <- Biobase::exprs(UPSlep25)
getNumberOfEmptyLines(qData)

gpaletteForLabels <- getPaletteForLabels

data(UPSlep25)
labels <- Biobase::pData(UPSlep25)[,"Label"]
gpaletteForLabels(labels)

Description

Selects colors for the plots in DAPAR based on the different conditions in the dataset. The palette is derived from the brewer palette "Dark2" (see RColorBrewer).

Usage

gpaletteForLabels(labels)

Arguments

labels
A vector of labels (strings).

Value

A palette designed for the data manipulated in DAPAR

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(UPSlep25)
labels <- Biobase::pData(UPSlep25)[,"Label"]
gpaletteForLabels(labels)
getPaletteForReplicates

Palette for plot the replicates in DAPAR

Description

Selects colors for the plots in DAPAR based on the replicates in the dataset. The palette is derived from the brewer palette "Dark2" (see RColorBrewer).

Usage

getPaletteForReplicates(nColors)

Arguments

nColors The desired number of colors

Value

A palette designed for the data manipulated in DAPAR

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
n <- nrow(Biobase::pData(UPSpep25))
getPaletteForLabels(5)

getPourcentageOfMV

Percentage of missing values

Description

Returns the percentage of missing values in the quantitative data (exprs() table of the dataset).

Usage

getPourcentageOfMV(obj)

Arguments

obj An object of class MSnSet.

Value

A floating number
**getProteinsStats**

*computes the number of proteins that are only defined by specific peptides, shared peptides or a mixture of two.*

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
getProteinsStats(UPSpep25)
```

---

**getProcessingInfo**

*Returns the contains of the slot processing of an object of class MSnSet*

**Description**

Returns the contains of the slot processing of an object of class MSnSet.

**Usage**

```r
getProcessingInfo(obj)
```

**Arguments**

- `obj` An object (peptides) of class `MSnbase`.

**Value**

The slot processing of `obj@processingData`

**Author(s)**

Samuel Wieczorek

**Examples**

```r
data(UPSpep25)
getProcessingInfo(UPSpep25)
```
GraphPepProt

Arguments

matUnique The adjacency matrix with only specific peptides.
matShared The adjacency matrix with both specific and shared peptides.

Value

A list

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
protID <- "Protein.group.IDs"
MShared <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
MUnique <- BuildAdjacencyMatrix(UPSpep25, protID, TRUE)
getProteinsStats(MUnique, MShared)

GraphPepProt Function to create a histogram that shows the repartition of peptides w.r.t. the proteins

Description

Method to create a plot with proteins and peptides on a MSnSet object (peptides)

Usage

GraphPepProt(mat)

Arguments

mat An adjacency matrix.

Value

A histogram

Author(s)

Alexia Dorffer, Samuel Wieczorek

Examples

data(UPSpep25)
mat <- BuildAdjacencyMatrix(UPSpep25, "Protein.group.IDs")
GraphPepProt(mat)
heatmap.DAPAR

This function is inspired from the function heatmap.2 that displays quantitative data in the exprs() table of an object of class MSnSet. For more information, please refer to the help of the heatmap.2 function.

Description

Heatmap inspired by the heatmap.2 function.

Usage

heatmap.DAPAR(x, col = heat.colors(100), srtCol = NULL, labCol = NULL, labRow = NULL, key = TRUE, key.title = NULL, main = NULL, ylab = NULL)

Arguments

x A dataframe that contains quantitative data.
col colors used for the image. Defaults to heat colors (heat.colors).
srtCol angle of column labels, in degrees from horizontal
labCol character vectors with column labels to use.
labRow character vectors with row labels to use.
key logical indicating whether a color-key should be shown.
key.title main title of the color key. If set to NA no title will be plotted.
main main title; default to none.
ylab y-axis title; default to none.

Value

A heatmap

Author(s)

Samuel Wieczorek

Examples

data(testWithoutNA)
qData <- Biobase::exprs(testWithoutNA)
heatmapD(qData)
**heatmapD**

*This function is a wrapper to heatmap.2 that displays quantitative data in the exprs() table of an object of class MSnSet*

---

**Description**

Heatmap of the quantitative proteomic data of a MSnSet object

**Usage**

```r
heatmapD(qData, distance = "euclidean", cluster = "average", dendro = FALSE)
```

**Arguments**

- `qData`: A dataframe that contains quantitative data.
- `distance`: The distance used by the clustering algorithm to compute the dendrogram. See help(heatmap.2)
- `cluster`: The clustering algorithm used to build the dendrogram. See help(heatmap.2)
- `dendro`: A boolean to indicate if the dendrogram has to be displayed

**Value**

A heatmap

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(testWithoutNA)
qData <- Biobase::exprs(testWithoutNA)
heatmapD(qData)
```

---

**limmaCompleteTest**

*Computes a hierarchical differential analysis*

---

**Description**

This function is a limmaCompleteTest

**Usage**

```r
limmaCompleteTest(qData, Conditions, RepBio, RepTech, Contrast = 1)
```
MeanPeptides

Arguments

qData A matrix of quantitative data, without any missing values.
Conditions A vector of factor which indicates the name of the biological condition for each replicate.
RepBio A vector of factor which indicates the number of the bio rep for each replicate.
RepTech A vector of factor which indicates the number of the tech rep for each replicate.
Contrast Indicates if the test consists of the comparison of each biological condition versus each of the other ones (Contrast=1; for example H0: "C1=C2" vs H1: "C1!=C2", etc.) or each condition versus all others (Contrast=2; e.g. H0: "C1=(C2+C3)/2" vs H1: "C1!=(C2+C3)/2", etc. if there are three conditions).

Value

fdsfdgfdg

Author(s)

Quentin Giai-Gianetto

Examples

data(UPSpep25)
obj <- wrapper.mvImputation(UPSpep25, "QRILC")
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(obj)
RepBio <- RepTech <- factor(1:6)
conds <- factor(c(rep(condition1, 3), rep(condition2, 3)))
limma <- limmaCompleteTest(qData, conds, RepBio, RepTech)

MeanPeptides

Compute the intensity of proteins as the mean of the intensities of their peptides.

Description

This function computes the intensity of proteins as the mean of the intensities of their peptides.

Usage

MeanPeptides(matAdj, expr)

Arguments

matAdj An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
expr A matrix of intensities of peptides

Value

A matrix of intensities of proteins
mvFilter

Author(s)
Alexia Dorffer

Examples
data(UPSpep25)
protID <- "Protein.group.IDs"
matAdj <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
MeanPeptides(matAdj, Biobase::exprs(UPSpep25))

mvFilter Filter lines in the matrix of intensities w.r.t. some criteria

Description
Filters the lines of exprs() table with conditions on the number of missing values. The user chooses the minimum amount of intensities that is acceptable and the filter delete lines that do not respect this condition. The condition may be on the whole line or condition by condition.

Usage
mvFilter(obj, type, th, processText = NULL)

Arguments
obj An object of class MSnSet containing quantitative data.
type Method used to choose the lines to delete. Values are: "none", "wholeMatrix", "allCond", "atLeastOneCond"
th An integer value of the threshold
processText A string to be included in the MSnSet object for log.

Details
The different methods are: "wholeMatrix": given a threshold th, only the lines that contain at least th values are kept. "allCond": given a threshold th, only the lines which contain at least th values for each of the conditions are kept. "atLeastOneCond": given a threshold th, only the lines that contain at least th values, and for at least one condition, are kept.

Value
An instance of class MSnSet that have been filtered.

Author(s)
Florence Combes, Samuel Wieczorek

Examples
data(UPSpep25)
mvFilter(UPSpep25, "wholeMatrix", 2)
mvFilterFromIndices  Filter lines in the matrix of intensities w.r.t. some criteria

Description

Filters the lines of `exprs()` table with conditions on the number of missing values. The user chooses the minimum amount of intensities that is acceptable and the filter delete lines that do not respect this condition. The condition may be on the whole line or condition by condition.

Usage

```
mvFilterFromIndices(obj, keepThat = NULL, processText = NULL)
```

Arguments

- `obj`: An object of class `MSnSet` containing quantitative data.
- `keepThat`: A vector of integers which are the indices of lines to keep.
- `processText`: A string to be included in the `MSnSet` object for log.

Details

The different methods are: "wholeMatrix": given a threshold `th`, only the lines that contain at least `th` values are kept. "allCond": given a threshold `th`, only the lines which contain at least `th` values for each of the conditions are kept. "atLeastOneCond": given a threshold `th`, only the lines that contain at least `th` values, and for at least one condition, are kept.

Value

An instance of class `MSnSet` that have been filtered.

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```
data(UPSpep25)
mvFilter(UPSpep25, c(1:10))
```

mvFilterGetIndices  Filter lines in the matrix of intensities w.r.t. some criteria

Description

Returns the indices of the lines of `exprs()` table to delete w.r.t. the conditions on the number of missing values. The user chooses the minimum amount of intensities that is acceptable and the filter delete lines that do not respect this condition. The condition may be on the whole line or condition by condition.
Usage

`mvFilterGetIndices(obj, type, th)`

Arguments

- **obj**: An object of class `MSnSet` containing quantitative data.
- **type**: Method used to choose the lines to delete. Values are: "none", "wholeMatrix", "allCond", "atLeastOneCond"
- **th**: An integer value of the threshold

Details

The different methods are: "wholeMatrix": given a threshold th, only the lines that contain at least th values are kept. "allCond": given a threshold th, only the lines which contain at least th values for each of the conditions are kept. "atLeastOneCond": given a threshold th, only the lines that contain at least th values, and for at least one condition, are kept.

Value

An vector of indices that correspond to the lines to keep.

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```
data(UPS pep 25)
mvFilterGetIndices(UPS pep 25, "wholeMatrix", 2)
```

---

`mvHisto`  
*Histogram of missing values*

Description

This method plots a histogram of missing values.

Usage

`mvHisto(qData, samplesData, labels, indLegend = "auto", showValues = FALSE)`

Arguments

- **qData**: A dataframe that contains quantitative data.
- **samplesData**: A dataframe where lines correspond to samples and columns to the meta-data for those samples.
- **labels**: A vector of the conditions (labels) (one label per sample).
- **indLegend**: The indices of the column name’s in `pData()` tab
- **showValues**: A logical that indicates wether numeric values should be drawn above the bars.
Value

A histogram

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
samplesData <- Biobase::pData(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
mvHisto(qData, samplesData, labels, indLegend="auto", showValues=TRUE)
```

Description

Plots a heatmap of the quantitative data. Each column represent one of the conditions in the object of class `MSnSet` and the color is proportional to the mean of intensity for each line of the dataset. The lines have been sorted in order to visualize easily the different number of missing values. A white square is plotted for missing values.

Usage

```r
mvImage(qData, labels)
```

Arguments

- **qData**: A dataframe that contains quantitative data.
- **labels**: A vector of the conditions (labels) (one label per sample).

Value

A heatmap

Author(s)

Samuel Wieczorek, Thomas Burger

Examples

```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
layers <- Biobase::pData(UPSpep25)[,"Label"]
mvImage(qData, labels)
```
mvImputation

Missing values imputation from a matrix

Description
This method is a wrapper to the imputeLCMD package adapted to a matrix.

Usage
mvImputation(qData, method)

Arguments
qData A dataframe that contains quantitative data.
method The imputation method to be used. Choices are QRILC, KNN, BPCA and MLE.

Value
The matrix imputed

Author(s)
Samuel Wieczorek

Examples
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
mvImputation(qData, "QRILC")

mvPerLinesHisto

Bar plot of missing values per lines

Description
This method plots a bar plot which represents the distribution of the number of missing values (NA) per lines (ie proteins).

Usage
mvPerLinesHisto(qData, samplesData, indLegend = "auto", showValues = FALSE)

Arguments
qData A dataframe that contains the data to plot.
samplesData A dataframe which contains informations about the replicates.
indLegend The indice of the column name’s in pData() tab
showValues A logical that indicates wether numeric values should be drawn above the bars.
Value
A bar plot

Author(s)
Florence Combes, Samuel Wieczorek

Examples
```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
samplesData <- Biobase::pData(UPSpep25)
mvPerLinesHisto(qData, samplesData)
```

mvPerLinesHistoPerCondition

**Bar plot of missing values per lines and per condition**

Description
This method plots a bar plot which represents the distribution of the number of missing values (NA) per lines (ie proteins) and per conditions.

Usage
```r
mvPerLinesHistoPerCondition(qData, samplesData, indLegend = "auto",
                          showValues = FALSE)
```

Arguments
- `qData` A dataframe that contains quantitative data.
- `samplesData` A dataframe where lines correspond to samples and columns to the meta-data for those samples.
- `indLegend` The indice of the column name’s in pData() tab
- `showValues` A logical that indicates wether numeric values should be drawn above the bars.

Value
A bar plot

Author(s)
Samuel Wieczorek

Examples
```r
data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
samplesData <- Biobase::pData(UPSpep25)
mvPerLinesHistoPerCondition(qData, samplesData)
```
mvTypePlot

Distribution of missing values with respect to intensity values

Description

This method plots a scatter plot which represents the distribution of missing values. The colors correspond to the different conditions (slot Label in in the dataset of class MSnSet). The x-axis represent the mean of intensity for one condition and one entity in the dataset (i. e. a protein) whereas the y-axis count the number of missing values for this entity and the considered condition. The data have been jittered for an easier visualization.

Usage

mvTypePlot(qData, labels, threshold = 0)

Arguments

qData A dataframe that contains quantitative data.
labels A vector of the conditions (labels) (one label per sample).
threshold An integer for the intensity that delimits MNAR and MCAR missing values.

Value

A scatter plot

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
mvTypePlot(qData, labels, threshold=0)

normalizeD

Normalisation

Description

Provides several methods to normalize data from a matrix. They are organized in four main families: Strong Rescaling, Median Centering, Mean Centering, Mean CenteringScaling. For the first family, two sub-categories are available: the sum by columns and the quantiles method. For the three other families, two categories are available: "Overall" which means that the value for each protein (ie line in the expression data tab) is computed over all the samples; "within conditions" which means that the value for each protein (ie line in the matrix) is computed condition by condition.

Usage

normalizeD(qData, labels, family, method)
pepAgregate

Arguments

qData     A dataframe that contains quantitative data.
labels    A vector of strings containing the column "Label" of the pData().
family One of the following : Global Rescaling, Median Centering, Mean Centering, Mean Centering Scaling.
method    "Overall" or "within conditions".

Value

A matrix normalized

Author(s)

Florence Combes, Samuel Wieczorek

Examples

data(UPSpep25)
qData <- Biobase::exprs(UPSpep25)
labels <- Biobase::pData(UPSpep25)[, "Label"]
normalizeD(qData, labels, "Median Centering", "within conditions")

pepAgregate  Function agregate peptides to proteins

Description

Method to agregate with a method peptides to proteins on a MSnSet object (peptides)

Usage

pepAgregate(obj.pep, protID, method = "sum overall", matAdj = NULL, 
             n = NULL)

Arguments

obj.pep     An object (peptides) of class MSnbase.
protID The name of proteins ID column
method The method used to aggregate the peptides into proteins. Values are "sum", "mean" or "sum on top n" : do the sum / mean of intensity on all peptides belonging to proteins. Default is "sum"
matAdj An adjacency matrix
n The number of peptides considered for the aggregation.

Value

An object of class MSnbase with proteins
**proportionConRev**

**Author(s)**
Alexia Dorffer, Samuel Wieczorek

**Examples**
```
data(UPSpep25)
protID <- "Protein.group.IDs"
mat <- BuildAdjacencyMatrix(UPSpep25, protID, TRUE)
pepAgregate(UPSpep25, protID, "sum overall", mat)
```

---

**Description**
Plots a barplot of proportion of contaminants and reverse

**Usage**
```
proportionConRev(obj, idContaminants = NULL, prefixContaminants = NULL, idReverse = NULL, prefixReverse = NULL)
```

**Arguments**
- **obj**: An object of class `MSnSet`.
- **idContaminants**: The name of a column of Contaminants
- **prefixContaminants**: The prefix to identify contaminants
- **idReverse**: The name of a column of Reverse
- **prefixReverse**: The name of a column of Reverse

**Value**
A barplot

**Author(s)**
Samuel Wieczorek

**Examples**
```
data(UPSpep25)
pref <- "*
proportionConRev(UPSpep25, "Potential.contaminant", pref, "Reverse", pref)
```
removeLines

Removes lines in the dataset based on a prefix string.

Description
This function removes lines in the dataset based on a prefix string.

Usage
removeLines(obj, idLine2Delete = NULL, prefix = NULL)

Arguments
- obj: An object of class MSnSet.
- idLine2Delete: The name of the column that correspond to the data to filter.
- prefix: A character string that is the prefix to find in the data.

Value
An object of class MSnSet.

Author(s)
Samuel Wieczorek

Examples
data(UPSpep25)
removeLines(UPSpep25, "Potential.contaminant")
removeLines(UPSpep25, "Reverse")

SumPeptides
Compute the intensity of proteins with the sum of the intensities of their peptides.

Description
This function computes the intensity of proteins based on the sum of the intensities of their peptides.

Usage
SumPeptides(matAdj, expr)

Arguments
- matAdj: An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
- expr: A matrix of intensities of peptides.
test

Value

A matrix of intensities of proteins

Author(s)

Alexia Dorffer

Examples

data(UPSpep25)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
SumPeptides(M, Biobase::exprs(UPSpep25))

test

Test dataset

Description

Partial (small) dataset for unit tests containing missing values.

Format

An object of class MSnSet

testWithoutNA

Test dataset

Description

Partial (small) dataset for unit tests without any missing values.

Format

An object of class MSnSet
TopnPeptides  
Compute the intensity of proteins as the sum of the intensities of their $n$ best peptides.

Description
This function computes the intensity of proteins as the sum of the intensities of their $n$ best peptides.

Usage
TopnPeptides(matAdj, expr, n)

Arguments
- `matAdj`: An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
- `expr`: A matrix of intensities of peptides
- `n`: The maximum number of peptides used to aggregate a protein.

Value
A matrix of intensities of proteins

Author(s)
Alexia Dorffer

Examples
```r
data(UPSpep25)
protID <- "Protein.group.IDs"
matAdj <- BuildAdjacencyMatrix(UPSpep25, protID, FALSE)
TopnPeptides(matAdj, Biobase::exprs(UPSpep25), 3)
```

UPSpep25  
**UPSpep25 dataset**

Description
This dataset is the final outcome of a quantitative mass spectrometry-based proteomic analysis of two samples containing different concentrations of 48 human proteins (UPS1 standard from Sigma-Aldrich) within a constant yeast background (see Giai Gianetto et al. (2016) for details). It contains the abundance values of the different human and yeast peptides identified and quantified in these two conditions. The two conditions represent the measured abundances of peptides when respectively 25fmol and 10fmol of UPS1 human proteins were mixed with the yeast extract before mass spectrometry analyses. Three technical replicates were acquired for each condition.

To identify and quantify peptides, spectra were searched using MaxQuant (version 1.5.1.2) against the Uniprot database, the UPS database and the frequently observed contaminants database. Maximum false discovery rates were set to 0.01 by employing a reverse database strategy.

The dataset is either available as a CSV file (see inst/extdata/UPSpep25.txt), or as a MSnSet structure (UPSpep25). In the latter case, the quantitative data are those of the raw intensities.
Usage

data(UPSpep25)

Format

An object of class `MSnSet` related to peptide quantification. It contains 6 samples divided into two conditions (25fmol and 10fmol) and 13918 peptides.
The data frame `exprs(UPSpep25)` contains six columns that are the quantitation of peptides for the six replicates.
The data frame `fData(UPSpep25)` contains the meta data about the peptides.
The data frame `pData(UPSpep25)` contains the experimental design and gives few informations about the samples.

Value

An object of class `MSnSet`.

References


---

`varianceDistD`  
*Distribution of variance of proteins*

Description

Builds a densityplot of the variance of entities in the `exprs()` table of a object. The variance is calculated for each condition (Label) present in the dataset (see the slot `Label` in the `pData()` table)

Usage

`varianceDistD(qData, labels = NULL)`

Arguments

- `qData`: A dataframe that contains quantitative data.
- `labels`: A vector of the conditions (labels) (one label per sample).

Value

A density plot

Author(s)

Florence Combes, Samuel Wieczorek
violinPlotD

See Also

densityPlotD.

Examples

data(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
varianceDistD(UPSpep25)

violinPlotD
Builds a violinplot from a dataframe

Description

ViolinPlot for quantitative proteomics data

Usage

violinPlotD(qData, dataForXAxis = NULL, labels = NULL, group2Color = "Condition")

Arguments

qData A dataframe that contains quantitative data.
dataForXAxis A vector containing the types of replicates to use as X-axis. Available values are: Label, Analyt.Rep, Bio.Rep and Tech.Rep. Default is "Label".
labels A vector of the conditions (labels) (one label per sample).
group2Color A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

Value

A violinplot

Author(s)

Florence Combes, Samuel Wieczorek

See Also
densityPlotD

Examples

data(UPSpep25)
library(vioplot)
qData <- Biobase::exprs(UPSpep25)
types <- c("Label","Analyt.Rep")
dataForXAxis <- Biobase::pData(UPSpep25)[,types]
labels <- Biobase::pData(UPSpep25)[,"Label"]
violinPlotD(qData, dataForXAxis, labels)
**wrapper.boxPlotD**

*Wrapper to the boxplotD function on an object MSnSet*

**Description**

This function is a wrapper for using the boxplotD function with objects of class `MSnSet`.

**Usage**

```r
wrapper.boxPlotD(obj, dataForXAxis = "Label", group2Color = "Condition")
```

**Arguments**

- `obj`: An object of class `MSnSet`.
- `dataForXAxis`: A vector of strings containing the names of columns in `pData()` to print labels on X-axis (Default is "Label").
- `group2Color`: A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

**Value**

A boxplot

**Author(s)**

Florence Combes, Samuel Wieczorek

**See Also**

`wrapper.densityPlotD`

**Examples**

```r
data(UPSpep25)
types <- c("Label", "Analyt.Rep")
wrapper.boxPlotD(UPSpep25, types)
```

---

**wrapper.compareNormalizationD**

*Builds a plot from a dataframe*

**Description**

Wrapper to the function that plot to compare the quantitative proteomics data before and after normalization.

**Usage**

```r
wrapper.compareNormalizationD(objBefore, objAfter, labelsForLegend = NULL, indData2Show = NULL, group2Color = "Condition")
```
wrapper.corrMatrixD

Arguments

- **objBefore**: A dataframe that contains quantitative data before normalization.
- **objAfter**: A dataframe that contains quantitative data after normalization.
- **labelsForLegend**: A vector of the conditions (labels) (one label per sample).
- **indData2Show**: A vector of the indices of the columns to show in the plot. The indices are those of indices of the columns in the dataframe qDataBefore.
- **group2Color**: A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

Value

A plot

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
objAfter <- wrapper.normalizeD(UPSpep25, "Median Centering", "within conditions")
wrapper.compareNormalizationD(UPSpep25, objAfter, labels)

wrapper.corrMatrixD

Displays a correlation matrix of the quantitative data of the exprs() table

Description

Builds a correlation matrix based on a MSnSet object.

Usage

wrapper.corrMatrixD(obj, rate = 5)

Arguments

- **obj**: An object of class MSnSet.
- **rate**: A float that defines the gradient of colors.

Value

A colored correlation matrix

Author(s)

Alexia Dorffer
Examples

```r
data(UPSpep25)
wrapper.corrMatrixD(UPSpep25)
```

```r
wrapper.densityPlotD(UPSpep25, labels)
```

**Description**

This function is a wrapper for using the densityPlotD function with objects of class `MSnSet`.

**Usage**

```r
wrapper.densityPlotD(obj, labelsForLegend = NULL, indData2Show = NULL, group2Color = "Condition")
```

**Arguments**

- `obj`: An object of class `MSnSet`.
- `labelsForLegend`: A vector of labels to show in densityplot.
- `indData2Show`: A vector of the indices of the columns to show in the plot. The indices are those of indices of the columns int the data frame `qDataBefore` in the density plot.
- `group2Color`: A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

**Value**

A density plot

**Author(s)**

Alexia Dorffer

**See Also**

`wrapper.boxPlotD, wrapper.varianceDistD`

**Examples**

```r
data(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
wrapper.densityPlotD(UPSpep25, labels)
```
wrapper.diffAnaLimma  

Performs differential analysis on an MSnSet object, calling the limma package functions.

Description

Method to perform differential analysis on a MSnSet object (calls the limma package function).

Usage

wrapper.diffAnaLimma(obj, condition1, condition2)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>An object of class MSnSet.</td>
</tr>
<tr>
<td>condition1</td>
<td>A vector that contains the names of the conditions considered as condition 1.</td>
</tr>
<tr>
<td>condition2</td>
<td>A vector that contains the names of the conditions considered as condition 2.</td>
</tr>
</tbody>
</table>

Value

A dataframe as returned by the limma package

Author(s)

Alexia Dorffer

Examples

data(UPSpep25)
condition1 <- '25fmol'
condition2 <- '10fmol'
wrapper.diffAnaLimma(UPSpep25, condition1, condition2)

wrapper.diffAnaWelch  

Performs a differential analysis on a MSnSet object using the Welch t-test.

Description

Computes differential analysis on a MSnSet object, using the Welch t-test (t.test(stats)).

Usage

wrapper.diffAnaWelch(obj, condition1, condition2)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>An object of class MSnSet.</td>
</tr>
<tr>
<td>condition1</td>
<td>A vector containing the names of the conditions considered as condition 1.</td>
</tr>
<tr>
<td>condition2</td>
<td>A vector containing the names of the conditions considered as condition 2.</td>
</tr>
</tbody>
</table>
Value

A dataframe with two slots: P.Value (for the p-value) and logFC (the log of the Fold Change).

Author(s)

Alexia Dorffer

Examples

data(UPSep25)
condition1 <- '25fmol'
condition2 <- '10fmol'
wrapper.diffAnaWelch(UPSep25, condition1, condition2)

wrapper.heatmapD

This function is a wrapper to heatmap.2 that displays quantitative data in the exprs() table of an object of class MSnSet

Description

Builds a heatmap of the quantitative proteomic data of a MSnSet object.

Usage

wrapper.heatmapD(obj, distance = "euclidean", cluster = "average", dendro = FALSE)

Arguments

obj An object of class MSnSet.
distance The distance used by the clustering algorithm to compute the dendrogram. See help(heatmap.2).
cluster the clustering algorithm used to build the dendrogram. See help(heatmap.2)
dendro A boolean to indicate if the dendrogram has to be displayed

Value

A heatmap

Author(s)

Alexia Dorffer

Examples

data(testWithoutNA)
wrapper.heatmapD(testWithoutNA)
wrapper.mvHisto  

*Histogram of missing values from a MSnSet object*

**Description**

This method plots from a MSnSet object a histogram of missing values.

**Usage**

```r
wrapper.mvHisto(obj, indLegend = "auto", showValues = FALSE)
```

**Arguments**

- `obj`: An object of class MSnSet.
- `indLegend`: The indices of the column name’s in pData() tab.
- `showValues`: A logical that indicates whether numeric values should be drawn above the bars.

**Value**

A histogram

**Author(s)**

Alexia Dorffer

**Examples**

```r
data(UPSpep25)
wrapper.mvHisto(UPSpep25, showValues = TRUE)
```

---

wrapper.mvImage  

*Heatmap of missing values from a MSnSet object*

**Description**

Plots a heatmap of the quantitative data. Each column represent one of the conditions in the object of class MSnSet and the color is proportional to the mean of intensity for each line of the dataset. The lines have been sorted in order to visualize easily the different number of missing values. A white square is plotted for missing values.

**Usage**

```r
wrapper.mvImage(obj)
```

**Arguments**

- `obj`: An object of class MSnSet.

**Value**

A heatmap
**Author(s)**
Alexia Dorffer

**Examples**
```
data(UPSpep25)
wrapper.mvImputation(UPSpep25)
```

---

**Description**
This method is a wrapper to the `imputeLCMD` package adapted to objects of class `MSnSet`.

**Usage**
```
wrapper.mvImputation(obj, method)
```

**Arguments**
- `obj` An object of class `MSnSet`.
- `method` The imputation method to be used. Choices are QRILC, KNN, BPCA and MLE.

**Value**
The object `obj` which has been imputed

**Author(s)**
Alexia Dorffer

**Examples**
```
data(UPSpep25)
wrapper.mvImputation(UPSpep25, "QRILC")
```

---

**Description**
This method is a wrapper to plots from a `MSnSet` object a histogram which represents the distribution of the number of missing values (NA) per lines (ie proteins).

**Usage**
```
wrapper.mvPerLinesHisto(obj, indLegend = "auto", showValues = FALSE)
```
wrapper.mvPerLinesHistoPerCondition

Arguments

obj          An object of class MSnSet.
indLegend    The indice of the column name’s in pData() tab.
showValues   A logical that indicates wether numeric values should be drawn above the bars.

Value

A histogram

Author(s)

Alexia Dorffer

Examples

data(UPSpep25)
wrapper.mvPerLinesHistoPerCondition(UPSpep25)

---

wrapper.mvPerLinesHistoPerCondition

*Bar plot of missing values per lines and per conditions from an object MSnSet*

Description

This method is a wrapper to plots from a MSnSet object a bar plot which represents the distribution of the number of missing values (NA) per lines (ie proteins) and per conditions.

Usage

wrapper.mvPerLinesHistoPerCondition(obj, indLegend = "auto",
showValues = FALSE)

Arguments

obj          An object of class MSnSet.
indLegend    The indice of the column name’s in pData() tab.
showValues   A logical that indicates wether numeric values should be drawn above the bars.

Value

A bar plot

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
wrapper.mvPerLinesHistoPerCondition(UPSpep25)
**wrapper.mvTypePlot**

*Distribution of missing values with respect to intensity values from a MSnSet object*

**Description**

This method plots a scatter plot which represents the distribution of missing values. The colors correspond to the different conditions (slot `Label` in in the dataset of class `MSnSet`). The x-axis represent the mean of intensity for one condition and one entity in the dataset (i.e. a protein) whereas the y-axis count the number of missing values for this entity and the considered condition. The data have been jittered for an easier visualisation.

**Usage**

```r
wrapper.mvTypePlot(obj, threshold = 0)
```

**Arguments**

- `obj` An object of class `MSnSet`.
- `threshold` An integer for the intensity that delimits MNAR and MCAR missing values.

**Value**

A scatter plot

**Author(s)**

Florence Combes, Samuel Wieczorek

**Examples**

```r
data(UPS Pep25)
wrapper.mvTypePlot(UPS pep25)
```

---

**wrapper.normalizeD**

*Normalisation*

**Description**

Provides several methods to normalize quantitative data from a `MSnSet` object. They are organized in four main families: Strong Rescaling, Median Centering, Mean Centering, Mean Centering Scaling. For the first family, two sub-categories are available: the sum by columns and the quantiles method. For the three other families, two categories are available: "Overall" which means that the value for each protein (i.e line in the expression data tab) is computed over all the samples; "within conditions" which means that the value for each protein (i.e line in the `exprs()` data tab) is computed condition by condition.

**Usage**

```r
wrapper.normalizeD(obj, family, method)
```
Arguments

- **obj**: An object of class `MSnSet`.
- **family**: One of the following: Global Rescaling, Median Centering, Mean Centering, Mean Centering Scaling.
- **method**: "Overall" or "within conditions".

Value

An instance of class `MSnSet` where the quantitative data in the `exprs()` tab has been normalized.

Author(s)

Alexia Dorffer

Examples

data(UPSpep25)
wrapper.normalizeD(UPSpep25, "Median Centering", "within conditions")

---

**wrapper.varianceDistD**  
*Distribution of variance of proteins*

Description

Builds a densityplot of the variance of entities in the `exprs()` table of an object `MSnSet`. The variance is calculated for each condition (Label) present in the dataset (see the slot 'Label' in the `pData()` table).

Usage

`wrapper.varianceDistD(obj)`

Arguments

- **obj**: An object of class `MSnSet`.

Value

A density plot

Author(s)

Alexia Dorffer

See Also

`wrapper.densityPlotD`

Examples

data(UPSpep25)
wrapper.varianceDistD(UPSpep25)
wrapper.violinPlotD

Wrapper to the violinPlotD function on an object MSnSet

Description
This function is a wrapper for using the violinPlotD function with objects of class MSnSet.

Usage
wrapper.violinPlotD(obj, dataForXAxis = "Label", group2Color = "Condition")

Arguments
- obj: An object of class MSnSet.
- dataForXAxis: A vector of strings containing the names of columns in pData() to print labels on X-axis (Default is "Label").
- group2Color: A string that indicates how to color the replicates: one color per condition (value "Condition") or one color per replicate (value "Replicate"). Default value is by Condition.

Value
A violin plot

Author(s)
Samuel Wieczorek

See Also
wrapper.densityPlotD, wrapper.boxPlotD

Examples
data(UPSpep25)
library(vioplot)
types <- c("Label","Analyt.Rep")
wrapper.violinPlotD(UPSpep25, types)

wrapperCalibrationPlot
Performs a calibration plot on an MSnSet object, calling the cp4p package functions.

Description
This function is a wrapper to the calibration.plot method of the cp4p package for use with MSnSet objects.
Usage

wrapperCalibrationPlot(vPVal, pi0Method = "pounds")

Arguments

vPVal       A dataframe that contains quantitative data.
pi0Method   A vector of the conditions (labels) (one label per sample).

Value

A plot

Author(s)

Samuel Wieczorek

Examples

data(UPSpep25)
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(UPSpep25)
labels <- Biobase::pData(UPSpep25)[,"Label"]
diffAnaWelch(qData, labels, condition1, condition2)

writeMSnsetToExcel

This function exports a MSnSet object to a Excel file.

Description

This function exports a MSnSet data object to a Excel file. Each of the three data.frames in the MSnSet object (ie experimental data, phenoData and metaData are respectively integrated into separate sheets in the Excel file).

Usage

writeMSnsetToExcel(obj, filename)

Arguments

obj       An object of class MSnSet.
filename  A character string for the name of the Excel file.

Value

A Excel file (.xlsx)

Author(s)

Samuel Wieczorek
Examples

data(UPS.pep25)
write.MSnsetToExcel(UPS.pep25, "foo")
Index

*Topic datasets
  USPsep25, 36

*Topic data
  test, 35
  testWithoutNA, 35
  USPsep25, 36

boxPlotD, 3, 9
BuildAdjacencyMatrix, 4
BuildColumnToProteinDataset, 4
compareNormalizationD, 5
corrMatrixD, 6
CountPep, 7
createMSnset, 7
deleteLinesFromIndices, 8
densityPlotD, 3, 9, 38
diffAna, 10, 11, 13
diffAnaComputeFDR, 10
diffAnaGetSignificant, 11
diffAnaLimma, 12
diffAnaSave, 13
diffAnaVolcanoplot, 14
diffAnaWelch, 15
diffIndicesConditions, 15
diffIndicesOfLinesToRemove, 16
diffNumberOf, 17
diffNumberOfEmptyLines, 17
getPaletteForLabels, 18
getPaletteForReplicates, 19
getPercentageOfMV, 19
getProcessingInfo, 20
getProteinsStats, 20
GraphPepProt, 21
heatmap, 2, 22, 23, 43
heatmap.DAPAR, 22
heatmapD, 23
limma, 10, 13
limmaCompleteTest, 23
MeanPeptides, 24
MSnbase, 4, 20, 32
mvFilter, 25
mvFilterFromIndices, 26
mvFilterGetIndices, 26
mvHisto, 27
mvImage, 28
mvImputation, 29
mvPerLinesHisto, 29
mvPerLinesHistoPerCondition, 30
mvTypePlot, 31
normalizeD, 31
pepAggregate, 32
proportionConRev, 33
RColorBrewer, 18, 19
removeLines, 34
SumPeptides, 34
t.test, 15, 42
test, 35
testWithoutNA, 35
TopnPeptides, 36
UPSepep25, 36

varianceDistD, 9, 37
violinPlotD, 38
wrapper.boxPlotD, 39, 41, 49
wrapper.compareNormalizationD, 39
wrapper.corrMatrixD, 40
wrapper.densityPlotD, 39, 41, 48, 49
wrapper.diffAnaLimma, 42
wrapper.diffAnaWelch, 42
wrapper.heatmapD, 43
wrapper.mvHisto, 44
wrapper.mvImage, 44
wrapper.mvImputation, 45
wrapper.mvPerLinesHisto, 45
INDEX

wrapper.mvPerLinesHistoPerCondition, 46
wrapper.mvTypePlot, 47
wrapper.normalizeD, 47
wrapper.varianceDistD, 47, 48
wrapper.violinPlotD, 49
wrapperCalibrationPlot, 49
writeMSNsetToExcel, 50