# Package 'DAPAR'

October 17, 2017

**Type** Package

**Title** Tools for the Differential Analysis of Proteins Abundance with R

**Version** 1.8.7

**Date** 2017-05-29

**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 (>= 3.4)

**Suggests** BiocGenerics, Biobase, testthat, BiocStyle, Prostar

**Imports** MSnbase, RColorBrewer,stats,preprocessCore,Cairo,png, lattice,reshape2,gplots,pcaMethods,ggplot2, limma,knitr,tmvtnorm,norm,impute, imputeLCMD, doParallel, parallel, foreach,grDevices, graphics, openxlsx, utils, cp4p (>= 0.3.5), scales, Matrix, vioplot, imp4p (>= 0.3), highcharter, DAPARdata

**biocViews** Proteomics, Normalization, Preprocessing, MassSpectrometry, QualityControl, DataImport

**NeedsCompilation** no

**RoxygenNote** 6.0.1

## R topics documented:

---

`boxPlotD` *Builds a boxplot from a dataframe*

---

## Description

Boxplot for quantitative proteomics data

## Usage

```
boxPlotD(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 boxplot

## Author(s)

Florence Combes, Samuel Wieczorek

## See Also

`densityPlotD`

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
types <- c("Label","Analyt.Rep")
dataForXAxis <- Biobase::pData(Exp1_R25_pept)[,types]
labels <- Biobase::pData(Exp1_R25_pept)[,"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

```
require(DAPARdata)
data(Exp1_R25_pept)
BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], "Protein.group.IDs", TRUE)
```

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

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| peptideData | A data.frame of meta data of peptides. It is the fData of the MSnset object. |
| matAdj | The adjacency matrix used to agregate 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. |
| proteinNames | The names of the protein in the new dataset (i.e. rownames) |

## Value

A vector

## Author(s)

Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, FALSE)
data <- Biobase::fData(Exp1_R25_pept[1:1000])
protData <- pepAgregate(Exp1_R25_pept[1:1000], 'Protein_group_IDs', 'sum overall', M)
name <- "organism"
proteinNames <- rownames(Biobase::fData(protData))
BuildColumnToProteinDataset(data, M, name,proteinNames )
```

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

```
require(DAPARdata)
data(Exp1_R25_pept)
qDataBefore <- Biobase::exprs(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
qDataAfter <- normalizeD(qDataBefore,labels,"Median Centering",
"within conditions")
compareNormalizationD(qDataBefore, qDataAfter, labels)
```

## corrMatrixD — *Displays a correlation matrix of the quantitative data of the* `exprs()` *table.*

### 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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
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

```
library(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], 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

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

## 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. |
| indFData | The name of column in file that will be the name of rows for the exprs() and fData() tables |
| 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

```
require(DAPARdata)
exprsFile <- system.file("extdata", "Exp1_R25_pept.txt",
package="DAPARdata")
metadataFile <- system.file("extdata", "samples_Exp1_R25.txt",
package="DAPARdata")
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")
```

---

# CVDistD *Distribution of CV of entities*

---

## Description

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

## Usage

```
CVDistD(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

## See Also

`densityPlotD`.

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
CVDistD(Biobase::exprs(Exp1_R25_pept), labels)
```

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

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

## 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. |

## Value

An instance of class MSnSet that have been filtered.

## Author(s)

Florence Combes, Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
deleteLinesFromIndices(Exp1_R25_pept, c(1:10))
```

densityPlotD *Builds a densityplot 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, CVDistD

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
labels <- lab2Show <- Biobase::pData(Exp1_R25_pept)[,"Label"]
densityPlotD(qData, labels)
```

---

## diffAna — *This function performs a differential analysis on an MSnSet object (adapted from limma)*

---

### Description

Performs a differential analysis on an MSnSet object, based on limma functions.

### Usage

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
design <- cbind(cond1=1,
cond2 = rep(0,nrow(Biobase::pData(Exp1_R25_pept[1:1000]))))
rownames(design) <- rownames(Biobase::pData(Exp1_R25_pept[1:1000]))
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"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 wrappper 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

```
diffAnaComputeFDR(data, threshold_PVal = 0, threshold_LogFC = 0,
  pi0Method = 1)
```

### Arguments

| | |
|---|---|
| data | The result of the differential analysis processed by diffAna |
| threshold_PVal | The threshold on p-pvalue 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

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- wrapper.mvImputation(Exp1_R25_pept[1:1000], "QRILC")
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

```
diffAnaGetSignificant(obj)
```

## Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |

## Value

A MSnSet

## Author(s)

Alexia Dorffer

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- "25fmol"
condition2 <- "10fmol"
resLimma <- wrapper.diffAnaLimma(Exp1_R25_pept[1:1000],
condition1, condition2)
obj <-diffAnaSave(Exp1_R25_pept[1:1000], resLimma, "limma",
condition1, condition2)
signif <- diffAnaGetSignificant(obj)
```

`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

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
samplesData <- Biobase::pData(Exp1_R25_pept[1:1000])
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"Label"]
diffAnaLimma(qData, samplesData, labels, condition1, condition2)
```

`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 choosen to discriminate differential proteins. |
| `threshold_logFC` | A float that indicates the threshold on log(Fold Change) to discriminatedifferential 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

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
limma <- wrapper.diffAnaLimma(Exp1_R25_pept[1:1000],
condition1, condition2)
obj <- diffAnaSave(Exp1_R25_pept[1:1000], 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

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
data <- wrapper.diffAnaLimma(Exp1_R25_pept[1:1000], condition1, condition2)
diffAnaVolcanoplot(data$logFC, data$P_Value)
```

diffAnaVolcanoplot_rCharts

*Volcanoplot of the differential analysis*

## Description

Plots an interactive 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. With the use of the package Highcharter, a customizable tooltip appears when the user put the mouse's pointer over a point of the scatter plot.

## Usage

```
diffAnaVolcanoplot_rCharts(df, threshold_pVal = 1e-60, threshold_logFC = 0,
  conditions = NULL, clickFunction = NULL)
```

## Arguments

| | |
|---|---|
| `df` | A dataframe which contains the following slots : x : a vector of the log(fold change) values of the differential analysis, y : a vector of the p-value values returned by the differential analysis. index : a vector of the rowanmes of the data. This dataframe must has been built with the option stringsAsFactors set to FALSE. There may be additional slots which will be used to show informations in the tooltip. The name of these slots must begin with the prefix "tooltip_". It will be automatically removed in the plot. |
| `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. |
| `clickFunction` | A string that contains a JavaScript function used to show info from slots in df. The variable this.index refers to the slot named index and allows to retrieve the right row to show in the tooltip |

## Value

An interactive volcanoplot

## Author(s)

Samuel Wieczorek

## Examples

```
library(highcharter)
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
```

```
condition1 <- '25fmol'
condition2 <- '10fmol'
cond <- c(condition1, condition2)
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', '6')
obj <- mvFilterFromIndices(obj, keepThat,
       'Filtered with wholeMatrix (threshold = 6 ).')
data <- wrapper.diffAnaLimma(obj, condition1, condition2)
df <- data.frame(x=data$logFC,
y = -log10(data$P_Value),
index = as.character(rownames(obj)),
stringsAsFactors = FALSE)
tooltipSlot <- c("Sequence", "Score")
df <- cbind(df,Biobase::fData(obj)[tooltipSlot])
colnames(df) <- gsub(".", "_", colnames(df), fixed=TRUE)
if (ncol(df) > 3){
    colnames(df)[4:ncol(df)] <-
    paste("tooltip_", colnames(df)[4:ncol(df)], sep="")
}
hc_clickFunction <- JS("function(event) {
Shiny.onInputChange('eventPointClicked', [this.index]);}")
diffAnaVolcanoplot_rCharts(df,threshold_logFC = 1,
threshold_pVal = 3,
conditions = cond,
clickFunction=hc_clickFunction)
```

---

`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{stats}`).

## Usage

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"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

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"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

```
require(DAPARdata)
data(Exp1_R25_pept)
getIndicesOfLinesToRemove(Exp1_R25_pept, "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

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
getNumberOf(Exp1_R25_pept, "Potential.contaminant", "+")
```

---

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

---

## Description

Returns the number of empty lines in a matrix.

## Usage

```
getNumberOfEmptyLines(qData)
```

## Arguments

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

## Value

An integer

## Author(s)

Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
getNumberOfEmptyLines(qData)
```

## getPaletteForLabels *Palette for plots in DAPAR*

### 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

```
getPaletteForLabels(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

```
require(DAPARdata)
data(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
getPaletteForLabels(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

```
require(DAPARdata)
data(Exp1_R25_pept)
n <- nrow(Biobase::pData(Exp1_R25_pept))
getPaletteForReplicates(n)
```

---

## 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

### Author(s)

Florence Combes, Samuel Wieczorek

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
getPourcentageOfMV(Exp1_R25_pept)
```

`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

```
getProcessingInfo(obj)
```

## Arguments

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

## Value

The slot processing of obj@processingData

## Author(s)

Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
getProcessingInfo(Exp1_R25_pept)
```

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

## Description

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

## Usage

```
getProteinsStats(matUnique, matShared)
```

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
MShared <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, FALSE)
MUnique <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], 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

```
require(DAPARdata)
data(Exp1_R25_pept)
mat <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], "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

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- mvFilter(Exp1_R25_pept, "wholeMatrix", 6)
qData <- Biobase::exprs(obj)
heatmap.DAPAR(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

```
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 fi the dendrogram has to be displayed |

## Value

A heatmap

## Author(s)

Florence Combes, Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- mvFilter(Exp1_R25_pept[1:1000], "wholeMatrix", 6)
qData <- Biobase::exprs(obj)
heatmapD(qData)
```

impute.pa2 *Missing values imputation from a* MSnSet *object*

## Description

This method is a variation to the function impute.pa from the package imp4p.

## Usage

```
impute.pa2(tab, conditions, q.min = 0, q.norm = 3, eps = 0,
  distribution = "unif")
```

### Arguments

| | |
|---|---|
| tab | An object of class MSnSet. |
| conditions | xxxxxxxxxxxx |
| q.min | A quantile value of the observed values allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0 (the maximal value is the minimum of observed values minus eps). |
| q.norm | A quantile value of a normal distribution allowing defining the minimal value which can be generated. Default is 3 (the minimal value is the maximal value minus qn*median(sd(observed values)) where sd is the standard deviation of a row in a condition). |
| eps | A value allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0. |
| distribution | The type of distribution used. Values are unif or beta. |

### Value

The object obj which has been imputed

### Author(s)

Thomas Burger, Samuel Wieczorek

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.impute.pa2(Exp1_R25_pept[1:1000], distribution="beta")
```

---

## limmaCompleteTest *Computes a hierarchical differential analysis*

---

### Description

This function is a limmaCompleteTest

### Usage

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

### 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

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- wrapper.mvImputation(Exp1_R25_pept[1:1000], "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

### Author(s)

Alexia Dorffer

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
matAdj <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, FALSE)
MeanPeptides(matAdj, Biobase::exprs(Exp1_R25_pept[1:1000]))
```

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

```
require(DAPARdata)
data(Exp1_R25_pept)
mvFilter(Exp1_R25_pept, "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 = "")
```

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
mvFilterFromIndices(Exp1_R25_pept, 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

```
require(DAPARdata)
data(Exp1_R25_pept)
mvFilterGetIndices(Exp1_R25_pept, "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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
mvHisto(qData, samplesData, labels, indLegend="auto", showValues=TRUE)
```

## mvImage *Heatmap of missing values*

### 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 vizualize easily the different number of missing values. A white square is plotted for missing values.

### Usage

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)[1:1000]
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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
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

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
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 vizualisation.

### 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

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"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)
```

## 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 Alignment, Median Centering, Mean Centering, Mean Centering Scaling. |
| `method` | "Overall" or "within conditions". |

## Value

A matrix normalized

## Author(s)

Florence Combes, Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"Label"]
normalizeD(qData, labels, "Median Centering", "within conditions")
```

normalizeD2 *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

```
normalizeD2(qData, labels, method, type, scaling = FALSE, quantile = 0.15)
```

## Arguments

| | |
|---|---|
| qData | A dataframe that contains quantitative data. |
| labels | A vector of strings containing the column "Label" of the pData(). |
| method | One of the following : Global Alignment, Quantile Centering, Mean Centering. |
| type | For the method "Global Alignment", the parameters are: "sum by columns": operates on the original scale (not the log2 one) and propose to normalize each abundance by the total abundance of the sample (so as to focus on the analyte proportions among each sample). "Alignment on all quantiles": proposes to align the quantiles of all the replicates; practically it amounts to replace abundances by order statistics. For the two other methods, the parameters are "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time). |
| scaling | A boolean that indicates if the variance of the data have to be forced to unit (variance reduction) or not. |
| quantile | A float that corresponds to the quantile used to align the data. |

## Value

A matrix normalized

## Author(s)

Samuel Wieczorek, Thomas Burger

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"Label"]
normalizeD2(qData, labels, "Quantile Centering", "within conditions", quantile = 0.15)
```

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

## Author(s)

Alexia Dorffer, Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
mat <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, TRUE)
pepAgregate(Exp1_R25_pept[1:1000], protID, "sum overall", mat)
```

proportionConRev *Barplot of proportion of contaminants and reverse*

## 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 prefix to identify Reverse |

### Value

A barplot

### Author(s)

Samuel Wieczorek

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
pref <- "+"
proportionConRev(Exp1_R25_pept, "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

```
require(DAPARdata)
data(Exp1_R25_pept)
removeLines(Exp1_R25_pept, "Potential.contaminant")
removeLines(Exp1_R25_pept, "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 |

## Value

A matrix of intensities of proteins

## Author(s)

Alexia Dorffer

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
M <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, FALSE)
SumPeptides(M, Biobase::exprs(Exp1_R25_pept[1:1000]))
```

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

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
matAdj <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], protID, FALSE)
TopnPeptides(matAdj, Biobase::exprs(Exp1_R25_pept[1:1000]), 3)
```

translatedRandomBeta *xxxxxxx*

## Description

This method xxxxxxxxxxx

## Usage

```
translatedRandomBeta(n, min, max, param1 = 3, param2 = 1)
```

### Arguments

| | |
|---|---|
| n | An integer which is the number of simulation (same as in rbeta) |
| min | An integer that corresponds to the lower bound of the interval |
| max | An integer that corresponds to the upper bound of the interval |
| param1 | An integer that is the first parameter of rbeta function. |
| param2 | An integer that is second parameter of rbeta function. |

### Value

A vector of n simulated values

### Author(s)

Thomas Burger

### Examples

```
translatedRandomBeta(1000, 5, 10, 1, 1)
```

---

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
library(vioplot)
qData <- Biobase::exprs(Exp1_R25_pept)
types <- c("Label","Analyt.Rep")
dataForXAxis <- Biobase::pData(Exp1_R25_pept)[,types]
labels <- Biobase::pData(Exp1_R25_pept)[,"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

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
types <- c("Label","Analyt.Rep")
wrapper.boxPlotD(Exp1_R25_pept, 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

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

## 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 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

```
require(DAPARdata)
data(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
objAfter <- wrapper.normalizeD(Exp1_R25_pept, "Median Centering",
"within conditions")
wrapper.compareNormalizationD(Exp1_R25_pept, 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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.corrMatrixD(Exp1_R25_pept)
```

## wrapper.CVDistD — *Distribution of CV of entities*

### Description

Builds a densityplot of the CV 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.CVDistD(obj)
```

### Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |

### Value

A density plot

## Author(s)

Alexia Dorffer

## See Also

wrapper.densityPlotD

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.CVDistD(Exp1_R25_pept)
```

---

# wrapper.dapar.impute.mi

*Missing values imputation using the LSimpute algorithm.*

---

## Description

This method is a wrapper to the function impute.mi of the package imp4p adapted to an object of class MSnSet.

## Usage

```
wrapper.dapar.impute.mi(obj, nb.iter = 3, nknn = 15, selec = 600,
  siz = 500, weight = 1, ind.comp = 1, progress.bar = TRUE,
  x.step.mod = 300, x.step.pi = 300, nb.rei = 100, method = 4,
  gridsize = 300, q = 0.95, q.min = 0, q.norm = 3, eps = 0,
  methodi = "slsa", lapala = TRUE, distribution = "unif")
```

## Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |
| nb.iter | Same as the function mi.mix in the package imp4p |
| nknn | Same as the function mi.mix in the package imp4p |
| selec | Same as the function mi.mix in the package imp4p |
| siz | Same as the function mi.mix in the package imp4p |
| weight | Same as the function mi.mix in the package imp4p |
| ind.comp | Same as the function mi.mix in the package imp4p |
| progress.bar | Same as the function mi.mix in the package imp4p |
| x.step.mod | Same as the function estim.mix in the package imp4p |
| x.step.pi | Same as the function estim.mix in the package imp4p |
| nb.rei | Same as the function estim.mix in the package imp4p |
| method | Same as the function estim.mix in the package imp4p |
| gridsize | Same as the function estim.mix in the package imp4p |
| q | Same as the function mi.mix in the package imp4p |
| q.min | Same as the function impute.pa in the package imp4p |

| | |
|---|---|
| q.norm | Same as the function impute.pa in the package imp4p |
| eps | Same as the function impute.pa in the package imp4p |
| methodi | Same as the function mi.mix in the package imp4p |
| lapala | xxxxxxxxxxx |
| distribution | xxxxxxxxxxx |

## Value

The exprs(obj) matrix with imputed values instead of missing values.

## Author(s)

Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
dat <- mvFilter(Exp1_R25_pept[1:1000], type="allCond", th = 1)
dat <- wrapper.dapar.impute.mi(dat, nb.iter=1)
```

---

# wrapper.densityPlotD *Builds a densityplot from an object of class* MSnSet

---

## Description

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

## Usage

```
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.CVDistD

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
labels <- Biobase::pData(Exp1_R25_pept)[,"Label"]
wrapper.densityPlotD(Exp1_R25_pept, 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

| | |
|---|---|
| obj | An object of class MSnSet. |
| 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)

Alexia Dorffer

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
wrapper.diffAnaLimma(Exp1_R25_pept[1:1000], 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

| | |
|---|---|
| obj | An object of class MSnSet. |
| condition1 | A vector containing the names of the conditions considered 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)

Alexia Dorffer

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
wrapper.diffAnaWelch(Exp1_R25_pept[1:1000], 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 fi the dendrogram has to be displayed |

### Value

A heatmap

### Author(s)

Alexia Dorffer

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- mvFilter(Exp1_R25_pept[1:1000], "wholeMatrix", 6)
wrapper.heatmapD(obj)
```

---

## wrapper.impute.pa *Imputation of peptides having no values in a biological condition.*

---

### Description

This method is a wrapper to the function impute.pa of the package imp4p adapted to an object of class MSnSet.

### Usage

```
wrapper.impute.pa(obj, q.min = 0.025)
```

### Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |
| q.min | Same as the function impute.pa in the package imp4p |

### Value

The exprs(obj) matrix with imputed values instead of missing values.

### Author(s)

Samuel Wieczorek

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
dat <- mvFilter(Exp1_R25_pept[1:1000], type="allCond", th = 1)
dat <- wrapper.impute.pa(dat)
```

wrapper.impute.pa2 *Missing values imputation from a* MSnSet *object*

## Description

This method is a wrapper to the function impute.pa from the package imp4p adapted to objects of class MSnSet.

## Usage

```
wrapper.impute.pa2(obj, q.min = 0, q.norm = 3, eps = 0,
  distribution = "unif")
```

## Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |
| q.min | A quantile value of the observed values allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0 (the maximal value is the minimum of observed values minus eps). |
| q.norm | A quantile value of a normal distribution allowing defining the minimal value which can be generated. Default is 3 (the minimal value is the maximal value minus qn*median(sd(observed values)) where sd is the standard deviation of a row in a condition). |
| eps | A value allowing defining the maximal value which can be generated. This maximal value is defined by the quantile q.min of the observed values distribution minus eps. Default is 0. |
| distribution | The type of distribution used. Values are unif or beta. |

## Value

The object obj which has been imputed

## Author(s)

Thomas Burger, Samuel Wieczorek

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.impute.pa2(Exp1_R25_pept[1:1000], distribution="beta")
```

## wrapper.mvHisto *Histogram of missing values from a MSnSet object*

### Description

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

### Usage

```
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 wether numeric values should be drawn above the bars. |

### Value

A histogram

### Author(s)

Alexia Dorffer

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvHisto(Exp1_R25_pept, 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 vizualize easily the different number of missing values. A white square is plotted for missing values.

### Usage

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

### Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |

## Value

A heatmap

## Author(s)

Alexia Dorffer

## Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvImage(Exp1_R25_pept)
```

---

`wrapper.mvImputation` *Missing values imputation from a* `MSnSet` *object*

---

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvImputation(Exp1_R25_pept[1:1000], "QRILC")
```

## wrapper.mvPerLinesHisto

*Histogram of missing values per lines from an object* MSnSet

### 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)
```

### 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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvPerLinesHisto(Exp1_R25_pept)
```

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvPerLinesHistoPerCondition(Exp1_R25_pept)
```

---

## 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 vizualisation.

### Usage

```
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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.mvTypePlot(Exp1_R25_pept)
```

`wrapper.normalizeD` *Normalization*

## Description

Provides several methods to normalize quantitative data from a `MSnSet` object. They are organized in four main families : Global Alignment, 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 (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 `exprs()` data tab) is computed condition by condition.

## Usage

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

## Arguments

| | |
|---|---|
| `obj` | An object of class `MSnSet`. |
| `family` | One of the following : Global Alignment, 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

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.normalizeD(Exp1_R25_pept[1:1000], "Median Centering", "within conditions")
```

`wrapper.normalizeD2` *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 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 `exprs()` data tab) is computed condition by condition.

### Usage

```
wrapper.normalizeD2(obj, method, type, scaling = FALSE, quantile = 0.15)
```

### Arguments

| | |
|---|---|
| obj | An object of class MSnSet. |
| method | One of the following : Global Alignment (for normalizations of important magnitude), Quantile Centering, Mean Centering. |
| type | For the method "Global Alignment", the parameters are: "sum by columns": operates on the original scale (not the log2 one) and propose to normalize each abundance by the total abundance of the sample (so as to focus on the analyte proportions among each sample). "Alignment on all quantiles": proposes to align the quantiles of all the replicates; practically it amounts to replace abundances by order statistics. For the two other methods, the parameters are "overall" (shift all the sample distributions at once) or "within conditions" (shift the sample distributions within each condition at a time). |
| scaling | A boolean that indicates if the variance of the data have to be forced to unit (variance reduction) or not. |
| quantile | A float that corresponds to the quantile used to align the data. |

### Value

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

### Author(s)

Samuel Wieczorek

### Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.normalizeD2(Exp1_R25_pept[1:1000], "Quantile Centering", "within conditions")
```

---

## 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

```
require(DAPARdata)
data(Exp1_R25_pept)
library(vioplot)
types <- c("Label","Analyt.Rep")
wrapper.violinPlotD(Exp1_R25_pept, 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

```
require(DAPARdata)
data(Exp1_R25_pept)
condition1 <- '25fmol'
condition2 <- '10fmol'
qData <- Biobase::exprs(Exp1_R25_pept[1:1000])
labels <- Biobase::pData(Exp1_R25_pept[1:1000])[,"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

```
require(DAPARdata)
data(Exp1_R2_pept)
obj <- Exp1_R2_pept[1:1000]
writeMSnsetToExcel(obj, "foo")
```

# Index