Package: qPLEXanalyzer (via r-universe)

November 25, 2026

Type Package

Title Tools for quantitative proteomics data analysis

Version 1.32.0

Date 2026-11-11

Description Tools for TMT based quantitative proteomics data analysis.

License GPL-2

Imports assertthat, BiocGenerics, Biostrings, dplyr (>= 1.0.0), ggdendro, ggplot2 (>= 3.5.0), graphics, grDevices, IRanges, limma, preprocessCore, purrr (>= 1.0.0), RColorBrewer, readr, rlang, scales, stats, stringr, tibble, tidyr, tidyselect, utils

Depends R (>= 4.1.0), Biobase, MSnbase

Suggests patchwork, knitr, qPLEXdata, rmarkdown, UniProt.ws

VignetteBuilder knitr

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

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

Encoding UTF-8

RoxygenNote 7.3.3

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-dev libx11-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:41:07 UTC

RemoteUrl https://github.com/bioc/qPLEXanalyzer

RemoteRef RELEASE_3_24

RemoteSha 559443ef7aec6da223fef39285740c5c7f181c68

Contents

qPLEXanalyzer-package . . . 2
assignColours . . . 4
coefVar . . . 4
computeDiffStats . . . 5
convertToMSnset . . . 7
corrPlot . . . 9
coveragePlot . . . 10
ER_ARID1A_KO_MCF7 . . . 11
exp2_Xlink . . . 12
exp3_OHT_ESR1 . . . 12
getContrastResults . . . 13
groupScaling . . . 14
hierarchicalPlot . . . 15
human_anno . . . 16
intensityBoxplot . . . 16
intensityPlot . . . 17
IRSnorm . . . 19
maVolPlot . . . 20
mergePeptides . . . 22
mergeSites . . . 23
mouse_anno . . . 24
normalizeQuantiles . . . 24
normalizeScaling . . . 25
pcaPlot . . . 26
peptideIntensityPlot . . . 28
plotMeanVar . . . 29
regressIntensity . . . 30
rliPlot . . . 31
rowScaling . . . 32
summarizeIntensities . . . 33

Index **35**

`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
- `mergePeptides`: Merges identical peptide sequences of the same protein into a single peptide intensity

- `mergeSites`: Merges identical modification sites of the same protein into a single site intensity
- `normalizeQuantiles`: Performs quantile normalization on the peptides/proteins intensities
- `normalizeScaling`: Performs scaling normalization on the peptides/proteins intensities (mean or median)
- `groupScaling`: Performs scaling normalization on the peptides/proteins intensities within group (median or mean)
- `rowScaling`: Normalization by scaling peptide/protein intensity across all samples
- `IRSnorm`: Batch correction of multiple TMT runs using an Internal Reference Scale sample measured in each run
- `regressIntensity`: Performs linear regression on protein intensities based on selected protein
- `coefVar`: Calculating the coefficient of variation by utilizing expression data within individual sample groups
- `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 selected protein in proteomics experiment
- `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 log intensity plot of all the samples

Author(s)

Matthew Eldridge, Kamal Kishore, Ashley Sawle (Maintainer)

<ads2202cu@gmail.com>

See Also

Useful links:

- Report bugs at https://github.com/crukci-bioinformatics/qPLEXanalyzer/issues

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 named character vector with one colour per unique value of the colourBy column (not one per sample). The names are the values of the column, sorted alphabetically as character strings. Colours are taken from the RColorBrewer "Dark2" palette, interpolated when there are more than 8 values.

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)
```

coefVar *Calculating the coefficient of variation by utilizing expression data within individual sample groups*

Description

Calculating the coefficient of variation by utilizing peptide/protein expression data within individual sample groups.

Usage

```
coefVar(MSnSetObj)
```

Arguments

MSnSetObj	MSnSet; an object of class MSnSet

Details

In this approach, we calculate distributions of the coefficient of variation (CV) for the dataset. The CVs are determined based on peptides or proteins intensities within each sample group, and the results are visualized through boxplots and cumulative fraction plots for each sample group.

The samples are grouped by the "SampleGroup" column of pData(MSnSetObj). For each feature and group the CV is computed as 100 * standard deviation / mean of the intensities as given (no transformation is applied, so raw intensities should be used). The cumulative fraction plot shows, for each group, the proportion of features with a CV less than or equal to the value on the x-axis. CVs that cannot be computed (e.g. features with missing values, or with a mean of zero) are excluded from the cumulative fraction plot, with a message giving their number.

Value

A `list` of two objects of class `ggplot`: `boxpl`, boxplots of the CVs of each sample group, and `cumulpl`, the cumulative fraction plot.

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)
# the IgG samples contain zero intensities, which give a mean of zero (and an
# undefined CV) for some peptides; a constant is added to avoid this
exprs(MSnSet_data) <- exprs(MSnSet_data)+1.1
res <- coefVar(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". All variables in the model, including "SampleGroup", are treated as categorical: numeric or logical columns (e.g. a run number stored as an integer) are converted to factors.
transform	logical; apply log2 transformation to the raw intensities before fitting. The transformation is `log2(x + 1e-10)`, so zero intensities become about -33; set to FALSE if the intensities are already on a log scale.
contrasts	character; named character vector of contrasts for differential statistics, written in terms of the SampleGroup names (e.g. "tam.24h - vehicle"). Group names containing spaces can be written as they are or with the spaces replaced by "_". A contrast written as "groupA_vs_groupB", where both are existing group names, is read as "groupA - groupB". Any other expression accepted by `makeContrasts` can be used, e.g. "(A - B) - (C - D)".
trend	logical; passed to `eBayes`: should an intensity-dependent trend be allowed for the prior variance? If FALSE the prior variance is constant.
robust	logical; passed to `eBayes`: should the estimation of the prior degrees of freedom and prior variance be robustified against outlier sample variances?

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 a variable for each sample group ("SampleGroup") plus any additional variables given in `batchEffect`. Then, log2 fold changes between comparisons 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 effects (e.g. MS run), additional variable(s) can be given in `batchEffect`. Each must correspond to a column name in pData(MSnSetObj). "SampleGroup" is always included in the model; the variables in `batchEffect` are added to it.

Repeated measurements of the same biological unit, such as technical replicates of a sample or several experiments performed on the same cell line, are accounted for through the "TechRep" column of pData(MSnSetObj). If this column contains any non-NA values, the correlation between measurements of the same unit is estimated with `duplicateCorrelation` using "TechRep" as the blocking variable, and the linear model is then fitted with `lmFit` using the same block and the consensus correlation. Samples sharing a "TechRep" value are treated as correlated measurements of one biological unit, so all measurements of the same unit must have the same value, and different units must have different values. A unit may be measured in more than one group (for example the same subject measured under two conditions, as in the "Multi-level experiments" section of the limma User's Guide); what must be avoided is reusing a replicate or cell line number for different

units. For example, with two cell lines per group, four experiments per cell line and each cell line belonging to one group only, label the samples by cell line (e.g. "Control_1", "Control_2", "POLG_1", "POLG_2") rather than by a cell line number that repeats across groups (e.g. 1, 2). If "TechRep" is used, every sample must have a value: a sample without repeated measurements should be given its own unique value. If all "TechRep" values are NA, the linear model is fitted without blocking. A variable nested within "SampleGroup", such as cell line in the example above, should not be given in `batchEffect`, because its effects cannot be estimated separately from the groups.

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. Required when type = "peptide"; must be NULL when type = "protein". The column is renamed "Sequences".
Accessions	numeric; a numeric value indicating the index of column consisting of protein accession in ExpObj. The column is renamed "Accessions". Neither this column nor the Sequences column may be one of the intensity columns given in indExpData.
type	character; what type of data set to create, either 'peptide' or 'protein'
rmMissing	logical; if TRUE, rows with a missing value in any of the intensity columns are removed. If FALSE, rows with missing values are kept, except rows in which all intensities are missing, which are always removed.

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. "TechRep" should be NA for all samples when there are no repeated measurements of the same biological unit (e.g. technical replicates, or several experiments on the same cell line). Otherwise, all samples from the same unit should share one value, different units must have different values (e.g. "CellLine_1", "CellLine_2" rather than a number reused for different cell lines), and each sample without repeated measurements should have its own unique value; it is used as the blocking variable in computeDiffStats. 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

Details

Pearson correlations between samples are computed on the intensities as given (no transformation is applied). The intensities must not contain missing values. Correlations outside low_cor_limit and high_cor_limit are coloured as the nearest limit, with a warning.

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.

The peptides are taken from the "Sequences" column of fData(MSnSetObj) for the rows whose Accessions match `ProteinID`. Flanking residues written as a single residue in square brackets (e.g. "[K].PEPTIDE.[R]") are removed before the peptides are matched exactly to the protein sequence; peptides that are not found in the sequence are ignored without a message. The "Number of Unique Peptides" shown in the subtitle is the number of distinct matched positions (start and end) in the sequence. The fasta file should contain a single protein sequence.

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; name of contrast of interest for which to retrieve differential statistics results
`controlGroup`	character; control group such as IgG
`transform`	logical; apply log2 transformation (`log2(x + 1e-10)`) to the sample intensity columns included in the output. This does not affect the statistics, which are computed by `computeDiffStats`.
`writeFile`	logical; whether to also write the results to a tab-separated text file in the working directory, named after the contrast (with " - " replaced by "_vs_") with the extension ".txt"

Details

The results for the contrast are obtained with `topTable` (with confidence intervals) and combined with the feature data and the sample intensities. Rows are sorted by decreasing B statistic.

If `controlGroup` is given, a `controlLogFoldChange` column is added. It is computed from the group means of the fitted linear model: for each group involved in the contrast the log2 fold change over the control group is calculated, and the maximum of these is reported.

Value

A `data.frame` with one row per feature, containing the feature data columns (a "Count" column is renamed "Unique_peptides"; for data summarised with `summarizeIntensities` it is the number of distinct peptide sequences), one column of intensities per sample, and the `topTable` results: `log2FC` (renamed from logFC), `CI.L`, `CI.R`, `AvgIntensity` (renamed from AveExpr), `t`, `P.Value`, `adj.P.Val` and `B`, plus `controlLogFoldChange` if `controlGroup` is given. log2FC, t, B and the sample intensities are rounded to 2 decimal places, and P.Value and adj.P.Val to 2 significant digits.

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="tam.24h_vs_vehicle")
```

---

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 transform the intensities with trFunc
xlab	character: label for x-axis. The default label "log2(intensity)" is not changed automatically when transform = FALSE or a different trFunc is used, so a suitable label should then be supplied.
trFunc	function: transformation applied to the intensities when transform = TRUE. The default is log2(x + 1e-10), which turns zero intensities into about -33. Missing values are removed before plotting.

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
# combine() keeps one copy of each annotation column per set (suffixed with
# the set name); keep the annotation columns of the first set
fData(MSnset_comb) <- fData(MSnset_comb)[, 1:4]
colnames(fData(MSnset_comb)) <- c("Accessions", "Gene", "Description",
                                  "GeneSymbol")
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; name of 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 of Accessions to highlight and label
`fdrCutOff`	numeric: False Discovery Rate (adj.P.Val) cut off for significance
`lfcCutOff`	numeric: cutoff on the absolute log2 fold change (log2FC) for significance
`controlLfcCutOff`	numeric: used only when `controlGroup` is given; cutoff on the absolute `controlLogFoldChange` for significance. It does not remove any proteins from the plot.
`plotType`	character: which type of plot to generate: "MA" or "Volcano"

Details

All proteins are plotted. Proteins with `adj.P.Val` <= `fdrCutOff` and `abs(log2FC)` >= `lfcCutOff` are highlighted as significant in red. If `controlGroup` is given, a protein must in addition have `abs(controlLogFoldChange)` >= `controlLfcCutOff` to be highlighted (see `getContrastResults`). The values are taken from the output of `getContrastResults`, i.e. after its rounding.

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 cyan 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 = "tam.24h_vs_vehicle", plotType="MA")
maVolPlot(diffstats, contrast = "tam.24h_vs_vehicle", 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`. fData(MSnSetObj) must contain "Sites" and "Type" columns, otherwise an error is raised. 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). None of the datasets provided with the package contains these columns, so the example below is not run.

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 and back-transforming (exponentiating) the difference, i.e. the scaling factor of a sample is its central tendency divided by the geometric mean of the central tendencies of all samples. The raw intensities are then divided by the scaling factor to get normalized intensities, so that all samples end up with the same central tendency. Missing values are ignored when computing the central tendencies. The intensities should be raw (not log transformed) and the central tendency of every sample must be greater than zero.

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. Samples are removed if their SampleGroup is "IgG" (in any letter case); other names for the control group are not recognised.
`sampleColours`	character: A named vector of colours for samples
`transFunc`	function: transformation applied to the intensities when `transform = TRUE`. The default is `log2(x + 1e-10)`, which turns zero intensities into about -33.
`transform`	logical: whether to transform the intensities with `transFunc`. Features with missing values are removed before the PCA, and the PCA is computed with `prcomp` on the centred, unscaled data.

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 or "none" to omit labels
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 principal 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, or NULL if no feature without missing values remains.

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

Details

The intensities are transformed with the natural logarithm after adding 0.001 (`log(x + 0.001)`), and the mean and variance of each feature are computed across all samples. A smoothing spline fitted to variance against mean is shown in red. The intensities must not contain missing values.

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; column indices of the control samples (e.g. IgG) to exclude from the regression and from the returned object
`plot`	logical; 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.

The returned intensities are these residuals: they are centred around zero for each protein, and the row of the target protein itself becomes zero in every sample. The samples given in `controlInd` are removed from the returned object, and the regression is performed on the intensities as given (no transformation is applied).

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. Samples are removed if their SampleGroup is "IgG" (in any letter case); other names for the control group are not recognised. Note that the default (TRUE) differs from that of `pcaPlot` (FALSE).

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. Intensities are log2 transformed and values that are not finite (zeros and missing values) are removed; the RLI of a feature in a sample is its log2 intensity minus the median log2 intensity of that feature across the samples.

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"

Details

The peptide intensities of each protein (grouped by the "Accessions" column of fData(MSnSetObj)) are combined in each sample with `summarizationFunction`. Missing values are passed to this function as they are: with `sum`, `mean` or `median` a protein gets a missing value in a sample if any of its peptides is missing in that sample. The resulting feature data contain the annotation columns and a "Count" column giving the number of distinct peptide sequences (values of the "Sequences" column) of each protein. Proteins whose Accession is not found in the annotation are kept, with missing annotation.

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

∗ **datasets**
ER_ARID1A_KO_MCF7, 11
exp2_Xlink, 12
exp3_OHT_ESR1, 12
human_anno, 16
mouse_anno, 24
∗ **data**
ER_ARID1A_KO_MCF7, 11
exp2_Xlink, 12
exp3_OHT_ESR1, 12
human_anno, 16
mouse_anno, 24
∗ **package**
qPLEXanalyzer-package, 2

assignColours, 4, *17, 18*, *27*, *31*

coefVar, 4
computeDiffStats, 5, *8*
convertToMSnset, 7
corrPlot, 9
coveragePlot, 10

data.frame, *13*, *16*, *24*
duplicateCorrelation, *6*

eBayes, *6, 7*
ER_ARID1A_KO_MCF7, 11
exp2_Xlink, 12
exp3_OHT_ESR1, 12

getContrastResults, 13, *21*
groupScaling, 14

hierarchicalPlot, 15
human_anno, 16

intensityBoxplot, 16
intensityPlot, 17
IRSnorm, 19

limma, *5*, *7*
list, *11, 12*
lmFit, *6*

makeContrasts, *6*
maVolPlot, 20
mergePeptides, 22
mergeSites, 23
mouse_anno, 24

normalizeQuantiles, 24
normalizeScaling, 25

pcaPlot, 26, *31*
peptideIntensityPlot, 28
plotMeanVar, 29
prcomp, *26*

qPLEXanalyzer *(*qPLEXanalyzer-package*)*, 2
qPLEXanalyzer-package, 2

regressIntensity, 30
rliPlot, 31
rowScaling, 32

summarizeIntensities, *13*, 33

topTable, *13*