

Package ‘DAPAR’

April 15, 2019

Type Package

Title Tools for the Differential Analysis of Proteins Abundance with R

Version 1.14.5

Date 2019-02-09

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 (>= 3.5), foreach, parallel, doParallel

Suggests BiocGenerics, Biobase, testthat, BiocStyle

Imports MSnbase, RColorBrewer,stats,preprocessCore,Cairo,png,
lattice,reshape2,gplots,pcaMethods,ggplot2,
limma,knitr,tmvtnorm,norm,impute, stringr, grDevices, graphics,
openxlsx, utils, cp4p (>= 0.3.5), scales, Matrix, vioplot,
imp4p (>= 0.7), highcharter (>= 0.5.0), DAPARdata (>= 1.11.2),
siggenes, graph, lme4, readxl, clusterProfiler, dplyr,
tidyr,AnnotationDbi, tidyverse, vsn, FactoMineR, factoextra

biocViews ImmunoOncology, Proteomics, Normalization, Preprocessing,
MassSpectrometry, QualityControl, GO, DataImport

NeedsCompilation no

RoxygenNote 6.1.1

git_url <https://git.bioconductor.org/packages/DAPAR>

git_branch RELEASE_3_8

git_last_commit 4425fb4

git_last_commit_date 2019-02-09

Date/Publication 2019-04-15

R topics documented:

addOriginOfValue	5
aggregateIter	5
aggregateIterParallel	6
aggregateMean	7
aggregateSum	7
aggregateTopn	8
barplotEnrichGO_HC	9
barplotGroupGO_HC	9
boxPlotD	10
boxPlotD_HC	11
BuildAdjacencyMatrix	11
BuildColumnToProteinDataset	12
BuildColumnToProteinDataset_par	13
check.conditions	14
check.design	14
compareNormalizationD	15
compareNormalizationD_HC	16
compute.t.tests	17
corrMatrixD	18
corrMatrixD_HC	18
CountPep	19
createMSnset	20
CVDistD	21
CVDistD_HC	22
deleteLinesFromIndices	22
densityPlotD	23
densityPlotD_HC	24
diffAnaComputeFDR	25
diffAnaGetSignificant	26
diffAnaSave	26
diffAnaVolcanoplot	27
diffAnaVolcanoplot_rCharts	28
enrich_GO	30
finalizeAggregation	31
findMECBlock	31
formatLimmaResult	32
fudge2LRT	33
getIndicesConditions	34
getIndicesOfLinesToRemove	34
getListNbValuesInLines	35
GetNbPeptidesUsed	36
getNumberOf	36
getNumberOfEmptyLines	37
getPourcentageOfMV	37
getProcessingInfo	38
getProteinsStats	39
getQuantile4Imp	39
getTextForAggregation	40
getTextForAnaDiff	41
getTextForFiltering	41

getTextForGOAnalysis	42
getTextForHypothesisTest	42
getTextForNewDataset	43
getTextForNormalization	43
getTextForpeptideImputation	44
getTextForproteinImputation	44
GOAnalysisSave	45
GraphPepProt	46
group_GO	46
hc_logFC_DensityPlot	47
hc_mvTypePlot2	48
heatmap.DAPAR	49
heatmapD	50
impute.detQuant	50
impute.pa2	51
inner.aggregate.iter	52
inner.aggregate.topn	53
inner.mean	53
inner.sum	54
is.MV	54
is.OfType	55
LH0	56
LH0.lm	56
LH1	57
LH1.lm	57
limmaCompleteTest	58
listSheets	58
make.contrast	59
make.design	60
make.design.1	60
make.design.2	61
make.design.3	61
mvFilter	62
mvFilterFromIndices	63
mvFilterGetIndices	64
mvHisto	65
mvHisto_HC	66
mvImage	67
mvPerLinesHisto	67
mvPerLinesHistoPerCondition	68
mvPerLinesHistoPerCondition_HC	69
mvPerLinesHisto_HC	70
my_hc_chart	70
my_hc_ExportMenu	71
nonzero	72
pepa.test	72
plotPCA_Eigen	73
plotPCA_Eigen_hc	74
plotPCA_Ind	74
plotPCA_Var	75
proportionConRev_HC	76
rbindMSnset	76

readExcel	77
reIntroduceMEC	78
removeLines	78
samLRT	79
saveParameters	80
scatterplotEnrichGO_HC	81
setMEC	81
StringBasedFiltering	82
StringBasedFiltering2	83
test.design	83
translatedRandomBeta	84
univ_AnnotDbPkg	85
violinPlotD	85
wrapper.compareNormalizationD	86
wrapper.compareNormalizationD_HC	87
wrapper.corrMatrixD	88
wrapper.corrMatrixD_HC	88
wrapper.CVDistD	89
wrapper.CVDistD_HC	90
wrapper.dapar.impute.mi	90
wrapper.hc_mvTypePlot2	92
wrapper.heatmapD	92
wrapper.impute.detQuant	93
wrapper.impute.fixedValue	94
wrapper.impute.KNN	94
wrapper.impute.mle	95
wrapper.impute.pa	96
wrapper.impute.pa2	96
wrapper.impute.slsa	97
wrapper.mvHisto	98
wrapper.mvHisto_HC	99
wrapper.mvImage	99
wrapper.mvPerLinesHisto	100
wrapper.mvPerLinesHistoPerCondition	101
wrapper.mvPerLinesHistoPerCondition_HC	101
wrapper.mvPerLinesHisto_HC	102
wrapper.normalizedD	103
wrapper.pca	104
wrapper.t_test_Complete	104
wrapperCalibrationPlot	105
writeMSnsetToCSV	106
writeMSnsetToExcel	107

addOriginOfValue *Sets the OriginOfValues dataframe*

Description

Sets the OriginOfValues dataframe in the fData table

Usage

```
addOriginOfValue(obj, index = NULL)
```

Arguments

obj	An object of class MSnSet
index	A list of integer xxxxxxx

Value

An instance of class MSnSet.

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
addOriginOfValue(Exp1_R25_pept)
```

aggregateIter *xxxx*

Description

Method to xxxxx

Usage

```
aggregateIter(obj.pep, X, init.method = "Sum", method = "Mean",
  n = NULL)
```

Arguments

obj.pep	xxxxxx
X	xxxx
init.method	xxxxxx
method	xxxxxx
n	xxxx

Value

A protein object of class MSnset

Author(s)

Samuel Wieczorek

Examples

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

aggregateIterParallel xxxx

Description

Method to xxxxx

Usage

```
aggregateIterParallel(obj.pep, X, init.method = "Sum", method = "Mean",
  n = NULL)
```

Arguments

obj.pep	xxxxxx
X	xxxx
init.method	xxxxxx
method	xxxxxx
n	xxxx

Value

xxxxxx

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
obj.pep <- Exp1_R25_pept[1:1000]
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
aggregateIterParallel(obj.pep, X)
```

aggregateMean	<i>Compute the intensity of proteins as the mean of the intensities of their peptides.</i>
---------------	--

Description

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

Usage

```
aggregateMean(obj.pep, X)
```

Arguments

obj.pep	A peptide object of class MSnset
X	An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

Value

A matrix of intensities of proteins

Author(s)

Alexia Dorffer

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj.pep <- Exp1_R25_pept[1:1000]
protID <- "Protein.group.IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
aggregateMean(obj.pep, X)
```

aggregateSum	<i>Compute the intensity of proteins with the sum of the intensities of their peptides.</i>
--------------	---

Description

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

Usage

```
aggregateSum(obj.pep, X)
```

Arguments

obj.pep A matrix of intensities of peptides
 X An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.

Value

A matrix of intensities of proteins

Author(s)

Alexia Dorffer

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
obj.pep <- Exp1_R25_pept[1:1000]
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
DAPAR::aggregateSum(obj.pep, X)
```

aggregateTopn	<i>Compute the intensity of proteins as the sum of the intensities of their n best peptides.</i>
---------------	--

Description

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

Usage

```
aggregateTopn(obj.pep, X, method = "Mean", n = 10)
```

Arguments

obj.pep A matrix of intensities of peptides
 X An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
 method xxx
 n The maximum number of peptides used to aggregate a protein.

Value

A matrix of intensities of proteins

Author(s)

Alexia Dorffer, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj.pep <- Exp1_R25_pept[1:1000]
protID <- "Protein.group.IDs"
X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
DAPAR::aggregateTopn(obj.pep, X, n=3)
```

barplotEnrichGO_HC	<i>A barplot that shows the result of a GO enrichment, using the package highcharter</i>
--------------------	--

Description

A barplot of GO enrichment analysis

Usage

```
barplotEnrichGO_HC(ego, maxRes = 5, title = NULL)
```

Arguments

ego	The result of the GO enrichment, provides either by the function <code>enrichGO</code> in the package <code>DAPAR</code> or the function <code>enrichGO</code> of the package <code>clusterProfiler</code>
maxRes	The maximum number of categories to display in the plot
title	The title of the plot

Value

A barplot

Author(s)

Samuel Wieczorek

barplotGroupGO_HC	<i>A barplot which shows the result of a GO classification, using the package highcharter</i>
-------------------	---

Description

A barplot of GO classification analysis

Usage

```
barplotGroupGO_HC(ggo, maxRes = 5, title = "")
```

Arguments

<code>ggo</code>	The result of the GO classification, provides either by the function <code>group_GO</code> in the package <code>DAPAR</code> or the function <code>groupGO</code> in the package clusterProfiler
<code>maxRes</code>	An integer which is the maximum number of classes to display in the plot
<code>title</code>	The title of the plot

Value

A barplot

Author(s)

Samuel Wieczorek

<code>boxPlotD</code>	<i>Builds a boxplot from a dataframe</i>
-----------------------	--

Description

Boxplot for quantitative proteomics data

Usage

```
boxPlotD(obj, legend = NULL, palette = NULL)
```

Arguments

<code>obj</code>	xxx
<code>legend</code>	A vector of the conditions (one string per sample).
<code>palette</code>	xxx

Value

A boxplot

Author(s)

Florence Combes, Samuel Wieczorek

See Also

[densityPlotD](#)

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
boxPlotD(Exp1_R25_pept, legend = conds)
```

boxPlotD_HC	<i>Builds a boxplot from a dataframe using the library highcharter</i>
-------------	--

Description

Boxplot for quantitative proteomics data using the library highcharter

Usage

```
boxPlotD_HC(obj, legend = NULL, palette = NULL)
```

Arguments

obj	xxx
legend	A vector of the conditions (one condition per sample).
palette	xxx

Value

A boxplot

Author(s)

Samuel Wieczorek

See Also

[densityPlotD_HC](#)

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
legend <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
boxPlotD_HC(Exp1_R25_pept, legend)
```

BuildAdjacencyMatrix	<i>Function matrix of appartenance group</i>
----------------------	--

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

<code>obj.pep</code>	An object (peptides) of class <code>MSnSet</code> .
<code>protID</code>	The name of proteins ID column
<code>unique</code>	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 Wiczorek, 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 aggregation by using the previous peptide dataset.

Usage

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

Arguments

<code>peptideData</code>	A data.frame of meta data of peptides. It is the <code>fData</code> of the <code>MSnset</code> object.
<code>matAdj</code>	The adjacency matrix used to agregate the peptides data.
<code>columnName</code>	The name of the column in <code>fData(peptides_MSnset)</code> that the user wants to keep in the new protein data.frame.
<code>proteinNames</code>	The names of the protein in the new dataset (i.e. rownames)

Value

A vector

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
protID <- "Protein.group.IDs"
obj.pep <- Exp1_R25_pept[1:1000]
M <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
data <- Biobase::fData(obj.pep)
protData <- DAPAR::aggregateMean(obj.pep, M)
name <- "Protein.group.IDs"
proteinNames <- rownames(Biobase::fData(protData))
BuildColumnToProteinDataset(data, M, name,proteinNames )
```

BuildColumnToProteinDataset_par

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. It is a parallel version of the function BuildColumnToProteinDataset

Usage

```
BuildColumnToProteinDataset_par(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"
obj.pep <- Exp1_R25_pept[1:1000]
M <- BuildAdjacencyMatrix(obj.pep, protID, FALSE)
data <- Biobase::fData(obj.pep)
protData <- DAPAR::aggregateSum(obj.pep, M)
```

```
name <- "Protein.group.IDs"
proteinNames <- rownames(Biobase::fData(protData))
BuildColumnToProteinDataset_par(data, M, name,proteinNames )
```

check.conditions *Check if the design is valid*

Description

This function check the validity of the conditions

Usage

```
## S3 method for class 'conditions'
check(conds)
```

Arguments

conds A vector

Value

A list

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
check.conditions(Biobase::pData(Exp1_R25_pept)$Condition)
```

check.design *Check if the design is valid*

Description

This function check the validity of the experimental design

Usage

```
## S3 method for class 'design'
check(sTab)
```

Arguments

sTab The data.frame which correspond to the pData function of MSnbase

Value

A boolean

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
check.design(Biobase::pData(Exp1_R25_pept)[,1:3])
```

compareNormalizationD *Builds a plot from a dataframe*

Description

Plot to compare the quantitative proteomics data before and after normalization

Usage

```
compareNormalizationD(qDataBefore, qDataAfter, condsForLegend = NULL,
  indData2Show = NULL, palette = NULL)
```

Arguments

qDataBefore	A dataframe that contains quantitative data before normalization.
qDataAfter	A dataframe that contains quantitative data after normalization.
condsForLegend	A vector of the conditions (one condition 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 data.frame qDataBefore.
palette	xxx

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qDataBefore <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
objAfter <- wrapper.normalized(Exp1_R25_pept,"QuantileCentering","within conditions")
compareNormalizationD(qDataBefore, Biobase::exprs(objAfter), conds)
```

compareNormalizationD_HC

Builds a plot from a dataframe. Same as compareNormalizationD but uses the library highcharter

Description

Plot to compare the quantitative proteomics data before and after normalization using the library highcharter

Usage

```
compareNormalizationD_HC(qDataBefore, qDataAfter, condsForLegend = NULL,  
  indData2Show = NULL, palette = NULL)
```

Arguments

qDataBefore A dataframe that contains quantitative data before normalization.

qDataAfter A dataframe that contains quantitative data after normalization.

condsForLegend A vector of the conditions (one condition 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 data.frame qDataBefore.

palette xxx

Value

A plot

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)  
data(Exp1_R25_pept)  
obj <- Exp1_R25_pept[1:1000]  
qDataBefore <- Biobase::exprs(obj)  
conds <- Biobase::pData(obj)[,"Condition"]  
objAfter <- wrapper.normalized(obj,"QuantileCentering","within conditions")  
compareNormalizationD_HC(qDataBefore, Biobase::exprs(objAfter), conds)
```

```
compute.t.tests      xxxxxx
```

Description

This function is xxxxxx

Usage

```
## S3 method for class 't.tests'
compute(qData, Conditions, Contrast = "OnevsOne",
        type = "Student")
```

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.
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).
type	xxxxx

Value

A list of two items : logFC and P_Value; both are dataframe. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

Author(s)

Florence Combes, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
lapala <- findMECBlock(obj)
obj <- wrapper.impute.detQuant(obj)
obj <- reIntroduceMEC(obj, lapala)
obj <- wrapper.impute.detQuant(obj)
ttest <- compute.t.tests(Biobase::exprs(obj), Biobase::pData(obj)[,"Condition"],1)
```

corrMatrixD	<i>Displays a correlation matrix of the quantitative data of the exprs() table.</i>
-------------	---

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 Wiczorek

Examples

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

corrMatrixD_HC	<i>Displays a correlation matrix of the quantitative data of the exprs() table.</i>
----------------	---

Description

Correlation matrix based on a MSnSet object. Same as the function [corrMatrixD](#) but uses the package highcharter

Usage

```
corrMatrixD_HC(object, samplesData = NULL, rate = 0.5)
```

Arguments

object	The result of the cor function.
samplesData	A dataframe in which lines correspond to samples and columns to the meta-data for those samples.
rate	The rate parameter to control the exponential law for the gradient of colors

Value

A colored correlation matrix

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
samplesData <- Biobase::pData(Exp1_R25_pept)
res <- cor(qData,use = 'pairwise.complete.obs')
corrMatrixD_HC(res, 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, indexForOriginOfValue = NULL, logData = FALSE,
             replaceZeros = FALSE, pep_prot_data = NULL, proteinId = NULL,
             versions = 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)
indexForOriginOfValue	xxxxxxxxxxx
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
proteinId	xxxx
versions	A list of the following items: Prostar_Version, DAPAR_Version peptides or proteins.

Value

An instance of class MSnSet.

Author(s)

Florence Combes, Samuel Wiczorek

Examples

```
require(DAPARdata)
require(Matrix)
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, indexForOriginOfValue = NULL, pep_prot_dat
```

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 present in the dataset (see the slot 'Condition' in the `pData()` table)

Usage

```
CVDistD(qData, conds = NULL, palette = NULL)
```

Arguments

<code>qData</code>	A dataframe that contains quantitative data.
<code>conds</code>	A vector of the conditions (one condition per sample).
<code>palette</code>	xxx

Value

A density plot

Author(s)

Florence Combes, Samuel Wiczorek

See Also

[densityPlotD](#).

Examples

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

CVDistD_HC	<i>Distribution of CV of entities</i>
------------	---------------------------------------

Description

Builds a densityplot of the CV of entities in the exprs() table of a object. The CV is calculated for each condition present in the dataset (see the slot 'Condition' in the pData() table) Same as the function CVDistD but uses the package highcharter

Usage

```
CVDistD_HC(qData, conds = NULL, palette = NULL)
```

Arguments

qData	A dataframe that contains quantitative data.
conds	A vector of the conditions (one condition per sample).
palette	xxx

Value

A density plot

Author(s)

Samuel Wieczorek

See Also

[densityPlotD](#).

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
CVDistD_HC(Biobase::exprs(Exp1_R25_pept), conds)
```

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 Wiczorek

Examples

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

densityPlotD	<i>Builds a densityplot from a dataframe</i>
--------------	--

Description

Densityplot of quantitative proteomics data over samples.

Usage

```
densityPlotD(obj, legend = NULL, palette = NULL)
```

Arguments

obj xxx
legend A vector of the conditions (one condition per sample).
palette xxx

Value

A density plot

Author(s)

Florence Combes, Samuel Wiczorek

See Also

[boxPlotD](#), [CVDistD](#)

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
densityPlotD(Exp1_R25_pept, conds)
```

densityPlotD_HC	<i>Builds a densityplot from a dataframe</i>
-----------------	--

Description

Densityplot of quantitative proteomics data over samples. Same as the function [densityPlotD](#) but uses the package `highcharter`

Usage

```
densityPlotD_HC(obj, legend = NULL, palette = NULL)
```

Arguments

obj	xxx
legend	A vector of the conditions (one condition per sample).
palette	xxx

Value

A density plot

Author(s)

Samuel Wieczorek

See Also

[boxPlotD](#), [CVDistD](#)

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
densityPlotD_HC(Exp1_R25_pept, conds)
```

diffAnaComputeFDR	<i>Computes the FDR corresponding to the p-values of the differential analysis using</i>
-------------------	--

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

```
diffAnaComputeFDR(logFC, pval, threshold_PVal = 0, threshold_LogFC = 0,  
pi0Method = 1)
```

Arguments

<code>logFC</code>	The result (logFC values) of the differential analysis processed by limmaCompleteTest
<code>pval</code>	The result (p-values) of the differential analysis processed by limmaCompleteTest
<code>threshold_PVal</code>	The threshold on p-value to distinguish between differential and non-differential data
<code>threshold_LogFC</code>	The threshold on log(Fold Change) to distinguish between differential and non-differential data
<code>pi0Method</code>	The parameter <code>pi0.method</code> of the method <code>adjust.p</code> in the package <code>cp4p</code>

Value

The computed FDR value (floating number)

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)  
data(Exp1_R25_pept)  
obj <- Exp1_R25_pept[1:1000]  
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))  
obj <- mvFilterFromIndices(obj, keepThat)  
qData <- Biobase::exprs(obj)  
sTab <- Biobase::pData(obj)  
limma <- limmaCompleteTest(qData, sTab)  
diffAnaComputeFDR(limma$logFC[,1], limma$P_Value[,1])
```

`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)
obj <- Exp1_R25_pept
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
allComp <- limmaCompleteTest(qData,sTab)
data <- list(logFC=allComp$logFC[1], P_Value = allComp$P_Value[1])
obj <- diffAnaSave(obj, allComp, data)
signif <- diffAnaGetSignificant(obj)
```

`diffAnaSave` *Returns a MSnSet object with the results of the differential analysis performed with [limma](#) package.*

Description

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

Usage

```
diffAnaSave(obj, allComp, data = NULL, th_pval = 0, th_logFC = 0)
```

Arguments

obj	An object of class MSnSet.
allComp	A list of two items which is the result of the function wrapper.limmaCompleteTest or xxxx
data	The result of the differential analysis processed by limmaCompleteTest
th_pval	xxx
th_logFC	xxx

Value

A MSnSet

Author(s)

Alexia Dorffer, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
allComp <- limmaCompleteTest(qData,sTab)
data <- list(logFC=allComp$logFC[1], P_Value = allComp$P_Value[1])
diffAnaSave(obj, allComp, data)
```

diffAnaVolcanoplot *Volcanoplot of the differential analysis*

Description

Plots a volcano plot 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, colors = 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.
colors	xxx

Value

A volcanoplot

Author(s)

Florence Combes, Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
limma <- limmaCompleteTest(qData,sTab)
diffAnaVolcanoplot(limma$logFC[,1], limma$P_Value[,1])
```

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,
  palette = 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 rownames 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 <code>this.index</code> refers to the slot named <code>index</code> and allows to retrieve the right row to show in the tooltip.
palette	xxx

Value

An interactive volcano plot

Author(s)

Samuel Wiczorek

Examples

```
library(highcharter)
library(tidyverse)
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
data <- limmaCompleteTest(qData,sTab)
df <- data.frame(x=data$logFC, y = -log10(data$P_Value), index = as.character(rownames(obj)))
colnames(df) <- c("x", "y", "index")
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]+'_'+ [this.s
cond <- c("25fmol", "10fmol")
diffAnaVolcanoplot_rCharts(df, 2.5, 1, cond,hc_clickFunction)
```

enrich_GO	<i>Calculates GO enrichment classes for a given list of proteins/genes ID. It results an enrichResult instance.</i>
-----------	---

Description

This function is a wrapper to the function `enrichGO` from the package `clusterProfiler`. Given a vector of genes/proteins, it returns an `enrichResult` instance.

Usage

```
enrich_GO(data, idFrom, orgdb, ont, readable = FALSE, pval, universe)
```

Arguments

<code>data</code>	A vector of ID (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT -can be different according to organisms)
<code>idFrom</code>	character indicating the input ID format (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT)
<code>orgdb</code>	annotation Bioconductor package to use (character format)
<code>ont</code>	One of "MF", "BP", and "CC" subontologies
<code>readable</code>	TRUE or FALSE (default FALSE)
<code>pval</code>	The qvalue cutoff (same parameter as in the function <code>enrichGO</code> of the package <code>clusterProfiler</code>)
<code>universe</code>	a list of ID to be considered as the background for enrichment calculation

Value

A `groupGOResult` instance.

Author(s)

Florence Combes

Examples

```
require(DAPARdata)
data(Exp1_R25_prot)
univ<-univ_AnnotDbPkg("org.Sc.sgd.db") #univ is the background
ego<-enrich_GO(data=fData(Exp1_R25_prot)$Protein.IDs, idFrom="UNIPROT",
orgdb="org.Sc.sgd.db",ont="MF", pval=0.05, universe = univ)
```

finalizeAggregation *Finalizes the aggregation process*

Description

Method to finalize the aggregation process

Usage

```
finalizeAggregation(obj.pep, pepData, protData, X, lib.loc = NULL)
```

Arguments

obj.pep	A peptide object of class MSnset
pepData	xxxx
protData	xxxxx
X	An adjacency matrix in which lines and columns correspond respectively to peptides and proteins.
lib.loc	A list of two items (lib.loc\$Prostar.loc and lib.loc\$DAPAR.loc) to provide the location of the installed packages

Value

A protein object of class MSnset

Author(s)

Samuel Wieczorek

findMECBlock *Finds the LAPALA into a MSnSet object*

Description

This method finds the LAPALA in a dataset.

Usage

```
findMECBlock(obj)
```

Arguments

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

Value

A data.frame that contains the indexes of LAPALA

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
lapala <- findMECBlock(obj)
```

formatLimmaResult	xxxx
-------------------	------

Description

This function is xxxx

Usage

```
formatLimmaResult(fit, conds, contrast)
```

Arguments

fit	xxxx
conds	xxxx
contrast	xxxx

Value

A list of two dataframes : logFC and P_Value. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
limma <- limmaCompleteTest(qData,sTab)
```

fudge2LRT	<i>Heuristic to choose the value of the hyperparameter (fudge factor) used to regularize the variance estimator in the likelihood ratio statistic</i>
-----------	---

Description

fudge2LRT: heuristic to choose the value of the hyperparameter (fudge factor) used to regularize the variance estimator in the likelihood ratio statistic (as implemented in samLRT). We follow the heuristic described in [1] and adapt the code of the fudge2 function in the siggene R package. [1] Tusher, Tibshirani and Chu, Significance analysis of microarrays applied to the ionizing radiation response, PNAS 2001 98: 5116-5121, (Apr 24).

Usage

```
fudge2LRT(lmm.res.h0, lmm.res.h1, cc, n, p, s, alpha = seq(0, 1, 0.05),
  include.zero = TRUE)
```

Arguments

lmm.res.h0	a vector of object containing the estimates (used to compute the statistic) under H0 for each connected component. If the fast version of the estimator was used (as implemented in this package), lmm.res.h0 is a vector containing averages of squared residuals. If a fixed effect model was used, it is a vector of lm objects and if a mixed effect model was used it is a vector or lmer object.
lmm.res.h1	similar to lmm.res.h0, a vector of object containing the estimates (used to compute the statistic) under H1 for each protein.
cc	a list containing the indices of peptides and proteins belonging to each connected component.
n	the number of samples used in the test
p	the number of proteins in the experiment
s	a vector containing the maximum likelihood estimate of the variance for the chosen model. When using the fast version of the estimator implemented in this package, this is the same thing as the input lmm.res.h1. For other models (e.g. mixed models) it can be obtained from samLRT.
alpha	A vector of proportions used to build candidate values for the regularizer. We use quantiles of s with these proportions. Default to seq(0, 1, 0.05)
include.zero	logical value indicating if 0 should be included in the list of candidates. Default to TRUE.

Value

(same as the fudge2 function of siggene): s.zero: the value of the fudge factor s0. alpha.hat: the optimal quantile of the 's' values. If s0=0, 'alpha.hat' will not be returned. vec.cv: the vector of the coefficients of variations. Following Tusher et al. (2001), the optimal 'alpha' quantile is given by the quantile that leads to the smallest CV of the modified test statistics. msg: a character string summarizing the most important information about the fudge factor.

Author(s)

Thomas Burger, Laurent Jacob

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(conds, cond1, cond2)
```

Arguments

conds	A vector of strings containing the column "Condition" of the pData().
cond1	A vector of Conditions (a slot in the pData() table) for the condition 1.
cond2	A vector of Conditions (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 Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
getIndicesConditions(conds, "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="+")
```

getListNbValuesInLines

Returns the possible number of values in lines in the data

Description

Returns the possible number of values in lines in a matrix.

Usage

```
getListNbValuesInLines(obj, type = "wholeMatrix")
```

Arguments

obj An object of class MSnSet
type xxxxxxxx

Value

An integer

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
getListNbValuesInLines(Exp1_R25_pept)
```

GetNbPeptidesUsed	<i>Computes the number of peptides used for aggregating each protein</i>
-------------------	--

Description

Method to compute the number of quantified peptides used for aggregating each protein

Usage

```
GetNbPeptidesUsed(X, pepData)
```

Arguments

X	An adjacency matrix
pepData	A data.frame of quantitative data

Value

A data.frame

Author(s)

Samuel Wieczorek

getNumberOf	<i>Number of lines with prefix</i>
-------------	------------------------------------

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)
```

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 Wiczorek

Examples

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

getProcessingInfo *Returns the contents of the slot processing of an object of class MSnSet*

Description

Returns the contents of the slot processing of an object of class MSnSet

Usage

```
getProcessingInfo(obj)
```

Arguments

obj An object (peptides) of class MSnSet.

Value

The slot processing of obj@processingData

Author(s)

Samuel Wiczorek

Examples

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

getProteinsStats	<i>Computes the number of proteins that are only defined by specific peptides, shared peptides or a mixture of two.</i>
------------------	---

Description

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

Usage

```
getProteinsStats(matShared)
```

Arguments

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"
obj <- Exp1_R25_pept[1:1000]
MShared <- BuildAdjacencyMatrix(obj, protID, FALSE)
getProteinsStats(MShared)
```

getQuantile4Imp	<i>Quantile imputation value definition</i>
-----------------	---

Description

This method returns the q-th quantile of each column of an expression set, up to a scaling factor

Usage

```
getQuantile4Imp(qData, qval = 0.025, factor = 1)
```

Arguments

qData An expression set containing quantitative values of various replicates
qval The quantile used to define the imputation value
factor A scaling factor to multiply the imputation value with

Value

A list of two vectors, respectively containing the imputation values and the rescaled imputation values

Author(s)

Thomas Burger

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
getQuantile4Imp(qData)
```

getTextForAggregation *Build the text information for the Aggregation process*

Description

Builds the text information for the Aggregation process

Usage

```
getTextForAggregation(l.params)
```

Arguments

l.params A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wiczorek

Examples

```
params <- list()
getTextForAggregation(params)
```

`getTextForAnaDiff` *Build the text information for the Aggregation process*

Description

Build the text information for the differential Analysis process

Usage

```
getTextForAnaDiff(l.params)
```

Arguments

`l.params` A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
getTextForAnaDiff(list(design="OnevsOne",method="Limma"))
```

`getTextForFiltering` *Build the text information for the filtering process*

Description

Build the text information for the filtering process

Usage

```
getTextForFiltering(l.params)
```

Arguments

`l.params` A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
getTextForFiltering(list(mvFilterType="wholeMatrix",mvThNA=3))
```

`getTextForGOAnalysis` *Build the text information for the Aggregation process*

Description

Build the text information for the Aggregation process

Usage

```
getTextForGOAnalysis(l.params)
```

Arguments

`l.params` A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
getTextForGOAnalysis(list())
```

`getTextForHypothesisTest`
Build the text information for the hypothesis test process

Description

Builds the text information for the hypothesis test process

Usage

```
getTextForHypothesisTest(l.params)
```

Arguments

`l.params` A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
params <- list(design='OnevsOne', method='limma')
getTextForHypothesisTest(params)
```

getTextForNewDataset *Build the text information for a new dataset*

Description

Build the text information for a new dataset

Usage

```
getTextForNewDataset(l.params)
```

Arguments

l.params A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
getTextForNewDataset(list(filename="foo.MSnet"))
```

getTextForNormalization
Build the text information for the Normalization process

Description

Build the text information for the Normalization process

Usage

```
getTextForNormalization(l.params)
```

Arguments

l.params A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
getTextForNormalization(list(method="SumByColumns"))
```

```
getTextForpeptideImputation
```

Build the text information for the peptide Imputation process

Description

Build the text information for the peptide Imputation process

Usage

```
getTextForpeptideImputation(l.params)
```

Arguments

l.params A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
params <- list()  
getTextForpeptideImputation(params)
```

```
getTextForproteinImputation
```

Build the text information for the protein Imputation process

Description

Build the text information for the Protein Imputation process

Usage

```
getTextForproteinImputation(l.params)
```

Arguments

l.params A list of parameters related to the process of the dataset

Value

A string

Author(s)

Samuel Wieczorek

Examples

```
params <- list()
getTextForproteinImputation(params)
```

GOAnalysisSave	<i>Returns an MSnSet object with the results of the GO analysis performed with the functions <code>enrichGO</code> and/or <code>groupGO</code> of the <code>clusterProfiler</code> package.</i>
----------------	---

Description

This method returns an MSnSet object with the results of the Gene Ontology analysis.

Usage

```
GOAnalysisSave(obj, ggo_res = NULL, ego_res = NULL, organism, ontology,
  levels, pvalueCutoff, typeUniverse)
```

Arguments

<code>obj</code>	An object of the class MSnSet
<code>ggo_res</code>	The object returned by the function <code>group_GO</code> of the package DAPAR or the function <code>groupGO</code> of the package <code>clusterProfiler</code>
<code>ego_res</code>	The object returned by the function <code>enrich_GO</code> of the package DAPAR or the function <code>enrichGO</code> of the package <code>clusterProfiler</code>
<code>organism</code>	The parameter <code>OrgDb</code> of the functions <code>bitr</code> , <code>groupGO</code> and <code>enrichGO</code>
<code>ontology</code>	One of "MF", "BP", and "CC" subontologies
<code>levels</code>	A vector of the different GO grouping levels to save
<code>pvalueCutoff</code>	The qvalue cutoff (same parameter as in the function <code>enrichGO</code> of the package <code>clusterProfiler</code>)
<code>typeUniverse</code>	The type of background to be used. Values are 'Entire Organism', 'Entire dataset' or 'Custom'. In the latter case, a file should be uploaded by the user

Value

An object of the class MSnSet

Author(s)

Samuel Wieczorek

GraphPepProt	<i>Function to create a histogram that shows the repartition of peptides w.r.t. the proteins</i>
--------------	--

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 Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
mat <- BuildAdjacencyMatrix(Exp1_R25_pept[1:1000], "Protein.group.IDs")
GraphPepProt(mat)
```

group_GO	<i>Calculates the GO profile of a vector of genes/proteins at a given level of the Gene Ontology</i>
----------	--

Description

This function is a wrapper to the function `groupGO` from the package `clusterProfiler`. Given a vector of genes/proteins, it returns the GO profile at a specific level. It returns a `groupGOResult` instance.

Usage

```
group_GO(data, idFrom, orgdb, ont, level, readable = FALSE)
```

Arguments

data	A vector of ID (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT -can be different according to organisms)
idFrom	character indicating the input ID format (among ENSEMBL, ENTREZID, GENENAME, REFSEQ, UNIGENE, UNIPROT)
orgdb	annotation Bioconductor package to use (character format)
ont	on which ontology to perform the analysis (MF, BP or CC)
level	level of the ontolofy to perform the analysis
readable	TRUE or FALSE (default FALSE)

Value

GO profile at a specific level

Author(s)

Florence Combes

Examples

```
require(DAPARdata)
data(Exp1_R25_prot)
ggo<-group_GO(data=fData(Exp1_R25_prot)$Protein.IDs, idFrom="UNIPROT",
orgdb="org.Sc.sgd.db", ont="MF", level=2)
```

hc_logFC_DensityPlot *Density plots of logFC values*

Description

This function show the density plots of Fold Change (the same as calculated by limma) for a list of the comparisons of conditions in a differnetial analysis.

Usage

```
hc_logFC_DensityPlot(df_logFC, threshold_LogFC = 0, palette = NULL)
```

Arguments

df_logFC	A dataframe that contains the logFC values
threshold_LogFC	The threshold on log(Fold Change) to distinguish between differential and non-differential data
palette	xxx

Value

A highcharts density plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
res <- limmaCompleteTest(qData,sTab)
hc_logFC_DensityPlot(res$logFC)
```

`hc_mvTypePlot2`*Distribution of Observed values with respect to intensity values*

Description

This method shows density plots which represents the repartition of Partial Observed Values for each replicate in the dataset. The colors correspond to the different conditions (slot Condition 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 observed values for this entity and the considered condition.

Usage

```
hc_mvTypePlot2(qData, conds, palette = NULL)
```

Arguments

<code>qData</code>	A dataframe that contains quantitative data.
<code>conds</code>	A vector of the conditions (one condition per sample).
<code>palette</code>	xxx

Value

Density plots

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
hc_mvTypePlot2(qData, conds)
```

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

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

<code>x</code>	A dataframe that contains quantitative data.
<code>col</code>	colors used for the image. Defaults to heat colors (<code>heat.colors</code>).
<code>srtCol</code>	angle of column conds, in degrees from horizontal
<code>labCol</code>	character vectors with column conds to use.
<code>labRow</code>	character vectors with row conds to use.
<code>key</code>	logical indicating whether a color-key should be shown.
<code>key.title</code>	main title of the color key. If set to <code>NA</code> no title will be plotted.
<code>main</code>	main title; default to none.
<code>ylab</code>	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	<i>This function is a wrapper to heatmap.2 that displays quantitative data in the <code>exprs()</code> table of an object of class MSnSet</i>
----------	---

Description

Heatmap of the quantitative proteomic data of a MSnSet object

Usage

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

Arguments

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

Value

A heatmap

Author(s)

Florence Combes, Samuel Wiczorek

Examples

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

impute.detQuant	<i>Deterministic imputation</i>
-----------------	---------------------------------

Description

This method replaces each missing value by a given value

Usage

```
impute.detQuant(qData, values)
```

Arguments

qData	An expression set containing quantitative or missing values
values	A vector with as many elements as the number of columns of qData

Value

An imputed dataset

Author(s)

Thomas Burger

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
values <- getQuantile4Imp(qData)$shiftedImpVal
impute.detQuant(qData, values)
```

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 <code>MSnSet</code> .
conditions	A vector of conditions in the dataset
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 <code>q.min</code> of the observed values distribution minus <code>eps</code> . Default is 0 (the maximal value is the minimum of observed values minus <code>eps</code>).
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 $q_n \cdot \text{median}(\text{sd}(\text{observed values}))$ where <code>sd</code> 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 <code>q.min</code> of the observed values distribution minus <code>eps</code> . Default is 0.
distribution	The type of distribution used. Values are <code>unif</code> or <code>beta</code> .

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")
```

`inner.aggregate.iter` *xxxx*

Description

Method to xxxxx

Usage

```
inner.aggregate.iter(pepData, X, init.method = "Sum", method = "Mean",
  n = NULL)
```

Arguments

pepData	xxxxxx
X	xxxx
init.method	xxx
method	xxx
n	xxxx

Value

xxxxxx

Author(s)

Samuel Wieczorek `require(DAPARdata) data(Exp1_R25_pept) protID <- "Protein.group.IDs" obj.pep <- Exp1_R25_pept[1:1000] X <- BuildAdjacencyMatrix(obj.pep, protID, FALSE) DAPAR::inner.aggregate.iter(exprs(obj.pep), X)`

inner.aggregate.topn xxxx

Description

Method to xxxxx

Usage

```
inner.aggregate.topn(pepData, X, method = "Mean", n = 10)
```

Arguments

pepData	A data.frame of quantitative data
X	An adjacency matrix
method	xxxxx
n	xxxxx

Value

xxxxx

Author(s)

Samuel Wieczorek

inner.mean xxxx

Description

Method to xxxxx

Usage

```
inner.mean(pepData, X)
```

Arguments

pepData	A data.frame of quantitative data
X	An adjacency matrix

Value

xxxxx

Author(s)

Samuel Wieczorek

inner.sum	xxxx
-----------	------

Description

Method to xxxxx

Usage

```
inner.sum(pepData, X)
```

Arguments

pepData	A data.frame of quantitative data
X	An adjacency matrix

Value

A matrix

Author(s)

Samuel Wieczorek

is.MV	<i>Similar to the function is.na but focused on the equality with the missing values in the dataset (type 'POV' and 'MEC')</i>
-------	--

Description

Similar to the function is.na but focused on the equality with the missing values in the dataset (type 'POV' and 'MEC')

Usage

```
is.MV(data)
```

Arguments

data	A data.frame
------	--------------

Value

A boolean dataframe

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept
data <- Biobase::fData(obj)[,obj@experimentData@other$OriginOfValues]
is.MV(data)
```

is.OfType	<i>Similar to the function is.na but focused on the equality with the paramter 'type'.</i>
-----------	--

Description

Similar to the function `is.na` but focused on the equality with the paramter 'type'.

Usage

```
is.OfType(data, type)
```

Arguments

data	A data.frame
type	The value to search in the dataframe

Value

A boolean dataframe

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept
data <- Biobase::fData(obj)[,obj@experimentData@other$OriginOfValues]
is.OfType(data, "MEC")
```

LH1	xxxxxx
-----	--------

Description

This function is xxxxxxxx

Usage

LH1(X, y1, y2, j)

Arguments

X	an n.pep*n.prot indicator matrix.
y1	n.pep*n.samples matrice giving the observed counts for
y2	n.pep*n.samples matrice giving the observed counts for
j	the index of the protein being tested, ie which has different

Value

xxxxxxxxxxx..

Author(s)

Thomas Burger, Laurent Jacob

LH1.lm	xxxxxx
--------	--------

Description

This function is xxxxxxxx

Usage

LH1.lm(X, y1, y2, j)

Arguments

X	an n.pep*n.prot indicator matrix.
y1	n.pep*n.samples matrix giving the observed counts for
y2	n.pep*n.samples matrix giving the observed counts for
j	the index of the protein being tested, ie which has different

Value

xxxxxxxxxxx..

Author(s)

Thomas Burger, Laurent Jacob

limmaCompleteTest	<i>Computes a hierarchical differential analysis</i>
-------------------	--

Description

This function is a limmaCompleteTest

Usage

```
limmaCompleteTest(qData, sTab, comp.type = "OnevsOne")
```

Arguments

qData	A matrix of quantitative data, without any missing values.
sTab	A dataframe of experimental design (pData()).
comp.type	A string that corresponds to the type of comparison. Values are: 'OnevsOne' and 'OnevsAll'; default is 'OnevsOne'.

Value

A list of two dataframes : logFC and P_Value. The first one contains the logFC values of all the comparisons (one column for one comparison), the second one contains the pvalue of all the comparisons (one column for one comparison). The names of the columns for those two dataframes are identical and correspond to the description of the comparison.

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
limma <- limmaCompleteTest(qData,sTab)
```

listSheets	<i>This function returns the list of the sheets names in a Excel file.</i>
------------	--

Description

This function lists all the sheets of an Excel file.

Usage

```
listSheets(file)
```

Arguments

file The name of the Excel file.

Value

A vector

Author(s)

Samuel Wiczorek

make.contrast	<i>Builds the contrast matrix</i>
---------------	-----------------------------------

Description

This function builds the contrast matrix

Usage

```
make.contrast(design, condition, contrast = 1)
```

Arguments

design The data.frame which correspond to the pData function of MSnbase

condition xxxxx

contrast An integer that 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

A constrat matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
design <- make.design(Biobase::pData(Exp1_R25_pept))
conds <- Biobase::pData(Exp1_R25_pept)$Condition
make.contrast(design, conds)
```

make.design	<i>Builds the design matrix</i>
-------------	---------------------------------

Description

This function builds the design matrix

Usage

```
make.design(sTab)
```

Arguments

sTab The data.frame which correspond to the pData function of MSnbase

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
make.design(Biobase::pData(Exp1_R25_pept))
```

make.design.1	<i>Builds the design matrix for designs of level 1</i>
---------------	--

Description

This function builds the design matrix for design of level 1

Usage

```
make.design.1(sTab)
```

Arguments

sTab The data.frame which correspond to the pData function of MSnbase

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
make.design.1(Biobase::pData(Exp1_R25_pept))
```

make.design.2	<i>Builds the design matrix for designs of level 2</i>
---------------	--

Description

This function builds the design matrix for design of level 2

Usage

```
make.design.2(sTab)
```

Arguments

sTab The data.frame which correspond to the pData function of MSnbase

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
make.design.2(Biobase::pData(Exp1_R25_pept))
```

make.design.3	<i>Builds the design matrix for designs of level 3</i>
---------------	--

Description

This function builds the design matrix for design of level 3

Usage

```
make.design.3(sTab)
```

Arguments

sTab The data.frame which correspond to the pData function of MSnbase

Value

A design matrix

Author(s)

Thomas Burger, Quentin Giai-Gianetto, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
make.design.3(Biobase::pData(Exp1_R25_pept))
```

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

<code>obj</code>	An object of class <code>MSnSet</code> containing quantitative data.
<code>type</code>	Method used to choose the lines to delete. Values are : "None", "wholeMatrix", "allCond", "atLeastOneCond"
<code>th</code>	An integer value of the threshold
<code>processText</code>	A string to be included in the <code>MSnSet</code> 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 Wiczorek

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

<code>obj</code>	An object of class <code>MSnSet</code> containing quantitative data.
<code>keepThat</code>	A vector of integers which are the indices of lines to keep.
<code>processText</code>	A string to be included in the <code>MSnSet</code> 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 Wiczorek

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

<code>obj</code>	An object of class <code>MSnSet</code> containing quantitative data.
<code>type</code>	Method used to choose the lines to delete. Values are : "None", "EmptyLines", "wholeMatrix", "allCond", "atLeastOneCond"
<code>th</code>	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 Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
mvFilterGetIndices(Exp1_R25_pept, "wholeMatrix", 2)
```

mvHisto	<i>Histogram of missing values</i>
---------	------------------------------------

Description

This method plots a histogram of missing values.

Usage

```
mvHisto(qData, samplesData, conds, indLegend = "auto",  
        showValues = FALSE, palette = NULL)
```

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.
conds	A vector of the conditions (one condition 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.
palette	xxx

Value

A histogram

Author(s)

Florence Combes, Samuel Wiczorek

Examples

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

mvHisto_HC	<i>Histogram of missing values</i>
------------	------------------------------------

Description

This method plots a histogram of missing values. Same as the function mvHisto but uses the package highcharter

Usage

```
mvHisto_HC(qData, samplesData, conds, indLegend = "auto",  
           showValues = FALSE, palette = NULL)
```

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.
conds	A vector of the conditions (one condition 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.
palette	xxx

Value

A histogram

Author(s)

Florence Combes, Samuel Wiczorek

Examples

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

mvImage	<i>Heatmap of missing values</i>
---------	----------------------------------

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

```
mvImage(qData, conds)
```

Arguments

qData	A dataframe that contains quantitative data.
conds	A vector of the conditions (one condition per sample).

Value

A heatmap

Author(s)

Samuel Wieczorek, Thomas Burger

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
qData <- Biobase::exprs(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
mvImage(qData, conds)
```

mvPerLinesHisto	<i>Bar plot of missing values per lines</i>
-----------------	---

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, palette = NULL)
```

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.
palette	xxx

Value

A bar plot

Author(s)

Samuel Wiczorek

Examples

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

`mvPerLinesHistoPerCondition_HC`*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. Same as the function `mvPerLinesHistoPerCondition` but uses the package `highcharter`.

Usage

```
mvPerLinesHistoPerCondition_HC(qData, samplesData, indLegend = "auto",
  showValues = FALSE, palette = NULL)
```

Arguments

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

Value

A bar plot

Author(s)

Samuel Wiczorek

Examples

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

mvPerLinesHisto_HC *Bar plot of missing values per lines using highcharter*

Description

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

Usage

```
mvPerLinesHisto_HC(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 Wiczorek

Examples

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

my_hc_chart *Customised resetZoomButton of highcharts plots*

Description

Customise the resetZoomButton of highcharts plots.

Usage

```
my_hc_chart(hc, chartType, zoomType = "None")
```

Arguments

hc	A highcharter object
chartType	The type of the plot
zoomType	The type of the zoom (one of "x", "y", "xy", "None")

Value

A highchart plot

Author(s)

Samuel Wieczorek

Examples

```
library("highcharter")
hc <- highchart()
hc_chart(hc,type = "line")
hc_add_series(hc,data = c(29, 71, 40))
my_hc_ExportMenu(hc,filename='foo')
```

my_hc_ExportMenu

Customised contextual menu of highcharts plots

Description

Customise the contextual menu of highcharts plots.

Usage

```
my_hc_ExportMenu(hc, filename)
```

Arguments

hc	A highcharter object
filename	The filename under which the plot has to be saved

Value

A contextual menu for highcharts plots

Author(s)

Samuel Wieczorek

Examples

```
library("highcharter")
hc <- highchart()
hc_chart(hc,type = "line")
hc_add_series(hc,data = c(29, 71, 40))
my_hc_ExportMenu(hc,filename='foo')
```

nonzero	<i>Retrieve the indices of non-zero elements in sparse matrices</i>
---------	---

Description

This function retrieves the indices of non-zero elements in sparse matrices of class dgCMatrix from package Matrix. This function is largely inspired from the package RINGO

Usage

```
nonzero(x)
```

Arguments

x	A sparse matrix of class dgCMatrix
---	------------------------------------

Value

A two-column matrix

Author(s)

Samuel Wieczorek

Examples

```
library(Matrix)
mat <- Matrix(c(0,0,0,0,0,1,0,0,1,1,0,0,0,0,1),nrow=5, byrow=TRUE, sparse=TRUE)
res <- nonzero(mat)
```

pepa.test	<i>PEptide based Protein differential Abundance test</i>
-----------	--

Description

This function is PEptide based Protein differential Abundance test

Usage

```
pepa.test(X, y, n1, n2, global = FALSE, use.lm = FALSE)
```

Arguments

X	Binary q x p design matrix for q peptides and p proteins. X_(ij)=1 if peptide i belongs to protein j, 0 otherwise.
y	q x n matrix representing the log intensities of q peptides among n MS samples.
n1	number of samples under condition 1. It is assumed that the first n1 columns of y correspond to observations under condition 1.
n2	number of samples under condition 2.

<code>global</code>	if TRUE, the test statistic for each protein uses all residues, including the ones for peptides in different connected components. Can be much faster as it does not require to compute connected components. However the p-values are not well calibrated in this case, as it amounts to adding a ridge to the test statistic. Calibrating the p-value would require knowing the amplitude of the ridge, which in turns would require computing the connected components.
<code>use.lm</code>	if TRUE (and if <code>global=FALSE</code>), use <code>lm()</code> rather than the result in Proposition 1 to compute the test statistic

Value

A list of the following elements: `llr`: log likelihood ratio statistic (maximum likelihood version). `llr.map`: log likelihood ratio statistic (maximum a posteriori version). `llr.pv`: p-value for `llr`. `llr.map.pv`: p-value for `llr.map`. `mse.h0`: Mean squared error under H_0 `mse.h1`: Mean squared error under H_1 `s`: selected regularization hyperparameter for `llr.map`. `wchi2`: weight used to make `llr.map` chi2-distributed under H_0 .

Author(s)

Thomas Burger, Laurent Jacob

<code>plotPCA_Eigen</code>	<i>Plots the eigen values of PCA</i>
----------------------------	--------------------------------------

Description

Plots the eigen values of PCA

Usage

```
plotPCA_Eigen(res.pca)
```

Arguments

`res.pca` `xxx`

Value

A histogram

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
res.pca <- wrapper.pca(Exp1_R25_pept, ncp=6)
plotPCA_Eigen(res.pca)
```

plotPCA_Eigen_hc *Plots the eigen values of PCA with the highcharts library*

Description

Plots the eigen values of PCA with the highcharts library

Usage

```
plotPCA_Eigen_hc(res.pca)
```

Arguments

res.pca xxx

Value

A histogram

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
res.pca <- wrapper.pca(Exp1_R25_pept, ncp=6)
plotPCA_Eigen_hc(res.pca)
```

plotPCA_Ind *Plots individuals of PCA*

Description

Plots the individuals of PCA

Usage

```
plotPCA_Ind(res.pca, chosen.axes = c(1, 2))
```

Arguments

res.pca xxx
chosen.axes The dimensions to plot

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
res.pca <- wrapper.pca(Exp1_R25_pept)
plotPCA_Ind(res.pca)
```

plotPCA_Var

Plots variables of PCA

Description

Plots the variables of PCA

Usage

```
plotPCA_Var(res.pca, chosen.axes = c(1, 2))
```

Arguments

res.pca	xxx
chosen.axes	The dimensions to plot

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
res.pca <- wrapper.pca(Exp1_R25_pept)
plotPCA_Var(res.pca)
```

proportionConRev_HC *Barplot of proportion of contaminants and reverse*

Description

Plots a barplot of proportion of contaminants and reverse. Same as the function `proportionConRev` but uses the package `highcharter`

Usage

```
proportionConRev_HC(nBoth = 0, nCont = 0, nRev = 0, lDataset = 0)
```

Arguments

nBoth	The number of both contaminants and reverse identified in the dataset.
nCont	The number of contaminants identified in the dataset.
nRev	The number of reverse entities identified in the dataset.
lDataset	The total length (number of rows) of the dataset

Value

A barplot

Author(s)

Samuel Wieczorek

Examples

```
proportionConRev_HC(10, 20, 100)
```

rbindMSnset	<i>Similar to the function rbind but applies on two subsets of the same MSnSet object.</i>
-------------	--

Description

Similar to the function `rbind` but applies on two subsets of the same `MSnSet` object.

Usage

```
rbindMSnset(df1 = NULL, df2)
```

Arguments

df1	An object (or subset of) of class <code>MSnSet</code> . May be <code>NULL</code>
df2	A subset of the same object as <code>df1</code>

Value

An instance of class MSnSet.

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R2_pept)
df1 <- Exp1_R2_pept[1:100]
df2 <- Exp1_R2_pept[200:250]
rbindMSnset(df1, df2)
```

readExcel

This function reads a sheet of an Excel file and put the data into a data.frame.

Description

This function reads a sheet of an Excel file and put the data into a data.frame.

Usage

```
readExcel(file, extension, sheet)
```

Arguments

file	The name of the Excel file.
extension	The extension of the file
sheet	The name of the sheet

Value

A data.frame

Author(s)

Samuel Wieczorek

reIntroduceMEC	<i>Put back LAPALA into a MSnSet object</i>
----------------	---

Description

This method is used to put back the LAPALA that have been identified previously

Usage

```
reIntroduceMEC(obj, MECIndex)
```

Arguments

obj	An object of class MSnSet.
MECIndex	A data.frame that contains index of MEC (see findMECBlock).

Value

The object obj where LAPALA have been reintroduced

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
lapala <- findMECBlock(obj)
obj <- wrapper.impute.detQuant(obj)
obj <- reIntroduceMEC(obj, lapala)
```

removeLines	<i>Removes lines in the dataset based on a prefix string.</i>
-------------	---

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")
```

samLRT	xxxxxx
--------	--------

Description

This function computes a regularized version of the likelihood ratio statistic. The regularization adds a user-input fudge factor *s1* to the variance estimator. This is straightforward when using a fixed effect model (cases 'numeric' and 'lm') but requires some more care when using a mixed model.

Usage

```
samLRT(lmm.res.h0, lmm.res.h1, cc, n, p, s1)
```

Arguments

<code>lmm.res.h0</code>	a vector of object containing the estimates (used to compute the statistic) under H0 for each connected component. If the fast version of the estimator was used (as implemented in this package), <code>lmm.res.h0</code> is a vector containing averages of squared residuals. If a fixed effect model was used, it is a vector of <code>lm</code> objects and if a mixed effect model was used it is a vector or <code>lmer</code> object.
<code>lmm.res.h1</code>	similar to <code>lmm.res.h0</code> , a vector of object containing the estimates (used to compute the statistic) under H1 for each protein.
<code>cc</code>	a list containing the indices of peptides and proteins belonging to each connected component.
<code>n</code>	the number of samples used in the test
<code>p</code>	the number of proteins in the experiment
<code>s1</code>	the fudge factor to be added to the variance estimate

Value

llr.sam: a vector of numeric containing the regularized log likelihood ratio statistic for each protein.
 s: a vector containing the maximum likelihood estimate of the variance for the chosen model. When using the fast version of the estimator implemented in this package, this is the same thing as the input Imm.res.h1.
 lh1.sam: a vector of numeric containing the regularized log likelihood under H1 for each protein.
 lh0.sam: a vector of numeric containing the regularized log likelihood under H0 for each connected component.
 sample.sizes: a vector of numeric containing the sample size (number of biological samples times number of peptides) for each protein. This number is the same for all proteins within each connected component.

Author(s)

Thomas Burger, Laurent Jacob

saveParameters	<i>Saves the parameters of a tool in the pipeline of Prostar</i>
----------------	--

Description

Saves the parameters of a tool in the pipeline of Prostar

Usage

```
saveParameters(obj, name.dataset = NULL, name = NULL,
  l.params = NULL)
```

Arguments

obj	An object of class MSnSet
name.dataset	The name of the dataset
name	The name of the tool. Available values are: "Norm, Imputation, anaDiff, GO-Analysis,Aggregation"
l.params	A list that contains the parameters

Value

An instance of class MSnSet.

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
l.params=list(method="Global quantile alignment", type="overall")
saveParameters(Exp1_R25_pept, "Filtered.peptide", "Imputation",l.params)
```

 scatterplotEnrichGO_HC

A dotplot that shows the result of a GO enrichment, using the package highcharter

Description

A scatter plot of GO enrichment analysis

Usage

```
scatterplotEnrichGO_HC(ego, maxRes = 10, title = NULL)
```

Arguments

ego	The result of the GO enrichment, provides either by the function <code>enrichGO</code> in DAPAR or the function <code>enrichGO</code> of the package clusterProfiler
maxRes	The maximum number of categories to display in the plot
title	The title of the plot

Value

A dotplot

Author(s)

Samuel Wiczorek

 setMEC

Sets the MEC tag in the OriginOfValues

Description

Sets the MEC tag in the OriginOfValues

Usage

```
setMEC(obj)
```

Arguments

obj	An object of class MSnSet
-----	---------------------------

Value

An instance of class MSnSet.

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
setMEC(Exp1_R25_pept)
```

StringBasedFiltering *Removes lines in the dataset based on a prefix strings (contaminants, reverse or both).*

Description

This function removes lines in the dataset based on prefix strings (contaminants, reverse or both).

Usage

```
StringBasedFiltering(obj, idCont2Delete = NULL, prefix_Cont = NULL,
  idRev2Delete = NULL, prefix_Rev = NULL)
```

Arguments

obj	An object of class MSnSet.
idCont2Delete	The name of the column that correspond to the contaminants to filter
prefix_Cont	A character string that is the prefix for the contaminants to find in the data
idRev2Delete	The name of the column that correspond to the reverse data to filter
prefix_Rev	A character string that is the prefix for the reverse to find in the data

Value

An list of 4 items : obj : an object of class MSnSet in which the lines have been deleted deleted.both : an object of class MSnSet which contains the deleted lines corresponding to both contaminants and reverse, deleted.contaminants : n object of class MSnSet which contains the deleted lines corresponding to contaminants, deleted.reverse : an object of class MSnSet which contains the deleted lines corresponding to reverse,

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
StringBasedFiltering(Exp1_R25_pept, 'Potential.contaminant', '+', 'Reverse', '+')
```

StringBasedFiltering2 *Removes lines in the dataset based on a prefix strings.*

Description

This function removes lines in the dataset based on prefix strings.

Usage

```
StringBasedFiltering2(obj, cname = NULL, tag = NULL)
```

Arguments

obj	An object of class MSnSet.
cname	The name of the column that correspond to the line to filter
tag	A character string that is the prefix for the contaminants to find in the data

Value

An list of 4 items : obj : an object of class MSnSet in which the lines have been deleted deleted : an object of class MSnSet which contains the deleted lines

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
StringBasedFiltering2(Exp1_R25_pept, 'Potential.contaminant', '+')
```

test.design	<i>Check if xxxxxx</i>
-------------	------------------------

Description

This function check xxxxx

Usage

```
test.design(tab)
```

Arguments

tab	A data.frame which correspond to xxxxxx
-----	---

Value

A list of two items

Author(s)

Thomas Burger, Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
test.design(Biobase::pData(Exp1_R25_pept)[,1:3])
```

translatedRandomBeta *Generator of simulated values*

Description

Generator of simulated values

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)
```

univ_AnnotDbPkg	<i>Returns the totality of ENTREZ ID (gene id) of an OrgDb annotation package. Careful : org.Pf.plasmo.db : no ENTREZID but ORF</i>
-----------------	---

Description

Function to compute the "universe" argument for the enrich_GO function, in case this latter should be the entire organism. Returns all the ID of the OrgDb annotation package for the corresponding organism.

Usage

```
univ_AnnotDbPkg(orgdb)
```

Arguments

orgdb a Bioconductor OrgDb annotation package

Value

A vector of ENTREZ ID

Author(s)

Florence Combes

violinPlotD	<i>Builds a violinplot from a dataframe</i>
-------------	---

Description

ViolinPlot for quantitative proteomics data

Usage

```
violinPlotD(obj, legend = NULL, palette = NULL)
```

Arguments

obj xxx
legend A vector of the conditions (one condition per sample).
palette xxx

Value

A violinplot

Author(s)

Samuel Wiczorek

See Also[densityPlotD](#)**Examples**

```
require(DAPARdata)
data(Exp1_R25_pept)
library(vioplot)
legend <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
violinPlotD(Exp1_R25_pept, legend=legend)
```

```
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, condsForLegend = NULL,
  indData2Show = NULL, ...)
```

Arguments

objBefore	A dataframe that contains quantitative data before normalization.
objAfter	A dataframe that contains quantitative data after normalization.
condsForLegend	A vector of the conditions (one condition 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.
...	arguments for palette

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]
objAfter <- wrapper.normalized(Exp1_R25_pept, "QuantileCentering","within conditions")
wrapper.compareNormalizationD(Exp1_R25_pept, objAfter, conds)
```

```
wrapper.compareNormalizationD_HC
```

Builds a plot from a dataframe

Description

Wrapper to the function that plot to compare the quantitative proteomics data before and after normalization. Same as the function [wrapper.compareNormalizationD](#) but uses the package `highcharter`

Usage

```
wrapper.compareNormalizationD_HC(objBefore, objAfter,  
  condsForLegend = NULL, indData2Show = NULL, ...)
```

Arguments

<code>objBefore</code>	A dataframe that contains quantitative data before normalization.
<code>objAfter</code>	A dataframe that contains quantitative data after normalization.
<code>condsForLegend</code>	A vector of the conditions (one condition per sample).
<code>indData2Show</code>	A vector of the indices of the columns to show in the plot. The indices are those of indices of the columns in the data.frame <code>qDataBefore</code> .
<code>...</code>	arguments for palette

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)  
data(Exp1_R25_pept)  
conds <- Biobase::pData(Exp1_R25_pept)[,"Condition"]  
objAfter <- wrapper.normalized(Exp1_R25_pept, "QuantileCentering",  
  "within conditions")  
wrapper.compareNormalizationD_HC(Exp1_R25_pept, objAfter, conds)
```

`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.corrMatrixD_HC` *Displays a correlation matrix of the quantitative data of the exprs() table*

Description

Builds a correlation matrix based on a MSnSet object. Same as the function `wrapper.corrMatrixD` but uses the package `highcharter`

Usage

```
wrapper.corrMatrixD_HC(obj, rate = 0.5)
```

Arguments

`obj` An object of class MSnSet.
`rate` A float that defines the gradient of colors.

Value

A colored correlation matrix

Author(s)

Samuel Wieczorek

Examples

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

wrapper.CVDistD	<i>Distribution of CV of entities</i>
-----------------	---------------------------------------

Description

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

Usage

```
wrapper.CVDistD(obj, ...)
```

Arguments

<code>obj</code>	An object of class <code>MSnSet</code> .
<code>...</code>	arguments for palette

Value

A density plot

Author(s)

Alexia Dorffer

Examples

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

wrapper.CVDistD_HC *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 present in the dataset (see the slot 'Condition' in the pData() table). Same as the function `wrapper.CVDistD` but uses the package `highcharter`

Usage

```
wrapper.CVDistD_HC(obj, ...)
```

Arguments

obj An object of class MSnSet
... arguments for palette.

Value

A density plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.CVDistD_HC(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 <code>mi.mix</code> in the package <code>imp4p</code>
nknn	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
selec	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
siz	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
weight	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
ind.comp	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
progress.bar	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
x.step.mod	Same as the function <code>estim.mix</code> in the package <code>imp4p</code>
x.step.pi	Same as the function <code>estim.mix</code> in the package <code>imp4p</code>
nb.rei	Same as the function <code>estim.mix</code> in the package <code>imp4p</code>
method	Same as the function <code>estim.mix</code> in the package <code>imp4p</code>
gridsize	Same as the function <code>estim.mix</code> in the package <code>imp4p</code>
q	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
q.min	Same as the function <code>impute.pa</code> in the package <code>imp4p</code>
q.norm	Same as the function <code>impute.pa</code> in the package <code>imp4p</code>
eps	Same as the function <code>impute.pa</code> in the package <code>imp4p</code>
methodi	Same as the function <code>mi.mix</code> in the package <code>imp4p</code>
lapala	xxxxxxxxxxx
distribution	The type of distribution used. Values are <code>unif</code> (default) or <code>beta</code> .

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.hc_mvTypePlot2

Distribution of observed values with respect to intensity values from a MSnSet object

Description

This method is a wrapper for the function [hc_mvTypePlot2](#) adapted to objects of class MSnSet).

Usage

```
wrapper.hc_mvTypePlot2(obj, ...)
```

Arguments

obj An object of class MSnSet.
... See [hc_mvTypePlot2](#)

Value

A scatter plot

Author(s)

Florence Combes, Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.hc_mvTypePlot2(Exp1_R25_pept)
```

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 = "complete",
  dendro = FALSE)
```

Arguments

obj	An object of class MSnSet.
distance	The distance used by the clustering algorithm to compute the dendrogram. See <code>help(heatmap.2)</code> .
cluster	the clustering algorithm used to build the dendrogram. See <code>help(heatmap.2)</code>
dendro	A boolean to indicate if 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.detQuant

Wrapper of the function [impute.detQuant](#) for objects of class MSnSet

Description

This method is a wrapper of the function [impute.detQuant](#) for objects of class MSnSet

Usage

```
wrapper.impute.detQuant(obj, qval = 0.025, factor = 1)
```

Arguments

obj	An instance of class MSnSet
qval	An expression set containing quantitative values of various replicates
factor	A scaling factor to multiply the imputation value with

Value

An imputed instance of class MSnSet

Author(s)

Samuel Wiczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.impute.detQuant(Exp1_R25_pept)
```

```
wrapper.impute.fixedValue
```

Missing values imputation from a MSnSet object

Description

This method is a wrapper to objects of class MSnSet and imputes missing values with a fixed value.

Usage

```
wrapper.impute.fixedValue(obj, fixVal)
```

Arguments

obj	An object of class MSnSet.
fixVal	A float .

Value

The object obj which has been imputed

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.impute.fixedValue(Exp1_R25_pept[1:1000], 0.001)
```

```
wrapper.impute.KNN
```

KNN missing values imputation from a MSnSet object

Description

This method is a wrapper for objects of class MSnSet and imputes missing values with a fixed value. This function imputes the missing values condition by condition.

Usage

```
wrapper.impute.KNN(obj, K)
```

Arguments

obj	An object of class MSnSet.
K	the number of neighbors.

Value

The object obj which has been imputed

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
wrapper.impute.KNN(Exp1_R25_pept[1:1000], 3)
```

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

Description

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

Usage

```
wrapper.impute.mle(obj)
```

Arguments

`obj` An object of class `MSnSet`.

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.mle(dat)
```

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 Wiczorek

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 $qn * \text{median}(sd(\text{observed values}))$ 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 (default) or beta.

Value

The object obj which has been imputed

Author(s)

Thomas Burger, Samuel Wiczorek

Examples

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

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

Description

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

Usage

```
wrapper.impute.slsa(obj)
```

Arguments

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

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.slsa(dat)
```

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.mvHisto_HC *Histogram of missing values from a MSnSet object*

Description

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

Usage

```
wrapper.mvHisto_HC(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.
...	xxx

Value

A histogram

Author(s)

Alexia Dorffer

Examples

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

<code>obj</code>	An object of class MSnSet.
<code>indLegend</code>	The indice of the column name's in <code>pData()</code> tab .
<code>showValues</code>	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.mvPerLinesHistoPerCondition_HC`

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

Description

This method is a wrapper to plots (using highcharts) 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_HC(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.
...	xxx

Value

A bar plot

Author(s)

Samuel Wieczorek

Examples

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

```
wrapper.mvPerLinesHisto_HC
```

*Histogram of missing values per lines from an object using highchar-
ter 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_HC(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.normalizeD *Normalisation*

Description

Provides several methods to normalize quantitative data from a MSnSet object. They are organized in six main families : GlobalQuantileAlignment, sumByColumns, QuantileCentering, MeanCentering, LOESS, vsn For the first family, there is no type. For the five other families, two type 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, method, type = NULL, scaling = FALSE,  
  quantile = 0.15, span = 0.7)
```

Arguments

obj	An object of class MSnSet.
method	One of the following : "GlobalQuantileAlignment" (for normalizations of important magnitude), "SumByColumns", "QuantileCentering", "Mean Centering", "LOESS" and "vsn".
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.
span	parameter for LOESS method

Value

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

Author(s)

Samuel Wiczorek, Thomas Burger, Helene Borges

Examples

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

wrapper.pca

Compute the PCA

Description

Compute the PCA

Usage

```
wrapper.pca(obj, var.scaling = TRUE, ncp = NULL)
```

Arguments

obj	xxx
var.scaling	The dimensions to plot
ncp	xxxx

Value

A xxxxxx

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
res.pca <- wrapper.pca(Exp1_R25_pept)
```

wrapper.t_test_Completexxxxx

Description

This function is a wrapper xxxxx

Usage

```
wrapper.t_test_Complete(obj, ...)
```

Arguments

obj	An object of class MSnSet with no missing values
...	See compute.t.tests

Value

xxxxxxx

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
lapala <- findMECBlock(obj)
obj <- wrapper.impute.detQuant(obj)
obj <- reIntroduceMEC(obj, lapala)
obj <- wrapper.impute.detQuant(obj)
ttest <- wrapper.t_test_Complete(obj, 1)
```

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 (one condition per sample).

Value

A plot

Author(s)

Samuel Wieczorek

Examples

```
require(DAPARdata)
data(Exp1_R25_pept)
obj <- Exp1_R25_pept[1:1000]
keepThat <- mvFilterGetIndices(obj, 'wholeMatrix', ncol(obj))
obj <- mvFilterFromIndices(obj, keepThat)
qData <- Biobase::exprs(obj)
sTab <- Biobase::pData(obj)
limma <- limmaCompleteTest(qData,sTab)
wrapperCalibrationPlot(limma$P_Value[,1])
```

writeMSnsetToCSV	<i>Exports a MSnset dataset into a zip archive containing three zipped CSV files.</i>
------------------	---

Description

This function exports a MSnset dataset into three csv files compressed in a zip file

Usage

```
writeMSnsetToCSV(obj, fname)
```

Arguments

obj	An object of class MSnSet.
fname	The name of the archive file.

Value

A compressed file

Author(s)

Samuel Wieczorek

Examples

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

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). The colored cells in the experimental data correspond to the original missing values which have been imputed.

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

```
Sys.setenv("R_ZIPCMD"= Sys.which("zip"))
require(DAPARdata)
data(Exp1_R2_pept)
obj <- Exp1_R2_pept[1:1000]
writeMSnsetToExcel(obj, "foo")
```

Index

addOriginOfValue, 5
aggregateIter, 5
aggregateIterParallel, 6
aggregateMean, 7
aggregateSum, 7
aggregateTopn, 8

barplotEnrichGO_HC, 9
barplotGroupGO_HC, 9
bitr, 45
boxPlotD, 10, 23, 24
boxPlotD_HC, 11
BuildAdjacencyMatrix, 11
BuildColumnToProteinDataset, 12
BuildColumnToProteinDataset_par, 13

check.conditions, 14
check.design, 14
clusterProfiler, 9, 10, 30, 45, 46, 81
compareNormalizationD, 15
compareNormalizationD_HC, 16
compute.t.tests, 17
corrMatrixD, 18, 18
corrMatrixD_HC, 18
CountPep, 19
createMSnset, 20
CVDistD, 21, 23, 24
CVDistD_HC, 22

deleteLinesFromIndices, 22
densityPlotD, 10, 21, 22, 23, 24, 86
densityPlotD_HC, 11, 24
diffAnaComputeFDR, 25
diffAnaGetSignificant, 26
diffAnaSave, 26
diffAnaVolcanoplot, 27
diffAnaVolcanoplot_rCharts, 28

enrich_GO, 30
enrichGO, 45

finalizeAggregation, 31
findMECBlock, 31
formatLimmaResult, 32
fudge2LRT, 33

getIndicesConditions, 34
getIndicesOfLinesToRemove, 34
getListNbValuesInLines, 35
GetNbPeptidesUsed, 36
getNumberOf, 36
getNumberOfEmptyLines, 37
getPourcentageOfMV, 37
getProcessingInfo, 38
getProteinsStats, 39
getQuantile4Imp, 39
getTextForAggregation, 40
getTextForAnaDiff, 41
getTextForFiltering, 41
getTextForGOAnalysis, 42
getTextForHypothesisTest, 42
getTextForNewDataset, 43
getTextForNormalization, 43
getTextForpeptideImputation, 44
getTextForproteinImputation, 44
GOAnalysisSave, 45
GraphPepProt, 46
group_GO, 46
groupGO, 45

hc_logFC_DensityPlot, 47
hc_mvTypePlot2, 48, 92
heatmap.2, 49, 50, 92
heatmap.DAPAR, 49
heatmapD, 50

impute.detQuant, 50, 93
impute.pa2, 51
inner.aggregate.iter, 52
inner.aggregate.topn, 53
inner.mean, 53
inner.sum, 54
is.MV, 54
is.OfType, 55

LH0, 56
LH0.lm, 56
LH1, 57
LH1.lm, 57
limma, 26

limmaCompleteTest, 25, 27, 58
listSheets, 58

make.contrast, 59
make.design, 60
make.design.1, 60
make.design.2, 61
make.design.3, 61
mvFilter, 62
mvFilterFromIndices, 63
mvFilterGetIndices, 64
mvHisto, 65
mvHisto_HC, 66
mvImage, 67
mvPerLinesHisto, 67
mvPerLinesHisto_HC, 70
mvPerLinesHistoPerCondition, 68, 69
mvPerLinesHistoPerCondition_HC, 69
my_hc_chart, 70
my_hc_ExportMenu, 71

nonzero, 72

pepa.test, 72
plotPCA_Eigen, 73
plotPCA_Eigen_hc, 74
plotPCA_Ind, 74
plotPCA_Var, 75
proportionConRev_HC, 76

rbindMSnset, 76
readExcel, 77
reIntroduceMEC, 78
removeLines, 78

samLRT, 79
saveParameters, 80
scatterplotEnrichGO_HC, 81
setMEC, 81
StringBasedFiltering, 82
StringBasedFiltering2, 83

test.design, 83
translatedRandomBeta, 84

univ_AnnotDbPkg, 85

violinPlotD, 85

wrapper.compareNormalizationD, 86, 87
wrapper.compareNormalizationD_HC, 87
wrapper.corrMatrixD, 88, 88
wrapper.corrMatrixD_HC, 88
wrapper.CVDistD, 89, 90
wrapper.CVDistD_HC, 90
wrapper.dapar.impute.mi, 90
wrapper.hc_mvTypePlot2, 92
wrapper.heatmapD, 92
wrapper.impute.detQuant, 93
wrapper.impute.fixedValue, 94
wrapper.impute.KNN, 94
wrapper.impute.mle, 95
wrapper.impute.pa, 96
wrapper.impute.pa2, 96
wrapper.impute.slsa, 97
wrapper.mvHisto, 98
wrapper.mvHisto_HC, 99
wrapper.mvImage, 99
wrapper.mvPerLinesHisto, 100
wrapper.mvPerLinesHisto_HC, 102
wrapper.mvPerLinesHistoPerCondition,
101
wrapper.mvPerLinesHistoPerCondition_HC,
101
wrapper.normalizedD, 103
wrapper.pca, 104
wrapper.t_test_Complete, 104
wrapperCalibrationPlot, 105
writeMSnsetToCSV, 106
writeMSnsetToExcel, 107