Package ‘scp’

July 8, 2021

Title  Mass Spectrometry-Based Single-Cell Proteomics Data Analysis
Version  1.2.0
Description  Utility functions for manipulating, processing, and analyzing mass spectrometry-based single-cell proteomics (SCP) data. The package is an extension to the 'QFeatures' package designed for SCP applications.
Depends  R (>= 4.0), QFeatures
Imports  methods, stats, utils, SingleCellExperiment,
          SummarizedExperiment, MultiAssayExperiment, MsCoreUtils,
          matrixStats, S4Vectors, dplyr, magrittr, rlang
Suggests  testthat, knitr, BiocStyle, BiocCheck, rmarkdown, patchwork,
          ggplot2, impute, scater, sva, preprocessCore, vsn, uwot
License  Artistic-2.0
Encoding  UTF-8
VignetteBuilder  knitr
biocViews  GeneExpression, Proteomics, SingleCell, MassSpectrometry,
            Preprocessing, CellBasedAssays
BugReports  https://github.com/UCLouvain-CBIO/scp/issues
URL  https://UCLouvain-CBIO.github.io/scp
Roxygen  list(markdown=TRUE)
RoxygenNote  7.1.1

git_url  https://git.bioconductor.org/packages/scp
git_branch  RELEASE_3_13
git_last_commit  a7c883a
git_last_commit_date  2021-05-19
Date/Publication  2021-07-08
Author  Christophe Vanderaa [aut, cre]
        (<https://orcid.org/0000-0001-7443-5427>),
        Laurent Gatto [aut] (<https://orcid.org/0000-0002-1520-2268>)
Maintainer  Christophe Vanderaa <christophe.vanderaa@uclouvain.be>
aggregateFeaturesOverAssays

**R topics documented:**

- aggregateFeaturesOverAssays ........................................... 2
- computeMedianCV_SCoPE2 ............................................. 3
- computeSCR .............................................................. 4
- divideByReference ....................................................... 5
- medianCVperCell ......................................................... 6
- mqScpData ................................................................. 7
- normalizeSCP ............................................................. 12
- pep2qvalue ................................................................. 13
- readSCP ..................................................................... 14
- readSingleCellExperiment ............................................. 16
- rowDataToDF ............................................................... 17
- sampleAnnotation ......................................................... 18
- scp1 ........................................................................ 19

Index 20

---

**aggregateFeaturesOverAssays**

*Aggregate features over multiple assays*

**Description**

This function is a wrapper function around `QFeatures::aggregateFeatures`. It allows the user to provide multiple assays for which `aggregateFeatures` will be applied sequentially.

**Usage**

```r
aggregateFeaturesOverAssays(object, i, fcol, name, fun, ...)
```

**Arguments**

- `object` A `QFeatures` object
- `i` A numeric(1) or character(1) indicating which assay to transfer the `colData` to.
- `fcol` The feature variables for each assays `i` defining how to summarise the `QFeatures`. If `fcol` has length 1, the variable name is assumed to be the same for all assays
- `name` A character() naming the new assay. `name` must have the same length as `i`. Note that the function will fail if of the names in `name` is already present.
- `fun` A function used for quantitative feature aggregation.
- `...` Additional parameters passed the `fun`.

**Value**

A `QFeatures` object
computeMedianCV_SCoPE2

See Also

QFeatures::aggregateFeatures

Examples

```r
data("scp1")
scp1 <- aggregateFeaturesOverAssays(scp1, 
i = 1:3, 
fcol = "peptide", 
name = paste0("peptides", 1:3), 
fun = colMeans, 
na.rm = TRUE)
scp1
```

duplicate

computeMedianCV_SCoPE2

(Deprecated) Compute the median coefficient of variation (CV) per cell

Description

This function is deprecated and should no longer be used. To reproduce the SCoPE2 script, you can now use medianCVperCell with the following arguments:

Usage

```r
computeMedianCV_SCoPE2(object, i, peptideCol, proteinCol, batchCol)
```

Arguments

- `object`: NULL
- `i`: NULL
- `peptideCol`: NULL
- `proteinCol`: NULL
- `batchCol`: NULL

Details

- `norm = "SCoPE2"
- `nobs = 6`

Make sure to provide the peptide data from separate assays so that the normalization factors are computed per batch.
computeSCR  

*Compute the sample over carrier ratio (SCR)*

**Description**

The function computes the ratio of the intensities of sample channels over the intensity of the carrier channel for each feature. The ratios are averaged within the assay.

**Usage**

```r
computeSCR(
  object,
  i,
  colDataCol,
  samplePattern,
  carrierPattern,
  rowDataName = "MeanSCR"
)
```

**Arguments**

- **object**: A `QFeatures` object.
- **i**: A character() or integer() indicating for which assay(s) the SCR needs to be computed.
- **colDataCol**: A character(1) indicating the variable to take from `colData(object)` that gives the sample annotation.
- **samplePattern**: A character(1) pattern that matches the sample encoding in `colDataCol`.
- **carrierPattern**: A character(1) pattern that matches the carrier encoding in `colDataCol`. Only one match per assay is allowed, otherwise only the first match is taken.
- **rowDataName**: A character(1) giving the name of the new variable in the rowData where the computed SCR will be stored. The name cannot already exist in any of the assay rowData.

**Value**

A `QFeatures` object for which the rowData of the given assay(s) is augmented with the mean SCR.

**Examples**

```r
data("scp1")
scp1 <- computeSCR(scp1,
  i = 1,
  colDataCol = "SampleType",
  carrierPattern = "Carrier",
  samplePattern = "Blank|Macrophage|Monocyte",
  rowDataName = "MeanSCR")
```
## Check results
rowDataToDF(scp1, 1, "MeanSCR")

---

**divideByReference**

**Divide assay columns by a reference column**

### Description

The function divides the sample columns by a reference column. The sample and reference columns are defined based on the provided `colDataCol` variable and on regular expression matching.

### Usage

```
divideByReference(object, i, colDataCol, samplePattern = ".", refPattern)
```

### Arguments

- **object**: A `QFeatures` object
- **i**: A numeric() or character() vector indicating from which assays the rowData should be taken.
- **colDataCol**: A character(1) indicating the variable to take from `colData(object)` that gives the sample annotation.
- **samplePattern**: A character(1) pattern that matches the sample encoding in `colDataCol`. By default all samples are divided (using the regex wildcard ".").
- **refPattern**: A character(1) pattern that matches the carrier encoding in `colDataCol`. Only one match per assay is allowed, otherwise only the first match is taken.

### Details

The supplied assay(s) are replaced with the values computed after reference division.

### Value

A `QFeatures` object

### Examples

```
data("scp1")
scp1 <- divideByReference(scp1,
    i = 1,
    colDataCol = "SampleType",
    samplePattern = "Macrophage",
    refPattern = "Ref")
```
medianCVperCell  
\textit{Compute the median coefficient of variation (CV) per cell}

\textbf{Description}

The function computes for each cell the median CV and stores them accordingly in the \texttt{colData} of the \texttt{QFeatures} object. The CVs in each cell are computed from a group of features. The grouping is defined by a variable in the \texttt{rowData}. The function can be applied to one or more assays, as long as the samples (column names) are not duplicated. Also, the user can supply a minimal number of observations required to compute a CV to avoid that CVs computed on too few observations influence the distribution within a cell. The quantification matrix can be optionally normalized before computing the CVs. Multiple normalizations are possible.

\textbf{Usage}

\begin{verbatim}
medianCVperCell(
  object, 
  i, 
  groupBy, 
  nobs = 5, 
  na.rm = TRUE, 
  colDataName = "MedianCV", 
  norm = "none", 
  ...
)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{object} \hspace{1cm} A \texttt{QFeatures} object
  \item \texttt{i} \hspace{1cm} A \texttt{numeric()} or \texttt{character()} vector indicating from which assays the \texttt{rowData} should be taken.
  \item \texttt{groupBy} \hspace{1cm} A \texttt{character(1)} indicating the variable name in the \texttt{rowData} that contains the feature grouping.
  \item \texttt{nobs} \hspace{1cm} An \texttt{integer(1)} indicating how many observations (features) should at least be considered for computing the CV. Since no CV can be computed for less than 2 observations, \texttt{nobs} should at least be 2.
  \item \texttt{na.rm} \hspace{1cm} A \texttt{logical(1)} indicating whether missing data should be removed before computation.
  \item \texttt{colDataName} \hspace{1cm} A \texttt{character(1)} giving the name of the new variable in the \texttt{colData} where the computed CVs will be stored. The name cannot already exist in the \texttt{colData}.
  \item \texttt{norm} \hspace{1cm} A \texttt{character()} of normalization methods that will be sequentially applied. Available methods and additional information about normalization can be found in \texttt{MsCoreUtils::normalizeMethods}. You can also specify \texttt{norm = "SCoPE2"} to reproduce the normalization performed before computing the CVs as suggested by Specht et al. \texttt{norm = "none"} will not normalize the data (default)
  \item \texttt{...} \hspace{1cm} Additional arguments that are passed to the normalization method.
\end{itemize}
**Details**

A new column is added to the `colData` of the object. The samples (columns) that are not present in the selection `i` will get assigned an NA.

**Value**

A `QFeatures` object.

**References**


**Examples**

data("scp1")
scp1 <- filterFeatures(scp1, ~ !is.na(Proteins))
scp1 <- medianCVperCell(scp1,
    i = 1:3,
    groupBy = "Proteins",
    nobs = 5,
    na.rm = TRUE,
    colDataName = "MedianCV",
    norm = "div.median")

## Check results
hist(scp1$MedianCV)

---

**mqScpData**

**Example MaxQuant/SCoPE2 output**

**Description**

A `data.frame` with 1088 observations and 139 variables, as produced by reading a MaxQuant output file with `read_delim()`.

- Sequence: a character vector
- Length: a numeric vector
- Modifications: a character vector
- Modified.sequence: a character vector
- Deamidation..N..Probabilities: a character vector
- Oxidation..M..Probabilities: a character vector
- Deamidation..N..Score.Diffs: a character vector
- Oxidation..M..Score.Diffs: a character vector
mqScpData

- Acetyl..Protein.N.term.: a numeric vector
- Deamidation..N.: a numeric vector
- Oxidation..M.: a numeric vector
- Missed.cleavages: a numeric vector
- Proteins: a character vector
- Leading.proteins: a character vector
- protein: a character vector
- Gene.names: a character vector
- Protein.names: a character vector
- Type: a character vector
- Set: a character vector
- MS.MS.m.z: a numeric vector
- Charge: a numeric vector
- m.z: a numeric vector
- Mass: a numeric vector
- Resolution: a numeric vector
- Uncalibrated...Calibrated.m.z..ppm.: a numeric vector
- Uncalibrated...Calibrated.m.z..Da.: a numeric vector
- Mass.error..ppm.: a numeric vector
- Mass.error..Da.: a numeric vector
- Uncalibrated.mass.error..ppm.: a numeric vector
- Uncalibrated.mass.error..Da.: a numeric vector
- Max.intensity.m.z.0: a numeric vector
- Retention.time: a numeric vector
- Retention.length: a numeric vector
- Calibrated.retention.time: a numeric vector
- Calibrated.retention.time.start: a numeric vector
- Calibrated.retention.time.finish: a numeric vector
- Retention.time.calibration: a numeric vector
- Match.time.difference: a logical vector
- Match.m.z.difference: a logical vector
- Match.q.value: a logical vector
- Match.score: a logical vector
- Number.of.data.points: a numeric vector
- Number.of.scans: a numeric vector
- Number.of.isotopic.peaks: a numeric vector
- PIF: a numeric vector
mqScepData

- Fraction.of.total.spectrum: a numeric vector
- Base.peak.fraction: a numeric vector
- PEP: a numeric vector
- MS.MS.count: a numeric vector
- MS.MS.scan.number: a numeric vector
- Score: a numeric vector
- Delta.score: a numeric vector
- Combinatorics: a numeric vector
- Intensity: a numeric vector
- Reporter.intensity.corrected.0: a numeric vector
- Reporter.intensity.corrected.1: a numeric vector
- Reporter.intensity.corrected.2: a numeric vector
- Reporter.intensity.corrected.3: a numeric vector
- Reporter.intensity.corrected.4: a numeric vector
- Reporter.intensity.corrected.5: a numeric vector
- Reporter.intensity.corrected.6: a numeric vector
- Reporter.intensity.corrected.7: a numeric vector
- Reporter.intensity.corrected.8: a numeric vector
- Reporter.intensity.corrected.9: a numeric vector
- Reporter.intensity.corrected.10: a numeric vector
- RI1: a numeric vector
- RI2: a numeric vector
- RI3: a numeric vector
- RI4: a numeric vector
- RI5: a numeric vector
- RI6: a numeric vector
- RI7: a numeric vector
- RI8: a numeric vector
- RI9: a numeric vector
- RI10: a numeric vector
- RI11: a numeric vector
- Reporter.intensity.count.0: a numeric vector
- Reporter.intensity.count.1: a numeric vector
- Reporter.intensity.count.2: a numeric vector
- Reporter.intensity.count.3: a numeric vector
- Reporter.intensity.count.4: a numeric vector
- Reporter.intensity.count.5: a numeric vector
• Reporter.intensity.count.6: a numeric vector
• Reporter.intensity.count.7: a numeric vector
• Reporter.intensity.count.8: a numeric vector
• Reporter.intensity.count.9: a numeric vector
• Reporter.intensity.count.10: a numeric vector
• Reporter.PIF: a logical vector
• Reporter.fraction: a logical vector
• Reverse: a character vector
• Potential.contaminant: a logical vector
• id: a numeric vector
• Protein.group.IDs: a character vector
• Peptide.ID: a numeric vector
• Mod..peptide.ID: a numeric vector
• MS.MS.IDs: a character vector
• Best.MS.MS: a numeric vector
• AIF.MS.MS.IDs: a logical vector
• Oxidation..M..site.IDs: a logical vector
• remove: a logical vector
• dart_PEP: a numeric vector
• dart_qval: a numeric vector
• razor_protein_fdr: a numeric vector
• Deamidation..N..site.IDs: a numeric vector
• Deamidation..N..Score.Diffs: a logical vector
• Deamidation..N.: a logical vector
• Reporter.intensity.corrected.11: a logical vector
• Reporter.intensity.corrected.12: a logical vector
• Reporter.intensity.corrected.13: a logical vector
• Reporter.intensity.corrected.14: a logical vector
• Reporter.intensity.corrected.15: a logical vector
• Reporter.intensity.corrected.16: a logical vector
• RI12: a logical vector
• RI13: a logical vector
• RI14: a logical vector
• RI15: a logical vector
• RI16: a logical vector
• Reporter.intensity.count.11: a logical vector
mqScpData

- Reporter.intensity.count.12: a logical vector
- Reporter.intensity.count.13: a logical vector
- Reporter.intensity.count.14: a logical vector
- Reporter.intensity.count.15: a logical vector
- Reporter.intensity.count.16: a logical vector
- Deamidation..NQ..site.IDs: a logical vector
- input_id: a logical vector
- rt_minus: a logical vector
- rt_plus: a logical vector
- mu: a logical vector
- muij: a logical vector
- sigmaij: a logical vector
- pep_new: a logical vector
- exp_id: a logical vector
- peptide_id: a logical vector
- stan_peptide_id: a logical vector
- exclude: a logical vector
- residual: a logical vector
- participated: a logical vector
- peptide: a character vector

Usage

data("mqScpData")

Format

An object of class data.frame with 1361 rows and 149 columns.

Details

The dataset is a subset of the SCoPE2 dataset (version 2, Specht et al. 2019, BioRXiv). The input file evidence_unfiltered.csv was downloaded from a Google Drive repository. The MaxQuant evidence file was loaded and the data was cleaned (renaming columns, removing duplicate fields,...). MS runs that were selected in the scp1 dataset (see ?scp1) were kept along with a blank run. The data is stored as a data.frame.

See Also

readSCP() for an example on how mqScpData is parsed into a QFeatures object.
normalizeSCP

**Normalize single-cell proteomics (SCP) data**

**Description**

This function normalises an assay in a QFeatures according to the supplied method (see Details). The normalized data is added as a new assay.

**Usage**

```r
normalizeSCP(object, i, name = "normAssay", method, ...)
```

**Arguments**

- `object`: An object of class QFeatures.
- `i`: A numeric vector or a character vector giving the index or the name, respectively, of the assay(s) to be processed.
- `name`: A character(1) naming the new assay name. Defaults is are normAssay.
- `method`: character(1) defining the normalisation method to apply. See Details.
- `...`: Additional parameters passed to MsCoreUtils::normalizeMethods().

**Details**

The `method` parameter in `normalize` can be one of "sum", "max", "center.mean", "center.median", "div.mean", "div.median", "diff.median", "quantiles", "quantiles.robust" or "vsn". The MsCoreUtils::normalizeMethods() function returns a vector of available normalisation methods.

- For "sum" and "max", each feature's intensity is divided by the maximum or the sum of the feature respectively. These two methods are applied along the features (rows).
- "center.mean" and "center.median" center the respective sample (column) intensities by subtracting the respective column means or medians. "div.mean" and "div.median" divide by the column means or medians. These are equivalent to sweeping the column means (medians) along MARGIN = 2 with FUN = "-" (for "center.*") or FUN = "/" (for "div.*").
- "diff.median" centers all samples (columns) so that they all match the grand median by subtracting the respective columns medians differences to the grand median.
- Using "quantiles" or "quantiles.robust" applies (robust) quantile normalisation, as implemented in preprocessCore::normalize.quantiles() and preprocessCore::normalize.quantiles.robust() for "vsn" uses the vsn::vsn2() function. Note that the latter also glog-transforms the intensities. See respective manuals for more details and function arguments.

For further details and examples about normalisation, see MsCoreUtils::normalize_matrix().

**Value**

A QFeatures object with an additional assay containing the normalized data.
See Also

QFeatures::normalize for more details about normalize
Value
A QFeatures object.

References


Examples
data("scp1")
scp1 <- pep2qvalue(scp1,
  i = 1,
  groupBy = "protein",
  PEP = "dart_PEP",
  rowDataName = "qvalue_protein")
## Check results
rowDataToDF(scp1, 1, c("dart_PEP", "qvalue_protein"))

readSCP
Read single-cell proteomics data as a QFeatures object from tabular data and metadata

Description
Convert tabular quantitative MS data and metadata from a spreadsheet or a data.frame into a QFeatures object containing SingleCellExperiment objects.

Usage
readSCP(
  featureData,
  colData,
  batchCol,
  channelCol,
  suffix = NULL,
  removeEmptyCols = FALSE,
  verbose = TRUE,
  ...)

Arguments

featureData  File or object holding the quantitative data. Can be either a character(1) with the path to a text-based spreadsheet (comma-separated values by default, but see ...) or an object that can be coerced to a data.frame. It is advised not to encode characters as factors.

colData  A data.frame or any object that can be coerced to a data.frame. colData is expected to contain all the sample meta information. Required fields are the acquisition batch (given by batchCol) and the acquisition channel within the batch (e.g. TMT channel, given by channelCol). Additional fields (e.g. sample type, acquisition date,...) are allowed and will be stored as sample meta data.

batchCol  A numeric(1) or character(1) pointing to the column of featureData and colData that contain the batch names. Make sure that the column name in both table are either identical (if you supply a character) or have the same index (if you supply a numeric).

channelCol  A numeric(1) or character(1) pointing to the column of colData that contains the column names of the quantitative data in featureData (see Example).

suffix  A character() giving the suffix of the column names in each assay. The length of the vector must equal the number of quantification channels and must contain unique character elements. If NULL, the names of the quantification columns in featureData are taken as suffix.

removeEmptyCols  A logical(1). If true, the function will remove in each batch the columns that contain only missing values.

verbose  A logical(1) indicating whether the progress of the data reading and formatting should be printed to the console. Default is TRUE.

...  Further arguments that can be passed on to read.csv except stringsAsFactors, which is always FALSE.

Value

An instance of class QFeatures. The expression data of each batch is stored in a separate assay as a SingleCellExperiment object.

Note

The SingleCellExperiment class is built on top of the RangedSummarizedExperiment class. This means that some column names are forbidden in the rowData. Avoid using the following names: seqnames, ranges, strand, start, end, width, element

Author(s)

Laurent Gatto, Christophe Vanderaa

Examples

```r
## Load an example table containing MaxQuant output
data("mqScpData")
```
## Load the (user-generated) annotation table
data("sampleAnnotation")

## Format the tables into a QFeatures object
readSCP(featureData = mqScpData,
        colData = sampleAnnotation,
        batchCol = "Raw.file",
        channelCol = "Channel")

---

**readSingleCellExperiment**

*Read SingleCellExperiment from tabular data*

**Description**

Convert tabular data from a spreadsheet or a data.frame into a SingleCellExperiment object.

**Usage**

```r
readSingleCellExperiment(table, ecol, fnames, ...)
```

**Arguments**

- `table`: File or object holding the quantitative data. Can be either a character(1) with the path to a text-based spreadsheet (comma-separated values by default, but see ...) or an object that can be coerced to a data.frame. It is advised not to encode characters as factors.
- `ecol`: A numeric indicating the indices of the columns to be used as assay values. Can also be a character indicating the names of the columns. Caution must be taken if the column names are composed of special characters like ( or - that will be converted to a . by the `read.csv` function. If `ecol` does not match, the error message will display the column names as seen by the `read.csv` function.
- `fnames`: An optional character(1) or numeric(1) indicating the column to be used as row names.
- `...`: Further arguments that can be passed on to `read.csv` except `stringsAsFactors`, which is always FALSE.

**Value**

An instance of class `SingleCellExperiment`.

**Note**

The `SingleCellExperiment` class is built on top of the `RangedSummarizedExperiment` class. This means that some column names are forbidden in the rowData. Avoid using the following names: `seqnames`, `ranges`, `strand`, `start`, `end`, `width`, `element`
rowDataToDF

Author(s)

Laurent Gatto, Christophe Vanderaa

See Also

The code relies on QFeatures::readSummarizedExperiment.

Examples

```r
## Load a data.frame with PSM-level data
data("mqScpData")

## Create the QFeatures object
sce <- readSingleCellExperiment(mqScpData,
                               grep("RI", colnames(mqScpData)))
```

rowDataToDF

Extract the rowData of a QFeatures object to a DataFrame

Description

The methods takes the rowData of one or more given assay in a QFeatures object and combines the data in a single DataFrame.

Usage

rowDataToDF(object, i, vars)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A QFeatures object</td>
</tr>
<tr>
<td>i</td>
<td>A numeric() or character() vector indicating from which assays the rowData should be taken.</td>
</tr>
<tr>
<td>vars</td>
<td>A character() vector indicating which variables from the rowData should be extracted.</td>
</tr>
</tbody>
</table>

Details

Along with the required rowData an additional .assay variable is created and holds the assay name from which the metadata was taken.

Value

A DataFrame object with the rowData row-binded over the required assays.
Examples

```r
## Extract the peptide length and sequence from the first 3 assays
data("scp1")
rowDataToDF(scp1, i = 1:3, c("Length", "Sequence"))
```

---

**sampleAnnotation**

*Single cell sample annotation*

**Description**

A data frame with 48 observations on the following 6 variables.

- Set: a character vector
- Channel: a character vector
- SampleType: a character vector
- lcbatch: a character vector
- sortday: a character vector
- digest: a character vector

**Usage**

```r
data("sampleAnnotation")
```

**Format**

An object of class `data.frame` with 64 rows and 6 columns.

**Details**

```r
## The dataset is a subset of the SCoPE2 dataset (version 2, Specht et al. 2019, BioRXiv). The input files batch.csv and annotation.csv were downloaded from a Google Drive repository. The two files were loaded and the columns names were adapted for consistency with `mqScpData` table (see ?mqScpData). The two tables were filtered to contain only sets present in “mqScpData. The tables were then merged based on the run ID, hence merging the sample annotation and the batch annotation. Finally, annotation for the blank run was added manually. The data is stored as a `data.frame`.
```

**See Also**

`readSCP()` to see how this file is used.
Description

A small QFeatures object with SCoPE2 data. The object is composed of 5 assays, including 3 PSM-level assays, 1 peptide assay and 1 protein assay.

Usage

data("scp1")

Format

An object of class QFeatures of length 5.

Details

The dataset is a subset of the SCoPE2 dataset (version 2, Specht et al. 2019, BioRXiv). This dataset was converted to a QFeatures object where each assay is stored as a SingleCellExperiment object. One assay per chromatographic batch ("LCA9", "LCA10", "LCB3") was randomly sampled. For each assay, 100 proteins were randomly sampled. PSMs were then aggregated to peptides and joined in a single assay. Then peptides were aggregated to proteins.

Examples

data("scp1")
scp1
Index

* datasets
  * mqScpData, 7
  * sampleAnnotation, 18
  * scp1, 19
aggregateFeaturesOverAssays, 2
computeMedianCV_SCoPE2, 3
computeSCR, 4
divideByReference, 5
medianCVperCell, 6
mqScpData, 7
MsCoreUtils::normalize_matrix(), 12
MsCoreUtils::normalizeMethods, 6
MsCoreUtils::normalizeMethods(), 12
normalizeSCP, 12
pep2qvalue, 13
preprocessCore::normalize.quantiles(), 12
preprocessCore::normalize.quantiles.robust(), 12
QFeatures, 11, 14, 15, 19
QFeatures::aggregateFeatures, 2, 3
QFeatures::normalize, 13
QFeatures::readSummarizedExperiment, 17
read.csv, 15, 16
read.delim(), 7
readSCP, 14
readSCP(), 11, 18
readSingleCellExperiment, 16
rowDataToDF, 17
sampleAnnotation, 18
scp1, 19
SingleCellExperiment, 14–16, 19
vsn::vsn2(), 12