Package 'qPLEXanalyzer'

May 10, 2024

Type Package

Title Tools for quantitative proteomics data analysis

Version 1.22.0

Date 2024-04-26

Description Tools for TMT based quantitative proteomics data analysis.

License GPL-2

Imports assertthat, BiocGenerics, Biostrings, dplyr (>= 1.0.0), ggdendro, ggplot2, graphics, grDevices, IRanges, limma, magrittr, preprocessCore, purrr, RColorBrewer, readr, rlang, scales, stats, stringr, tibble, tidyr, tidyselect, utils

Depends R (>= 4.0), Biobase, MSnbase

Suggests gridExtra, knitr, qPLEXdata, rmarkdown, statmod, testthat, UniProt.ws, vdiffr

VignetteBuilder knitr

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

BugReports https://github.com/crukci-bioinformatics/qPLEXanalyzer/issues

Encoding UTF-8

RoxygenNote 7.2.1

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

git_branch RELEASE_3_19

git_last_commit 249d9c8

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-09

Author Matthew Eldridge [aut], Kamal Kishore [aut], Ashley Sawle [aut, cre]

Maintainer Ashley Sawle <ads2202cu@gmail.com>

Contents

`qPLEXanalyzer-package` *Tools for qPLEX-RIME data analysis*

Description

Tools for quantitiative proteomics data analysis generated from qPLEX-RIME method The package offers the following functionalities Data processing, normalization & analysis:

- `convertToMSnset`: Converts quantitative data to a MSnSet
- `summarizeIntensities`: Summarizes multiple peptide measurements for a protein
- `normalizeQuantiles`: Performs quantile normalization on the peptides/proteins intensities
- `normalizeScaling`: Performs scaling normalization on the peptides/proteins intensities (mean, median or sum)

- groupScaling: Performs scaling normalization on the peptides/proteins intensities within group (median or mean)
- rowScaling: Normalization by scaling peptide/protein intensity across all samples
- regressIntensity: Performs linear regression on protein intensities based on selected protein
- computeDiffStats: Compute differential statistics for the given contrasts
- getContrastResults: Get differential statistics results for given contrast

Visualization:

- assignColours: Assigns colours to samples in groups
- corrPlot: Correlation plot of all the samples
- coveragePlot: Computes and display protein sequence coverage of
- hierarchicalPlot: Hierarchical clustering plot of all the samples
- intensityBoxplot: Intensity distribution boxplot of all the samples
- intensityPlot: Intensity distribution plot of all the samples
- maVolPlot: MA or Volcano plot of differential analysis results
- pcaPlot: PCA plot of all the samples
- peptideIntensityPlot: Peptide intensity distribution plot of specific protein
- plotMeanVar: Computes and plots mean-variance for samples in MSnSet
- rliPlot: Relative intensity plot of all the samples selected protein in proteomics experiment

Author(s)

Matthew Eldridge, Kamal Kishore, Ashley Sawle (Maintainer)
<ads2202cu@gmail.com>

`assignColours` *Assigns colours to samples in groups*

Description

Assigns colours to samples in groups for plotting

Usage

```
assignColours(MSnSetObj, colourBy = "SampleGroup")
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
colourBy	character: column name from pData(MSnSetObj) to use for coloring samples

Value

A character vector of colors for samples.

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
metadata=exp3_OHT_ESR1$metadata_qPLEX1,
indExpData=c(7:16), Sequences=2, Accessions=6)
sampleColours <- assignColours(MSnSet_data)
```

---

computeDiffStats *Compute differential statistics*

Description

Compute differential statistics on the given contrasts, based on limma functions.

Usage

```
computeDiffStats(
  MSnSetObj,
  batchEffect = NULL,
  transform = TRUE,
  contrasts,
  trend = TRUE,
  robust = TRUE
)
```

Arguments

MSnSetObj	MSnSet; An object of class MSnSet
batchEffect	character; vector of variable(s) to correct for batch effect, Default : "SampleGroup"
transform	logical; apply log2 transformation to the raw intensitites
contrasts	character; named character vector of contrasts for differential statistics
trend	logical; TRUE or FALSE
robust	logical; TRUE or FALSE

Details

A statistical analysis for the identification of differentially regulated or bound proteins is carried out using limma based analysis. It uses linear models to assess differential expression in the context of multifactor designed experiments. Firstly, a linear model is fitted for each protein where the model includes variables for each group and MS run. Then, log2 fold changes between comparisions are estimated. Multiple testing correction of p-values are applied using the Benjamini-Hochberg method to control the false discovery rate (FDR).

In order to correct for batch effect, variable(s) can be defined. It should corresponds to a column name in pData(MSnSetObj). The default variable is "SampleGroup" that distinguish between two groups. If more variables are defined they are added to default.

Value

A list object containing three components: MSnSetObj of class `MSnSet` (see `MSnSet-class`) object), fittedLM (fitted linear model) and fittedContrasts. This object should be input into getContrastResults function to get differential results. See `eBayes` function of `limma` for more details on differential statistics.

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- groupScaling(MSnSet_data, scalingFunction=median)
MSnset_Pnorm <- summarizeIntensities(MSnset_norm, sum, human_anno)
contrasts <- c(tam.24h_vs_vehicle = "tam.24h - vehicle",
               tam.6h_vs_vehicle = "tam.6h - vehicle")
diffstats <- computeDiffStats(MSnSetObj=MSnset_Pnorm, contrasts=contrasts)
```

`convertToMSnset` *Converts proteomics TMT intensity data to MSnSet*

Description

Converts processed TMT peptide intensities to MSnSet

Usage

```
convertToMSnset(
  ExpObj,
  metadata,
  indExpData,
  Sequences = NULL,
```

```
  Accessions,
  type = "peptide",
  rmMissing = TRUE
)
```

Arguments

ExpObj	data.frame; a data.frame consisting of quantitative peptide intensities and peptide annotation
metadata	data.frame; a data.frame describing the samples
indExpData	numeric; a numeric vector indicating the column indexes of intensities in ExpObj
Sequences	numeric; a numeric value indicating the index of column consisting of peptide sequence in ExpObj
Accessions	numeric; a numeric value indicating the index of column consisting of protein accession in ExpObj
type	character; what type of data set to create, either 'peptide' or 'protein'
rmMissing	logical; TRUE or FALSE to indicate whether to remove missing data or not

Details

This function builds an object of class MSnSet from a dataframe consisting of quantitative proteomics intensities data and a meta-data describing the samples information. This function creates an MSnSet object from the intensities and metadata file. The metadata must contain "SampleName", "SampleGroup", "BioRep" and "TechRep" columns. The function can be used for either peptide intensities or data that has already been summarized to protein level. The type argument should be set to 'protein' for the latter.

Value

An object of class MSnSet (see MSnSet-class) object).

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
```

corrPlot *Correlation plot*

Description

Computes and display correlation plot for samples within MSnSet

Usage

```
corrPlot(
  MSnSetObj,
  addValues = TRUE,
  title = "",
  low_cor_colour = "#FFFFFF",
  high_cor_colour = "#B90505",
  low_cor_limit = 0,
  high_cor_limit = 1,
  textsize = 3
)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
addValues	logical: adds correlation values to the plot
title	character; title of the plot
low_cor_colour	colour; colour for lowest correlation in scale
high_cor_colour	colour; colour for highest correlation in scale
low_cor_limit	numeric; lower limit for correlation in colour scale
high_cor_limit	numeric; upper limit for correlation in colour scale
textsize	integer: set the size of correlation values text

Value

An object created by ggplot

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
    metadata=exp3_OHT_ESR1$metadata_qPLEX1,
    indExpData=c(7:16),
    Sequences=2,
    Accessions=6)
corrPlot(MSnSet_data, addValues=TRUE, title="Correlation plot")
```

```
# change colours
corrPlot(MSnSet_data, addValues=TRUE, title="Correlation plot",
    low_cor_colour="yellow", high_cor_colour="pink")
```

`coveragePlot` *Plot peptide sequence coverage*

Description

Computes and displays peptide sequence coverage in proteomics experiment

Usage

```
coveragePlot(MSnSetObj, ProteinID, ProteinName, fastaFile, myCol = "brown")
```

Arguments

`MSnSetObj`	MSnSet: an object of class MSnSet
`ProteinID`	character: Uniprot ID of the protein
`ProteinName`	character: name of the protein
`fastaFile`	character: fasta file of protein sequence
`myCol`	colour: colour for plotting

Details

In the qPLEX-RIME experiment it is imperative for bait protein to have good sequence coverage. This function plots the protein sequence coverage of the bait protein in the qPLEX-RIME experiment. It requires the fasta sequence file of bait protein as input to generate the plot.

Value

An object created by `ggplot`

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
mySequenceFile <- system.file('extdata',
                              "P03372.fasta",
                              package="qPLEXanalyzer")
coveragePlot(MSnSet_data,
```

```
ProteinID="P03372",
ProteinName="ERa",
fastaFile=mySequenceFile)
```

ER_ARID1A_KO_MCF7 *ER_ARID1A_KO_MCF7 dataset*

Description

Five ER qPLEX-RIME (9plex) experiments were performed on two wild type clones, two ARID1A knockout clones and one parental cell line with Tamoxifen treatment in MCF7 cell lines.

Format

An object of class `list` related to peptides quantification. It consists of qPLEX-RIME data from five experimental runs. Each run contains 9 samples divided into nine conditions (T_14, V_14, T_11, V_11, ECACC.T, ECACC.V, T_221, V_221 and Ref).

Value

An object of class `list` related to peptides quantification.

---

exp2_Xlink *exp2_Xlink dataset*

Description

An ER qPLEX-RIME experiment was performed to compare two different methods of crosslinking. MCF7 cells were double crosslinked with DSG/formaldehyde (double) or with formaldehyde alone (single). Four biological replicates were obtained for each condition along with two IgG pooled samples from each replicate.

Format

An object of class `list` related to peptides quantification. It consists of qPLEX-RIME data of 10 samples divided into three conditions (FA, DSG.FA and IgG).

Value

An object of class `list` related to peptides quantification.

exp3_OHT_ESR1 *exp3_OHT_ESR1 dataset*

Description

Three ER qPLEX-RIME (10plex) experiments were performed to investigate the dynamics of the ER complex assembly upon 4-hydrotamoxifen (OHT) treatment at 2h, 6h and 24h or at 24h post-treatment with the drug-vehicle alone (ethanol). Two biological replicates of each condition were included in each experiment to finally consider a total of six replicates per time point. Additionally, MCF7 cells were treated with OHT or ethanol and cross-linked at 24h post-treatment in each experiment to be used for mock IgG pull-downs and to enable discrimination of non-specific binding in the same experiment. This is a timecourse experiment to study the effect of tamoxifen in ER interactome using qPLEX-RIME method.

Format

An object of class `list` related to peptides quantification. It consists of qPLEX-RIME data from three experimental runs. Each run contains 10 samples divided into five conditions (IgG, vehicle, tam.2h, tam.6h and tam.24h).

Value

An object of class `list` related to peptides quantification.

getContrastResults *Get differential statistics results*

Description

Get differential statistics results for given contrasts.

Usage

```
getContrastResults(
  diffstats,
  contrast,
  controlGroup = NULL,
  transform = TRUE,
  writeFile = FALSE
)
```

Arguments

diffstats	list; output of computeDiffStats function
contrast	character; contrast of interest for which to retrieve differential statistics results
controlGroup	character; control group such as IgG
transform	logical; apply log2 transformation to the raw intensities
writeFile	logical; whether to write the results into a text file

Value

A data.frame object and text file containing the result of the differential statistics.

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- groupScaling(MSnSet_data, scalingFunction=median)
MSnset_Pnorm <- summarizeIntensities(MSnset_norm, sum, human_anno)
contrasts <- c(tam.24h_vs_vehicle = "tam.24h - vehicle")
diffstats <- computeDiffStats(MSnset_Pnorm, contrasts=contrasts)
diffexp <- getContrastResults(diffstats=diffstats, contrast=contrasts)
```

---

groupScaling *Normalization by scaling within group*

Description

Performs scaling normalization on the intensities within group (median or mean)

Usage

```
groupScaling(
  MSnSetObj,
  scalingFunction = median,
  groupingColumn = "SampleGroup"
)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
scalingFunction	function; median or mean
groupingColumn	character; the feature on which groups would be based; default="SampleGroup"

Details

In this normalization method the central tendencies (mean or median) of the samples within groups are aligned. The argument "groupingColumn" is used to define separate groups to normalize. The function takes one of the column of pData(data) as the variable for classifying group. The default variable is "SampleGroup". It is imperative in qPLEX-RIME experiment to define IgG as a separate group and normalize it separately from others. You could add a column into the metadata to define this classification.

Value

An object of class `MSnSet` (see `MSnSet-class`)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- groupScaling(MSnSet_data,
                            scalingFunction=median,
                            groupingColumn="SampleGroup")
```

`hierarchicalPlot` *Hierarchical clustering plot*

Description

Computes and displays hierarchical clustering plot for samples in MSnSet

Usage

```
hierarchicalPlot(
  MSnSetObj,
  sampleColours = NULL,
  colourBy = "SampleGroup",
  horizontal = TRUE,
  title = ""
)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`sampleColours`	character: a named vector of colors for samples, names should be values of `colourBy` column

colourBy	character: column name from pData(MSnSetObj) to use for coloring samples
horizontal	logical: define orientation of the dendrogram
title	character: the main title for the dendrogram

Value

An object created by ggplot

Examples

```
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
exprs(MSnSet_data) <- exprs(MSnSet_data)+0.01
hierarchicalPlot(MSnSet_data, title="qPLEX_RIME_ER")
```

---

human_anno *human_anno dataset*

Description

Uniprot Human protein annotation table.

Format

An object of class data.frame consisting of uniprot human protein annotation.

---

intensityBoxplot *Intensity Distribution boxplot*

Description

Intensity distribution boxplot of all the samples

Usage

```
intensityBoxplot(
  MSnSetObj,
  title = "",
  sampleColours = NULL,
  colourBy = "SampleGroup"
)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
title	character; title of the plot
sampleColours	character: a named character vector of colors for samples
colourBy	character: column name from pData(MSnSetObj) to use for coloring samples

Details

The column provided to the colourBy argument will be used to colour the samples. The colours will be determined using the function assignColours, alternatively the user may specify a named vector of colours using the sampleColours argument. The names of the sampleColours vector should match the unique values in the colourBy column.

Value

An object created by ggplot

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
intensityBoxplot(MSnSet_data, title = "qPLEX_RIME_ER")

# custom colours
customCols <- rainbow(length(unique(pData(MSnSet_data)$SampleGroup)))
names(customCols) <- unique(pData(MSnSet_data)$SampleGroup)
intensityBoxplot(MSnSet_data,
                 title = "qPLEX_RIME_ER",
                 sampleColours = customCols)
```

---

intensityPlot *Intensity Distribution Plot*

Description

Intensity distribution plot of all the samples

Usage

```
intensityPlot(
  MSnSetObj,
  sampleColours = NULL,
  title = "",
  colourBy = "SampleGroup",
  transform = TRUE,
  xlab = "log2(intensity)",
  trFunc = log2xplus1
)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
sampleColours	character: a vector of colors for samples
title	character: title for the plot
colourBy	character: column name from pData(MSnSetObj) to use for coloring samples
transform	logical: whether to log transform intensities
xlab	character: label for x-axis
trFunc	func: internal helper function for log transformation

Details

The column provided to the colourBy argument will be used to colour the samples. The colours will be determined using the function assignColours, alternatively the user may specify a named vector of colours using the sampleColours argument. The names of the sampleColours vector should match the unique values in the colourBy column.

Value

An object created by ggplot

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
intensityPlot(MSnSet_data, title = "qPLEX_RIME_ER")

# custom colours
customCols <- rainbow(length(unique(pData(MSnSet_data)$SampleGroup)))
names(customCols) <- unique(pData(MSnSet_data)$SampleGroup)
intensityPlot(MSnSet_data,
              title = "qPLEX_RIME_ER",
```

```
                sampleColours = customCols)
```

IRSnorm *Batch Correction by Internal Reference Scale*

Description

Performs batch correction on multiple runs using an Internal Reference Scale sample.

Usage

```
IRSnorm(MSnSetObj, IRSname = "RefPool", groupingColumn = "Plex")
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
IRSname	character; name of the Reference group within the SampleGroup column
groupingColumn	character; the pData(MSnSetObj) column name used to define batches; default="Plex"

Details

The Internal Reference Scale sample (IRS) should ideally be representative of the entire proteome detectable across all sample in the experiment, e.g. a pooled sample made up of aliquots of protein from all samples. The IRS is then run and measured in each TMT experiment. The normalization procedure makes measurements of the IRS from different TMT batches exactly the same, and puts all of the reporter ions on the same "intensity scale". The argument 'IRSname' is used to define the name of the Reference group within the SampleGroup column. The argument "groupingColumn" takes one of the column of pData(MSnSetObj) to define separate batches to correct; the default variable name is "Plex".

Value

An object of class MSnSet (see MSnSet-class)

Examples

```
data(human_anno)
data(ER_ARID1A_KO_MCF7)
MSnset_SET1 <- convertToMSnset(ER_ARID1A_KO_MCF7$intensities_Set1,
                               metadata=ER_ARID1A_KO_MCF7$metadata_Set1,
                               indExpData=c(7:15),
                               Sequences=2,
                               Accessions=6)
MSnset_SET2 <- convertToMSnset(ER_ARID1A_KO_MCF7$intensities_Set2,
                               metadata=ER_ARID1A_KO_MCF7$metadata_Set2,
                               indExpData=c(7:15),
```

```
                              Sequences=2,
                              Accessions=6)
MSnset_SET3 <- convertToMSnset(ER_ARID1A_KO_MCF7$intensities_Set3,
                              metadata=ER_ARID1A_KO_MCF7$metadata_Set3,
                              indExpData=c(7:15),
                              Sequences=2,
                              Accessions=6)
MSnset_SET4 <- convertToMSnset(ER_ARID1A_KO_MCF7$intensities_Set4,
                              metadata=ER_ARID1A_KO_MCF7$metadata_Set4,
                              indExpData=c(7:14),
                              Sequences=2,
                              Accessions=6)
MSnset_SET5 <- convertToMSnset(ER_ARID1A_KO_MCF7$intensities_Set5,
                              metadata=ER_ARID1A_KO_MCF7$metadata_Set5,
                              indExpData=c(7:15),
                              Sequences=2,
                              Accessions=6)
MSnset_SET1_norm <- normalizeScaling(MSnset_SET1, median)
MSnset_SET2_norm <- normalizeScaling(MSnset_SET2, median)
MSnset_SET3_norm <- normalizeScaling(MSnset_SET3, median)
MSnset_SET4_norm <- normalizeScaling(MSnset_SET4, median)
MSnset_SET5_norm <- normalizeScaling(MSnset_SET5, median)
MSnset_SET1_Pnorm <- summarizeIntensities(MSnset_SET1_norm, sum, human_anno)
MSnset_SET2_Pnorm <- summarizeIntensities(MSnset_SET2_norm, sum, human_anno)
MSnset_SET3_Pnorm <- summarizeIntensities(MSnset_SET3_norm, sum, human_anno)
MSnset_SET4_Pnorm <- summarizeIntensities(MSnset_SET4_norm, sum, human_anno)
MSnset_SET5_Pnorm <- summarizeIntensities(MSnset_SET5_norm, sum, human_anno)
MSnset_SET1_Pnorm <- updateSampleNames(updateFvarLabels(MSnset_SET1_Pnorm))
MSnset_SET2_Pnorm <- updateSampleNames(updateFvarLabels(MSnset_SET2_Pnorm))
MSnset_SET3_Pnorm <- updateSampleNames(updateFvarLabels(MSnset_SET3_Pnorm))
MSnset_SET4_Pnorm <- updateSampleNames(updateFvarLabels(MSnset_SET4_Pnorm))
MSnset_SET5_Pnorm <- updateSampleNames(updateFvarLabels(MSnset_SET5_Pnorm))
MSnset_comb <- MSnbase::combine(MSnset_SET1_Pnorm,
                               MSnset_SET2_Pnorm,
                               MSnset_SET3_Pnorm,
                               MSnset_SET4_Pnorm,
                               MSnset_SET5_Pnorm)
tokeep <- complete.cases(fData(MSnset_comb))
MSnset_comb <- MSnset_comb[tokeep,]
sampleNames(MSnset_comb) <- pData(MSnset_comb)$SampleName
fData(MSnset_comb) <- fData(MSnset_comb)[,c(2,3,6)]
colnames(fData(MSnset_comb)) <- c("Sequences", "Modifications", "Accessions")
MSnset_comb_corr <- IRSnorm(MSnset_comb, IRSname="Ref", groupingColumn="Run")
```

`maVolPlot` *MA or Volcano Plot*

Description

MA or Volcano plot of differential statistics results

Usage

```
maVolPlot(
  diffstats,
  contrast,
  title = "",
  controlGroup = NULL,
  selectedGenes = NULL,
  fdrCutOff = 0.05,
  lfcCutOff = 1,
  controlLfcCutOff = 1,
  plotType = "MA"
)
```

Arguments

diffstats	list; output of computeDiffStats function
contrast	character; contrast of interest to plot differential statistics results
title	character: title for the plot
controlGroup	character; control group such as IgG
selectedGenes	character: a vector defining genes to plot
fdrCutOff	numeric: False Discovery Rate (adj.P.Val) cut off
lfcCutOff	numeric: Log Fold Change (log2FC) cutoff for
controlLfcCutOff	numeric: only plot genes above controlLogFoldChange cutoff
plotType	character: which type of plot to generate: "MA" or "Volcano"

Details

Genes determined as significant according to the Log Fold Change and False Discovery Rate cutoffs are highlighted in red.

A user specified selection of genes can be highlighted by passing a character vector of Accessions to the `selectedGenes` argument. The contents of this vector will be matched with the Accessions column of the `diffstats` object to identify rows to highlight. These will be plotted in blue and labeled with the contents of the `GeneSymbol` column. Note that if the `GeneSymbol` for a selected gene is missing, no label will be apparent.

Value

An object created by `ggplot`

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
```

```
                                indExpData=c(7:16),
                                Sequences=2,
                                Accessions=6)
MSnset_norm <- groupScaling(MSnSet_data, scalingFunction=median)
MSnset_Pnorm <- summarizeIntensities(MSnset_norm, sum, human_anno)
contrasts <- c(tam.24h_vs_vehicle = "tam.24h - vehicle")
diffstats <- computeDiffStats(MSnset_Pnorm, contrasts=contrasts)
maVolPlot(diffstats, contrast = contrasts, plotType="MA")
maVolPlot(diffstats, contrast = contrasts, plotType="Volcano")
```

`mergePeptides` *Merge identical modified peptides intensities*

Description

Merge modified peptides with identical sequences to single peptide intensity. This function is especially useful for modified peptide enrichment based method such as phosphopeptide analysis.

Usage

```
mergePeptides(MSnSetObj, summarizationFunction, annotation, keepCols = NULL)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`summarizationFunction`	function; method used to aggregate the peptides. sum, mean or median
`annotation`	data.frame; a data.frame of protein annotation of four columns: "Accessions", "Gene", "Description" and "GeneSymbol"
`keepCols`	a vector of additional columns from fData(MSnSetObj) to keep. either be a numeric vector of column indices or a character vector of column names

Details

Rows of the intensity matrix with identical peptide sequences are merged by summarising the intensities using `summarizationFunction`.

Columns specified with `keepCols` are retained in the final output. Non-unique entries in different rows are concatenated with ';'.

Value

An object of class `MSnSet` (see `MSnSet-class`)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_P <- mergePeptides(MSnSet_data, sum, human_anno)
```

`mergeSites` *Merge identical modification sites intensities*

Description

Merge peptides with identical modification sites to single site intensity. This function is especially useful for data based on enrichment of specific peptide modification.

Usage

```
mergeSites(MSnSetObj, summarizationFunction, annotation, keepCols = NULL)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`summarizationFunction`	function; method used to aggregate the peptides. sum, mean or median
`annotation`	data.frame; a data.frame of protein annotation of four columns: "Accessions", "Gene", "Description" and "GeneSymbol"
`keepCols`	a vector of additional columns from fData(MSnSetObj) to keep. either be a numeric vector of column indices or a character vector of column names

Details

Rows of the intensity matrix with identical sites on same protein are merged by summarising the intensities using `summarizationFunction`. The merging will only take place if "Sites" and "Type" column are present in the fData(MSnSetObj). Sites contains the information of modified site position within the protein sequence and Type tells us about whether the modification is single (1xPhospho/Acetyl) or multi (2xPhospho/Acetyl).

Columns specified with `keepCols` are retained in the final output. Non-unique entries in different rows are concatenated with ';'.

Value

An object of class `MSnSet` (see `MSnSet-class`)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
#MSnset_P <- mergeSites(MSnSet_data, sum, human_anno)
```

---

mouse_anno *mouse_anno dataset*

Description

Uniprot Mouse protein annotation table.

Format

An object of class `data.frame` consisting of uniprot mouse protein annotation.

---

normalizeQuantiles *Quantile normalization*

Description

Performs quantile normalization on the intensities within columns

Usage

```
normalizeQuantiles(MSnSetObj)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet

Details

The peptide intensities are roughly replaced by the order statics on their abundance. This normalization technique has the effect of making the distributions of intensities from the different samples identical in terms of their statistical properties. It is the strongest normalization method and should be used carefully as it erases most of the difference between the samples.

Value

An object of class `MSnSet` (see `MSnSet-class`)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- normalizeQuantiles(MSnSet_data)
```

---

normalizeScaling *Normalization by scaling*

Description

Performs scaling normalization on the peptide/protein intensities (median or mean)

Usage

```
normalizeScaling(MSnSetObj, scalingFunction = median, ProteinId = NULL)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
scalingFunction	function; median or mean
ProteinId	character; protein Id

Details

In this normalization method the central tendencies (mean or median) of the samples are aligned. The central tendency for each sample is computed and log transformed. A scaling factor is determined by subtracting from each central tendency the mean of all the central tendencies. The raw intensities are then divided by the scaling factor to get normalized intensities.

The intensities can also be normalized based on the peptide intensities of a selected protein. For this the argument "ProteinId" allows you to define the protein that will be used for scaling the intensities.

Value

An object of class MSnSet (see MSnSet-class)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- normalizeScaling(MSnSet_data, scalingFunction=median)
```

`pcaPlot` *PCA plot*

Description

PCA plots of the samples within MSnset

Usage

```
pcaPlot(
  MSnSetObj,
  omitIgG = FALSE,
  sampleColours = NULL,
  transFunc = log2xplus1,
  transform = TRUE,
  colourBy = "SampleGroup",
  title = "",
  labelColumn = "BioRep",
  labelsize = 4,
  pointsize = 4,
  x.nudge = 4,
  x.PC = 1
)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`omitIgG`	Logical: whether to remove IgG from the PCA plot
`sampleColours`	character: A named vector of colours for samples
`transFunc`	func: internal helper function for log transformation
`transform`	logical: whether to log transform intensities
`colourBy`	character: column name to use for colouring samples from `pData(MSnSetObj)`
`title`	character: title for the plot
`labelColumn`	character: column name from `pData(MSnSetObj)` to use for labelling points on the plot

labelsize	numeric: size of the labels
pointsize	numeric: size of plotting points
x.nudge	numeric: distance to move labels along the x-axis away from the plotting points
x.PC	numeric: The principle component to plot on the x-axis; the following PC will be plotted on the y-axis

Details

The column provided to the "colourBy" argument will be used to colour the samples. The colours will be determined using the function assignColours, alternatively the user may specify a named vector of colours using the "sampleColours" argument. The names of the "sampleColours" vector should match the unique values in the "colourBy" column.

Value

An object created by ggplot

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
exprs(MSnSet_data) <- exprs(MSnSet_data)+0.01
pcaPlot(MSnSet_data, omitIgG = TRUE, labelColumn = "BioRep")

# custom colours and PC2 v PC3
customCols <- rainbow(length(unique(pData(MSnSet_data)$SampleGroup)))
names(customCols) <- unique(pData(MSnSet_data)$SampleGroup)
pcaPlot(MSnSet_data,
        omitIgG = TRUE,
        labelColumn = "BioRep",
        sampleColours = customCols,
        x.PC=2)
```

---

peptideIntensityPlot *Plot peptide intensities*

Description

Plots all the peptide intensities for the selected protein

Usage

```
peptideIntensityPlot(
  MSnSetObj,
  ProteinID,
  ProteinName,
  combinedIntensities = NULL,
  selectedSequence = NULL,
  selectedModifications = NULL
)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet containing peptide level intensities
`ProteinID`	character: Uniprot ID of the protein
`ProteinName`	character: name of the protein
`combinedIntensities`	MSnSet; an object of class MSnSet containing protein level intensities
`selectedSequence`	character: sequence present in the "Sequences" column in `fData(MSnSetObj)`
`selectedModifications`	character: modification present in the "Modifications" column in `fData(MSnSetObj)`

Details

Providing a summarised protein level MSnSet object to the `combinedIntensities` argument will add a summed protein intensity trace to the plot along with the peptide intensities.

Value

An object created by `ggplot`

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_P <- summarizeIntensities(MSnSet_data, sum, human_anno)
peptideIntensityPlot(MSnSet_data,
                     combinedIntensities=MSnset_P,
                     ProteinID="P03372",
                     ProteinName= "ESR1")
```

plotMeanVar *Mean variance plot*

Description

Computes and plots variance v mean intensity for peptides in MSnset

Usage

```
plotMeanVar(MSnSetObj, title = "")
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
title	character: title for the plot

Value

An object created by ggplot

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
plotMeanVar(MSnSet_data, title="Mean_Variance")
```

regressIntensity *Regression based analysis*

Description

Performs linear regression on protein intensities based on selected protein (qPLEX-RIME bait)

Usage

```
regressIntensity(MSnSetObj, ProteinId, controlInd = NULL, plot = TRUE)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
ProteinId	character; Uniprot protein ID
controlInd	numeric; index of IgG within MSnSet
plot	character; Whether or not to plot the QC histograms

Details

This function performs regression based analysis upon protein intensities based on a selected protein. In qPLEX RIME this method could be used to regress out the effect of target protein on other interactors. This function corrects this dependency of many proteins on the target protein levels by linear regression. It sets the target protein levels as the independent variable (x) and each of the other proteins as the dependent variable (y). The resulting residuals of the linear regressions y=ax+b are the protein levels corrected for target protein dependency.

Value

An object of class MSnSet (see MSnSet-class). This consists of corrected protein levels. In addition, the function can also plot histograms of correlation of target protein with all other proteins before and after this correction.

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_P <- summarizeIntensities(MSnSet_data, sum, human_anno)
IgG_ind <- which(pData(MSnset_P)$SampleGroup == "IgG")
MSnset_reg <- regressIntensity(MSnset_P,
                               controlInd=IgG_ind,
                               ProteinId="P03372")
```

---

rliPlot *Relative log intensity plot*

Description

Relative log intensity (RLI) plots of the samples within MSnset

Usage

```
rliPlot(
  MSnSetObj,
  title = "",
  sampleColours = NULL,
  colourBy = "SampleGroup",
  omitIgG = TRUE
)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet
title	character: title for the plot
sampleColours	character: a named vector of colours for samples
colourBy	character: column name to use for colouring samples from pData(MSnSetObj)
omitIgG	logical: whether to remove IgG from the RLI plot

Details

An RLI-plot is a boxplot that can be used to visualise unwanted variation in a data set. It is similar to the relative log expression plot developed for microarray analysis - see Gandolfo and Speed (2018). Rather than examining gene expression, the RLI plot uses the MS intensities for each peptide or the summarised protein intensities.

The column provided to the colourBy argument will be used to colour the samples. The colours will be determined using the function assignColours, alternatively the user may specify a named vector of colours using the sampleColours argument. The names of the sampleColours vector should match the unique values in the colourBy column.

Value

An object created by ggplot

References

Gandolfo LC, Speed TP (2018) RLE plots: Visualizing unwanted variation in high dimensional data. PLoS ONE 13(2): e0191629. https://doi.org/10.1371/journal.pone.0191629

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
rliPlot(MSnSet_data, title = "qPLEX_RIME_ER")
```

```
# custom colours
customCols <- rainbow(length(unique(pData(MSnSet_data)$SampleGroup)))
names(customCols) <- unique(pData(MSnSet_data)$SampleGroup)
rliPlot(MSnSet_data, title = "qPLEX_RIME_ER", sampleColours = customCols)
```

`rowScaling` *Normalization by scaling peptide/protein intensity across all samples*

Description

Divide each peptide/protein by the row mean/median and transform to log2

Usage

```
rowScaling(MSnSetObj, scalingFunction)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`scalingFunction`	function; median or mean

Value

An object of class `MSnSet` (see `MSnSet-class`).

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_norm <- rowScaling(MSnSet_data, scalingFunction=median)
```

`summarizeIntensities` *Summarizes peptides intensities to proteins*

Description

Summarizes multiple peptides intensities measurements to protein level.

Usage

```
summarizeIntensities(MSnSetObj, summarizationFunction, annotation)
```

Arguments

`MSnSetObj`	MSnSet; an object of class MSnSet
`summarizationFunction`	function; method used to aggregate the peptides into proteins. Sum, mean or median
`annotation`	data.frame; a data.frame of protein annotation of four columns: "Accessions", "Gene", "Description" and "GeneSymbol"

Value

An object of class `MSnSet` (see `MSnSet-class`)

Examples

```
data(human_anno)
data(exp3_OHT_ESR1)
MSnSet_data <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX1,
                               metadata=exp3_OHT_ESR1$metadata_qPLEX1,
                               indExpData=c(7:16),
                               Sequences=2,
                               Accessions=6)
MSnset_P <- summarizeIntensities(MSnSet_data, sum, human_anno)
```

Index