Package 'pulsedSilac'

April 12, 2022

Type Package

Title Analysis of pulsed-SILAC quantitative proteomics data

Version 1.8.0

Author Marc Pagès-Gallego, Tobias B. Dansen

Maintainer Marc Pagès-Gallego <M.PagesGallego@umcutrecht.nl>

Description This package provides several tools for pulsed-SILAC data analysis. Functions are provided to organize the data, calculate isotope ratios, isotope fractions, model protein turnover, compare turnover models, estimate cell growth and estimate isotope recycling. Several visualization tools are also included to do basic data exploration, quality control, condition comparison, individual model inspection and model comparison.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 3.6.0)

biocViews Proteomics

Suggests testthat (>= 2.1.0), knitr, rmarkdown, gridExtra

Imports robustbase, methods, R.utils, taRifx, S4Vectors, SummarizedExperiment, ggplot2, ggridges, stats, utils, UpSetR, cowplot, grid, MuMIn

RoxygenNote 7.0.2

VignetteBuilder knitr

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

git_branch RELEASE_3_14

git_last_commit 69698be

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`addMisscleavedPeptides` *Add miss-cleaved peptide expression levels for old isotope recycling estimation*

Description

To estimate how much of the "old" isotope is being used in "new" proteins the expression level of miss-cleaved peptides that contain a mix of isotopes (one old and one new) and miss-cleaved peptides that contain only new isotopes can be used to estimate amino acid recycling. To add this information a `data.frame` with the following information is required:

- A column with ids that can be mapped to the peptide rowData (not necessary for SilacProteinExperiment objects).
- A column indicating which isotope configuration the peptide has: two heavy isotopes, mix of light and heavy isotope, etc.
- A set of columns with peptide expression quantification. The number of columns should be the same, and in the same order, as the number of samples in the SilacProteinExperiment, SilacPeptideExperiment or SilacProteomicsExperiment.

Non-detected measurements should be NA.

Usage

```
addMisscleavedPeptides(x, ...)

## S4 method for signature 'SilacProteinExperiment'
addMisscleavedPeptides(x, newdata, modCol, dataCols, idColPept)

## S4 method for signature 'SilacPeptideExperiment'
addMisscleavedPeptides(x, newdata, modCol, dataCols, idColPept)

## S4 method for signature 'SilacProteomicsExperiment'
addMisscleavedPeptides(x, newdata, modCol, dataCols, idColPept)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or SilacProteomicsExperiment.
...	Unused.
newdata	a data.frame containing the data described in the description.
modCol	character or numeric indicating which column of newdata contains the peptide's isotope configuration.
dataCols	character or numeric vector indicating which columns of newdata contain the quantitative data. It should have the same length as the number of samples in x (ncol(x)). Samples should be in the same order.
idColPept	character indicating which column contains the ids that will be used in the merge with rowDataPept(x). The column in newdata should have the same name as in rowDataPept(x).

Value

A SilacPeptideExperiment or SilacProteomicsExperiment with new assay entries. If x is a SilacProteinExperiment then a SilacPeptideExperiment is returned.

Examples

```
data('wormsPE')
data('recycleLightLysine')
protPE <- ProtExp(wormsPE)
missPE <- addMisscleavedPeptides(x = protPE,
```

```
                                newdata = recycleLightLysine,
                                idColPept = 'Sequence',
                                modCol = 'Modifications',
                                dataCols = c(18:31))

names(assays(missPE))[1:2] <- c('int_lys8lys8', 'int_lys8lys0')
missPE <- calculateOldIsotopePool(x = missPE, 'int_lys8lys8', 'int_lys8lys0')

plotDistributionAssay(missPE, assayName = 'oldIsotopePool')
```

---

barplotCounts *Number of detected features per sample*

Description

How many proteins/peptides are detected in each sample. NA are considered missing values.

Usage

```
barplotCounts(x, ...)

## S4 method for signature 'SilacProteinExperiment'
barplotCounts(x, assayName, returnDataFrame = FALSE, conditionCol)

## S4 method for signature 'SilacPeptideExperiment'
barplotCounts(x, assayName, returnDataFrame = FALSE, conditionCol)

## S4 method for signature 'SilacProteomicsExperiment'
barplotCounts(x, assayName, returnDataFrame = FALSE, conditionCol)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
assayName	Name of the assay to use in the plot.
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.

Value

A ggplot2 barplot object or a data.frame.

Examples

```
data('wormsPE')
barplotCounts(wormsPE, assayName = 'ratio')
```

---

barplotTimeCoverage *Number of detected features per sample*

Description

How many proteins/peptides are detected in each sample. Anything else than NA is considered detected.

Usage

```
barplotTimeCoverage(x, ...)

## S4 method for signature 'SilacProteinExperiment'
barplotTimeCoverage(x, assayName, returnDataFrame = FALSE, conditionCol)

## S4 method for signature 'SilacPeptideExperiment'
barplotTimeCoverage(x, assayName, returnDataFrame = FALSE, conditionCol)

## S4 method for signature 'SilacProteomicsExperiment'
barplotTimeCoverage(x, assayName, returnDataFrame = FALSE, conditionCol)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
assayName	Name of the assay to use in the plot.
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.

Value

A barplot or a data.frame.

Examples

```
data('wormsPE')
barplotTimeCoverage(wormsPE, assayName = 'ratio')
```

`buildLinkerDf` *Constructs a linkerDf that can be used as input when constructing a SilacProteomicsExperiment*

Description

Constructs a 4 column `data.frame` that contains the relationships between proteins and peptides: which peptides belong to which proteins and vice versa.

Usage

```
buildLinkerDf(protIDs, pepIDs, protToPep, pepToProt)
```

Arguments

`protIDs`	a `character` vector with unique ids that can be mapped back to the proteins. Must be in the same order as in the rowDataProt data frame.
`pepIDs`	a `character` vector with unique ids that can be mapped back to the peptides. Must be in the same order as in the rowDataPep data frame.
`protToPep`	a list with the same length as the number of protIDs. Every entry of the list contains the peptide ids that are linked to that protein. Items in the list must be in the same order as in protIDs.
`pepToProt`	a list with the same length as the number of pepIDs Every entry of the list contains the protein ids that are linked to that peptide. Items in the list must be in the same order as in pepIDs

Details

This data frame is used in several functions and operations involving the `SilacProteomicsExperiment` class. Especially in object merging and subsetting. The arguments protIDs and pepIDs are mandatory, but only one of the protToPep or pepToProt arguments is necessary to build the linkerDf.

Value

A `data.frame` with the following 4 columns:

protID Column with the protein IDs.

pepID Column with the peptide IDs.

protRow Column with row numbers of protein IDs.

protID Column with the row numbers of peptide IDs.

Examples

```
## list with the relationships
protein_to_peptide <- list(A = c('a', 'b'), B = c('c'), C = c('d', 'e'))
## function to build the data.frame
linkerDf <- buildLinkerDf(protIDs = LETTERS[1:3],
                          pepIDs  = letters[1:5],
                          protToPep = protein_to_peptide)
linkerDf
```

`calculateAIC` *Calculates the Akaike Information Criteria (AIC)*

Description

Calculates the AIC for each of the computed models. Requires that `modelTurnover` is run with `reuturnModel = TRUE`.

Usage

```
calculateAIC(modelList, smallSampleSize = TRUE)
```

Arguments

`modelList`	a `list` with the model metrics, the output from `modelTurnover`.
`smallSampleSize`	a `logical` indicating if the AIC small sample size correction formula should be used.

Details

The following formulas are used to compute the AIC and AICc (small sample size correction):

$$AIC = 2k - 2ln(logLik)$$

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

Value

a `list` with the model metrics (the given input) plus a matrix named "AIC" with the AIC for each value

See Also

`compareAIC`, `modelTurnover`

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList <- modelTurnover(x = wormsPE[1:10],
                           assayName = 'fraction',
                           formula = 'fraction ~ 1 - exp(-k*t)',
                           start = list(k = 0.02),
                           mode = 'protein',
                           robust = FALSE,
                           returnModel = TRUE)

modelList <- calculateAIC(modelList, smallSampleSize = TRUE)
```

---

calculateIsotopeFraction

Calculates the incorporated isotope fraction

Description

Calculates the fraction of an isotope ratio using the following formula:

$$Isotope fraction_A = \frac{ratio}{ratio + 1}$$

The ratio should be calculated as:

$$ratio = isotope_A / isotope_B$$

Usage

```
calculateIsotopeFraction(x, ...)

## S4 method for signature 'SilacProteinExperiment'
calculateIsotopeFraction(
  x,
  ratioAssay = "ratio",
  oldIsoAssay,
  newIsoAssay,
  earlyTimepoints,
  lateTimepoints,
  conditionCol
)

## S4 method for signature 'SilacPeptideExperiment'
calculateIsotopeFraction(
  x,
```

```
  ratioAssay = "ratio",
  oldIsoAssay,
  newIsoAssay,
  earlyTimepoints,
  lateTimepoints,
  conditionCol
)

## S4 method for signature 'SilacProteomicsExperiment'
calculateIsotopeFraction(
  x,
  ratioAssay = "ratio",
  oldIsoAssay,
  newIsoAssay,
  earlyTimepoints,
  lateTimepoints,
  conditionCol
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
ratioAssay	A character with the assay name that has the ratio data.
oldIsoAssay	A character with the assay name that has the new isotope intensity data.
newIsoAssay	A character with the assay name that has the old isotope intensity data.
earlyTimepoints	A numeric indicating which timepoints should be considered early.
lateTimepoints	A numeric indicating which timepoints should be considered late.
conditionCol	A character indicating which column of the colData data.frame indicates the different conditions.

Details

If oldIsoAssay and newIsoAssay arguments are given, then the earlyTimepoints and lateTiempoints arguments can be used. These can be used for example if certain proteins do not have any new isotope intensity during the early timepoint. Because of that, no ratio can be calculated and could lead to additional missing values. If old isotope intensity is detected, then a fraction of 0 for new isotope is given. Same principle applies for the late timepoint but with the isotopes in reverse.

Value

a SilacProteinExperiment, SilacPeptideExperiment or SilacProteomicsExperiment object with additional assays named "fraction".

Examples

```
data('wormsPE')
calculateIsotopeFraction(wormsPE)
```

`calculateIsotopeRatio` *Ratio calculation*

Description

Calculates the ratio between the new isotope and the old isotope (new/old).

Usage

```
calculateIsotopeRatio(x, newIsotopeAssay, oldIsotopeAssay, ...)

## S4 method for signature 'SilacProteinExperiment'
calculateIsotopeRatio(x, newIsotopeAssay, oldIsotopeAssay)

## S4 method for signature 'SilacPeptideExperiment'
calculateIsotopeRatio(x, newIsotopeAssay, oldIsotopeAssay)

## S4 method for signature 'SilacProteomicsExperiment'
calculateIsotopeRatio(x, newIsotopeAssay, oldIsotopeAssay)
```

Arguments

`x`	A `SilacProteinExperiment`, `SilacPeptideExperiment` or a `SilacProteomicsExperiment` object.
`newIsotopeAssay`	A `character` indicating the assay name that has the new isotope intensity data.
`oldIsotopeAssay`	A `character` indicating the assay name that has the old isotope intensity data.
`...`	Unused.

Value

A `SilacProteinExperiment`, `SilacPeptideExperiment` or a `SilacProteomicsExperiment` object with an added assay named 'ratio'.

Examples

```
data('wormsPE')
calculateIsotopeRatio(x = wormsPE,
                      newIsotopeAssay = 'int_heavy',
                      oldIsotopeAssay = 'int_light')
```

`calculateOldIsotopePool` *Estimates the fraction of old isotope for each time point*

Description

To estimate how much of the "old" isotope is being used in "new" proteins we can use the expression level of miss-cleaved peptides that contain a mix of isotopes (one old and one new) and miss-cleaved peptides that contain only new isotopes. This can be done using the following formula:

$$\frac{1}{2\frac{Intensity_{lys8lys8}}{Intensity_{lys8lys0}} + 1} = lys_0(Fraction)$$

Which gives an idea of how much recyling (turnover understimation) is happening.

Both peptide types, mix of old/new isotope and two new isotopes, have to be found in a time point to calculate the fraction of old isotope.

Usage

```
calculateOldIsotopePool(x, ...)

## S4 method for signature 'SilacPeptideExperiment'
calculateOldIsotopePool(x, newIsotopeAssayName, mixIsotopeAssayName)

## S4 method for signature 'SilacProteomicsExperiment'
calculateOldIsotopePool(x, newIsotopeAssayName, mixIsotopeAssayName)
```

Arguments

`x`	A `SilacPeptideExperiment` or `SilacProteomicsExperiment` object.
`...`	Unused.
`newIsotopeAssayName`	`character` indicating the assay that contains quantification data for miss-cleaved peptides with two new isotopes incorporated.
`mixIsotopeAssayName`	`character` indicating the assay that contains quantification data for miss-cleaved peptides with one old isotope and one new isotope incorporated.

Value

A `SilacPeptideExperiment` or `SilacProteomicsExperiment` with a peptide assay entry named "oldIsotopePool".

Examples

```
data('wormsPE')
data('recycleLightLysine')
protPE <- ProtExp(wormsPE)
missPE <- addMisscleavedPeptides(x = protPE,
                                 newdata = recycleLightLysine,
                                 idColPept = 'Sequence',
                                 modCol = 'Modifications',
                                 dataCols = c(18:31))

names(assays(missPE))[1:2] <- c('int_lys8lys8', 'int_lys8lys0')
missPE <- calculateOldIsotopePool(x = missPE, 'int_lys8lys8', 'int_lys8lys0')

plotDistributionAssay(missPE, assayName = 'oldIsotopePool')
```

`compareAIC` *Calculates the probability of each model from a set of models.*

Description

For a given set of AIC from models, the probability of each model relative to the rest of the models of the set is calculated using the following formula:

$$\prod AIC_i = \frac{exp(\frac{AIC_{min} - AIC_i}{2})}{\sum_j exp(\frac{AIC_{min} - AIC_j}{2})}$$

Usage

```
compareAIC(...)
```

Arguments

`...`	a `list` with the model metrics, the output from `modelTurnover` and `calculateAIC`.

Value

a `list` with a matrix for each experiment condition. The matrix contains the probabilities of each model (columns) for each protein/peptide (rows).

See Also

`calculateAIC`, `modelTurnover`.

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList1 <- modelTurnover(x = wormsPE[1:10],
                            assayName = 'fraction',
                            formula = 'fraction ~ 1 - exp(-k*t)',
                            start = list(k = 0.02),
                            mode = 'protein',
                            robust = FALSE,
                            returnModel = TRUE)

modelList1 <- calculateAIC(modelList1, smallSampleSize = TRUE)

modelList2 <- modelTurnover(x = wormsPE[1:10],
                            assayName = 'fraction',
                            formula = 'fraction ~ 1 - exp(-k*t) + b',
                            start = list(k = 0.02, b = 0),
                            mode = 'protein',
                            robust = FALSE,
                            returnModel = TRUE)

modelList2 <- calculateAIC(modelList2, smallSampleSize = TRUE)

modelProbabilities <- compareAIC(modelList1, modelList2)
```

---

filterByMissingTimepoints

Filter proteins/peptides by the amount of measurements overtime

---

Description

Searches for proteins/peptides that are not found in all timepoints. This can be done for each condition independently (strict = FALSE) or shared across conditions (strict = TRUE).

Usage

```
filterByMissingTimepoints(x, ...)

## S4 method for signature 'SilacProteinExperiment'
filterByMissingTimepoints(
  x,
  assayName,
  maxMissing = 0,
  strict = TRUE,
  conditionCol,
  returnVector = FALSE
```

```
)

## S4 method for signature 'SilacPeptideExperiment'
filterByMissingTimepoints(
  x,
  assayName,
  maxMissing = 0,
  strict = TRUE,
  conditionCol,
  returnVector = FALSE
)

## S4 method for signature 'SilacProteomicsExperiment'
filterByMissingTimepoints(
  x,
  assayName,
  maxMissing = 0,
  strict = TRUE,
  conditionCol,
  returnVector = FALSE
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
assayName	A character indicating which assay will be used to count the number of missed timepoints.
maxMissing	A numeric indicating how many timepoints are allowed to be missed.
strict	Logical: if TRUE, then proteins have to meet the maxMissing criteria in all conditions and time replicates to pass; if FALSE then proteins only have to meet the maxMissing criteria in one condition or time replicate to pass.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
returnVector	Logical: if TRUE then a vector with the positions to be subset is returned.

Value

A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object or a logical vector with the rows that pass the minimum number of desired timepoints.

Examples

```
data('wormsPE')
filterByMissingTimepoints(wormsPE,
                          assayName = 'ratio',
                          maxMissing = 2,
```

```
                    strict = FALSE)
```

`mefPE` *SilacProteomicsExperiment with pulsed silac data from MEFs*

Description

A pre-built `SilacProteinExperiment` object with data from a pulsed silac experiment done in mouse embryonic fibroblasts (MEFs). Two cell cultures are compared: cultured with or without serum.

Usage

```
data(mefPE)
```

Format

A `SilacProteinExperiment` object with 5223 proteins in a total of 10 samples.

colData A `DataFrame` with the design of the experiment: samples, timepoints, replicates...

assays A list of matrices with quantification data at protein level: ratio and fraction.

rowData A `DataFrame` with 3 columns that general protein id information.

Details

This dataset is used as an example, in the pulsed silac vignette, to show the effect of comparing protein turnover between cell lines growing at different rates.

`merge` *Merge*

Description

Merges two objects of the same class: `SilacProteinExperiment`, `SilacPeptideExperiment` or `SilacProteomicsExperiment`.

Usage

```
## S4 method for signature 'SilacProteinExperiment,ANY'
merge(x, y, by, by.x = by, by.y = by, all = TRUE, ...)

## S4 method for signature 'SilacPeptideExperiment,ANY'
merge(x, y, by, by.x = by, by.y = by, all = TRUE, ...)

## S4 method for signature 'SilacProteomicsExperiment,ANY'
merge(
  x,
  y,
  by.prot,
  by.prot.x = by.prot,
  by.prot.y = by.prot,
  by.pept,
  by.pept.x = by.pept,
  by.pept.y = by.pept,
  all = TRUE,
  ...
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
y	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
by, by.x, by.y	A character indicating the columns used for the merging.
all	A logical indicating if all proteins/peptides should be returned or only the intersect.
...	Further parameters passed into base::merge.
by.prot, by.prot.x, by.prot.y	For SilacProteomicsExperiment objects a character indicating the columns used for the merging of the protein level.
by.pept, by.pept.x, by.pept.y	For SilacProteomicsExperiment objects a character indicating the columns used for the merging of the protein level.

Details

This function is designed to be able to merge different samples from different experiments since it is probable that not the exact same proteins are found in both experiments and therefore cbind cannot be used. It uses the merge base function to merge the rowData data frames and merges the assays based on such merge. The colData data.frame are joined.

For a SilacProteomicsExperiment object it gets a bit more complicated since it is possible that some peptides that were assigned to one protein in one experiment are assigned to another one in another experiment. Therefore the linkerDf data.frame is recalculated.

Value

A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.

Examples

```
data('wormsPE')
merge(wormsPE[1:10, 1:3], wormsPE[3:10, 4:5])
```

---

mergeModelsLists *Merge several models lists.*

Description

Merges several models lists into one. The models lists must come from modelTurnover with the same arguments with the exception of the input data. This function should be used in cases where a model is fit for different conditions of an experiment with NO overlap in the samples and NO missing samples. Otherwise the plotting functions might give incorrect outputs.

Usage

```
mergeModelsLists(...)
```

Arguments

...	Lists with model data, output from modelTurnover.

Details

When merging the attributes are also merged. Some of these need to be recalculated since they contain information about the input data positioning (column number). These attributes are used in the plotting functions.

Take this into consideration so that the order of the models lists follows the order of the columns in the original data. This also means no skipped conditions and no skipped samples. If that is the case, build an intermediari object that contains only the samples to be used (see examples).

Value

A list of models.

See Also

modelTurnover

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList1 <- modelTurnover(x = wormsPE[1:10, 1:7],
                            assayName = 'fraction',
                            formula = 'fraction ~ 1 - exp(-k*t)',
                            start = list(k = 0.02),
                            mode = 'protein',
                            robust = FALSE,
                            returnModel = TRUE)

modelList2 <- modelTurnover(x = wormsPE[1:10, 8:14],
                            assayName = 'fraction',
                            formula = 'fraction ~ 1 - exp(-k*t)',
                            start = list(k = 0.02),
                            mode = 'protein',
                            robust = FALSE,
                            returnModel = TRUE)

mergedModelList <- mergeModelsLists(modelList1, modelList2)
```

`modelTurnover` *Estimate protein/peptide turnover*

Description

Method to apply turnover models on protein/peptide data

Usage

```
modelTurnover(x, ...)

## S4 method for signature 'SilacProteinExperiment'
modelTurnover(
  x,
  assayName = "fraction",
  formula = "fraction ~ 1-exp(-k*t)",
  start = list(k = 0.02),
  robust = FALSE,
  mode = "protein",
  verbose = FALSE,
  returnModel = FALSE,
  conditionCol,
  timeCol,
  silent = TRUE,
  ...
```

```
)

## S4 method for signature 'SilacPeptideExperiment'
modelTurnover(
  x,
  assayName = "fraction",
  formula = "fraction ~ 1-exp(-k*t)",
  start = list(k = 0.02),
  robust = FALSE,
  mode = c("grouped", "peptide"),
  verbose = FALSE,
  returnModel = FALSE,
  conditionCol,
  timeCol,
  proteinCol,
  silent = TRUE,
  ...
)

## S4 method for signature 'SilacProteomicsExperiment'
modelTurnover(
  x,
  assayName = "fraction",
  formula = "fraction ~ 1-exp(-k*t)",
  start = list(k = 0.02),
  robust = FALSE,
  mode = c("protein", "grouped", "peptide"),
  verbose = FALSE,
  returnModel = FALSE,
  conditionCol,
  timeCol,
  proteinCol,
  silent = TRUE,
  ...
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or SilacProteomicsExperiment object.
...	further parameters passed into nls or nlrob.
assayName	character indicating which assay to use as data input for the model.
formula	formula to be used. Time must always be named "t" and the data must be named "fraction".
start	named list with the initical values for the parameters in formula.
robust	logical indicating if robust modelling from the robustbase package should be used.

mode	character indicating which type of data should be used. Can be "protein": one model per protein; "grouped": one model per protein using peptide data; "peptide" one model per peptide.
verbose	logical indicating if a progress bar should be printed.
returnModel	logical indicating if the model objects should be returned also in the output.
conditionCol	character indicating which column of colData(x) describes the conditions.
timeCol	character indicating which column of colData(x) describes time.
silent	logical indicating if the errors given by nls/nlrob should be printed.
proteinCol	character indicating which column of rowData(x) describes the assigned protein to a peptide. (Only for peptide data)

Details

The nls and nlrob functions have many arguments that can be tunned for parameter fitting. Unfortunately, not all the possible argument combinations have been tested. It is recommended to first test one model with the desired parameters with silent = FALSE to see that it runs smoothly and then run the whole proteome with silent = TRUE to supress failed convergence errors. For example, some methods for nlrob use upper and lower bounds instead of start.

Please open an issue on github if the function is having trouble with a particular argument.

For robust modelling the method 'CM' and 'mtl' are not yet supported.

Value

A named list with either model metrics in matrices or the model objects.

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList <- modelTurnover(x = wormsPE[1:10],
                           assayName = 'fraction',
                           formula = 'fraction ~ 1 - exp(-k*t)',
                           start = list(k = 0.02),
                           mode = 'protein',
                           robust = FALSE,
                           returnModel = TRUE,
                           silent = TRUE)
```

mostStable *Most stable proteins/peptides*

Description

Finds which are the most stable proteins/peptides across the entire experiment. These proteins/peptides can be used to estimate the cell growth of each condition.

Usage

```
mostStable(x, ...)

## S4 method for signature 'SilacProteinExperiment'
mostStable(x, assayName, n, conditionCol)

## S4 method for signature 'SilacPeptideExperiment'
mostStable(x, assayName, n, conditionCol)

## S4 method for signature 'SilacProteomicsExperiment'
mostStable(x, assayName, n, mode, conditionCol)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
assayName	Name of the assay to use.
n	A numeric indicating how many proteins should be returned.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
mode	A character indicating which level of data to use, either "protein" or "peptide". Only relevant for ProteomicsExperiment inputs.

Details

Proteins/peptides that are not found in all timepoints and in all conditions are not considered. The stability is based on ranking heavy label incorporation for each timepoint; therefore, lower values are correlated to higher stability.

Value

A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object with the n most stable proteins/peptides.

Examples

```
data('mefPE')
mostStable(mefPE, assayName = 'fraction', n = 50)
```

`plotDistributionAssay` *Distribution of assay data per condition and timepoint.*

Description

Plot the distribution of the data stored in an assay using boxplots or density distributions.

Usage

```
plotDistributionAssay(x, ...)

## S4 method for signature 'SilacProteinExperiment'
plotDistributionAssay(
  x,
  assayName,
  plotType = "boxplot",
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)

## S4 method for signature 'SilacPeptideExperiment'
plotDistributionAssay(
  x,
  assayName,
  plotType = "boxplot",
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)

## S4 method for signature 'SilacProteomicsExperiment'
plotDistributionAssay(
  x,
  assayName,
  mode = "protein",
  plotType = "boxplot",
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
assayName	Name of the assay to use in the plot.
plotType	A character indicating which geometry to plot: 'boxplot' or 'density'. (default = 'density')
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
timeCol	A character, which indicates the column name in colData(x) that defines the different timepoints.
mode	A character indicating which level of data to use, either "protein" or "peptide". Only relevant for ProteomicsExperiment inputs.

Value

A ggplot2 object or a data.frame with the data that would be plotted.

Examples

```
data('wormsPE')
plotDistributionAssay(wormsPE, assayName = 'ratio')
```

---

plotDistributionModel *Distribution of modelling output*

Description

Plot the distribution of the different model parameters and metrics for each condition.

Usage

```
plotDistributionModel(
  modelList,
  value = "param_values",
  plotType = "density",
  returnDataFrame = FALSE
)
```

Arguments

modelList	A list containing all the model objects, this should be the output of modelTurnover.
value	A character indicating which metric to plot. Check names(modelList) for available options. (Default = 'param_values')
plotType	A character indicating which geometry to plot: 'boxplot' or 'density'. (default = 'density')
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.

Value

A ggplot density or boxplot object, or the data.frame used to make the plot.

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList <- modelTurnover(x = wormsPE[1:10],
                           assayName = 'fraction',
                           formula = 'fraction ~ 1 - exp(-k*t)',
                           start = list(k = 0.02),
                           mode = 'protein',
                           robust = FALSE,
                           returnModel = TRUE)

plotDistributionModel(modelList = modelList,
                      value = 'param_values',
                      plotType = 'density')
```

---

plotIndividualModel *Fitted model(s) for a feature*

Description

Plot the model fit for a specific protein/peptide in different conditions.

Usage

```
plotIndividualModel(x, ...)

## S4 method for signature 'SilacProteinExperiment'
plotIndividualModel(x, modelList, num, returnDataFrame = FALSE)

## S4 method for signature 'SilacPeptideExperiment'
```

```
plotIndividualModel(x, modelList, num, returnDataFrame = FALSE)

## S4 method for signature 'SilacProteomicsExperiment'
plotIndividualModel(x, modelList, num, returnDataFrame = FALSE)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or SilacProteomicsExperiment object.
...	Unused.
modelList	A list containing all the model objects, this should be the output of modelTurnover with returnModel as TRUE.
num	The feature number to be plotted.
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.

Value

A scatter plot with a fitted line or a data.frame.

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList <- modelTurnover(x = wormsPE[1:10],
                           assayName = 'fraction',
                           formula = 'fraction ~ 1 - exp(-k*t)',
                           start = list(k = 0.02),
                           mode = 'protein',
                           robust = FALSE,
                           returnModel = TRUE)

plotIndividualModel(x = wormsPE,
                    modelList = modelList,
                    num = 2)
```

---

recycleLightLysine *Data frame with miss cleaved peptides quantifications from MaxQuant*

Description

A data.frame that contains the output from searching heavy label incorporation as amino acid modifications. This search was done using the same data as in the wormsPE data.

Usage

```
data(recycleLightLysine)
```

Format

A `data.frame`

Sequence Column used as id to match with the peptides in the `wormsPE` ProteomicsExperiment.

Modifications Column indicating how many heavy isotopes the peptide has.

Intensity.* Columns containing quantification data in each sample.

... Other columns containing peptide related information.

Details

The reason why the search was done using isotopes as modifications is because MaxQuant only looks for peptides in which all amino acids are isotope labelled, miss-cleaved amino acids that contain an isotope mix are not measured by the normal silac search engine. This allows to find peptides which a mix of incoporated isotopes.

This dataset is used as an example, in the pulsed silac vignette, to estimate the amount of old isotope label in newly synthesized proteins (amino acid recycling).

References

https://www.ncbi.nlm.nih.gov/pubmed/28679685

`scatterCompareAssays` *Scatter plot of two conditions for each timepoint of an assay.*

Description

Scatter plot of two conditions/replicates for a selected assay. Timepoints are separated using `facet_wrap`.

Usage

```
scatterCompareAssays(x, ...)

## S4 method for signature 'SilacProteinExperiment'
scatterCompareAssays(
  x,
  conditions,
  assayName,
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)
```

```
## S4 method for signature 'SilacPeptideExperiment'
scatterCompareAssays(
  x,
  conditions,
  assayName,
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)

## S4 method for signature 'SilacProteomicsExperiment'
scatterCompareAssays(
  x,
  conditions,
  assayName,
  mode = "protein",
  returnDataFrame = FALSE,
  conditionCol,
  timeCol
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Unused.
conditions	A character of length 2 indicating which 2 conditions should be compared.
assayName	Name of the assay to use in the plot.
returnDataFrame	A logical indicating if the data.frame used for the plot should be returned instead.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
timeCol	A character, which indicates the column name in colData(x) that defines the different timepoints.
mode	A character indicating which level of data to use, either "protein" or "peptide". Only relevant for ProteomicsExperiment inputs.

Value

A ggplot object or the data.frame that would be used instead in the plot.

Examples

```
data('wormsPE')
scatterCompareAssays(x = wormsPE,
                     conditions = c('OW40', 'OW450'),
```

```
                    assayName = 'ratio',
                    mode = 'protein')
```

`scatterCompareModels` *Scatter plot of two conditions for a model metric.*

Description

Scatter plot of two conditions/replicates for a selected metric of a model. For example to compare turnover rates, model errors... `facet_wrap`.

Usage

```
scatterCompareModels(
  modelList,
  conditions,
  value = "param_values",
  returnDataFrame = FALSE
)
```

Arguments

`modelList`	A list containing all the model objects, this should be the output of `modelTurnover` with returnModel as TRUE.
`conditions`	A `character` of length 2 indicating which 2 conditions should be compared.
`value`	A `character` indicating which metric to plot. Check `names(modelList)` for available options. (Default = 'param_values')
`returnDataFrame`	A `logical` indicating if the `data.frame` used for the plot should be returned instead.

Value

A `ggplot` object or the `data.frame` that would be used instead in the plot.

Examples

```
data('wormsPE')
wormsPE <- calculateIsotopeFraction(wormsPE, ratioAssay = 'ratio')

modelList <- modelTurnover(x = wormsPE[1:10],
                           assayName = 'fraction',
                           formula = 'fraction ~ 1 - exp(-k*t)',
                           start = list(k = 0.02),
                           mode = 'protein',
                           robust = FALSE,
                           returnModel = TRUE)
```

```
scatterCompareModels(modelList = modelList,
                     conditions = c('OW40', 'OW450'),
                     value = 'param_values')
```

SilacPeptideExperiment-class
SilacPeptideExperiment class

Description

S4 class that extends the SummarizedExperiment class. This class is designed for proteomics data, more especifically peptide level data. The metadata slot comes already initialized with the metaoptions (see details).

Details

The SilacPeptideExperiment class has been designed to store peptide level data and to be used in the functions provided in this package for pulsed SILAC data analysis; in combination with the other two classes from the package: the SilacProteinExperiment and SilacProteomicsExperiment classes.

ProteinExperiment metaoptions are stored in the metadata slot This contains a list with some parameters that are automatically initialized by the constructor. Some parameters are mandatory for certain functions or operations. The user can add or remove items at their discretion. These parameters are meant to help automate certain pipeline or data analysis steps. These metaoptions are:

conditionCol character indicating the column name of colData(x) that defines the different experiment conditions.

timeCol character indicating the column name of colData(x) that defines the different time-points of the experiment.

proteinCol character indicating the column name of rowData(x) that defines to which protein a peptide is assigned.

Constructor

See SilacPeptideExperiment-constructor for details.

Accessors

See SilacProteinPeptideExperiment-accessors for details.

See Also

SilacPeptideExperiment-constructor, SilacProteinPeptideExperiment-accessors, SummarizedExperiment

SilacPeptideExperiment-constructor
SilacPeptideExperiment constructor

Description

Constructor function for the SilacPeptideExperiment class object.

Usage

```
SilacPeptideExperiment(
  assays,
  rowData = NULL,
  colData = NULL,
  conditionCol = NA,
  timeCol = NA,
  proteinCol = NA,
  metadata = NULL
)
```

Arguments

assays	A named list of matrices (assays) with peptide level data.
rowData	A data.frame with peptide feature data like protein names, molecular weight, etc.
colData	A data.frame with sample information like conditions, replicates, etc.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
timeCol	A character, which indicates the column name in colData(x) that defines the different experiment timepoints.
proteinCol	A character, which indicates the column name in rowData(x) that defines to which protein a peptide is assigned.
metadata	A list to store any kind of experiment-wide data; like authors, dates, machines used...

Value

An object of class SilacPeptideExperiment.

Class description

See SilacPeptideExperiment-class for details.

Accessors

See SilacProteinPeptideExperiment-accessors for details.

Examples

```
## assays
assays_peptide <- list(expression = matrix(1:15, ncol = 3))

## colData
colData <- data.frame(sample = c('A1', 'A2', 'A3'),
                      condition = c('A', 'A', 'A'),
                      time = c(1, 2, 3))
## rowData
rowData_peptide <- data.frame(pept_id = letters[1:5],
                              prot_id = c('A', 'A', 'B', 'C', 'C'))
## construct the ProteinExperiment
peptExp <- SilacPeptideExperiment(assays = assays_peptide,
                                  rowData = rowData_peptide,
                                  colData = colData,
                                  conditionCol = 'condition',
                                  timeCol = 'time')
```

SilacProteinExperiment-class

SilacProteinExperiment class

Description

S4 class that extends the SummarizedExperiment class. This class is designed for proteomics data, more especifically protein level data. The metadata slot comes already initialized with the metaoptions (see details).

Details

The SilacProteinExperiment class has been designed to store protein level data and to be used in the functions provided in this package for pulsed SILAC data analysis; in combination with the other two classes from the package: the SilacPeptideExperiment and SilacProteomicsExperiment classes.

SilacProteinExperiment metaoptions are stored in the metadata slot This contains a list with some parameters that are automatically initialized by the constructor. Some parameters are mandatory for certain functions or operations. The user can add or remove items at their discretion. These parameters are meant to help automate certain pipeline or data analysis steps. These metaoptions are:

conditionCol character indicating the column name of colData(x) that defines the different experiment conditions.

timeCol character indicating the column name of colData(x) that defines the different time-points of the experiment.

Constructor

See SilacProteinExperiment-constructor for details.

Accessors

See SilacProteinPeptideExperiment-accessors for details.

See Also

SilacProteinExperiment-constructor, SilacProteinPeptideExperiment-accessors, SummarizedExperiment

---

SilacProteinExperiment-constructor

SilacProteinExperiment constructor

Description

Constructor function for the SilacProteinExperiment class object.

Usage

```
SilacProteinExperiment(
  assays,
  rowData = NULL,
  colData = NULL,
  conditionCol = NA,
  timeCol = NA,
  metadata = NULL
)
```

Arguments

assays	A named list of matrices (assays) with protein level data.
rowData	A data.frame with protein feature data like protein names, molecular weight, etc.
colData	A data.frame with sample information like conditions, replicates, etc.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
timeCol	A character, which indicates the column name in colData(x) that defines the different experiment timepoints.
metadata	A list to store any kind of experiment-wide data; like authors, dates, machines used...

Value

An object of class SilacProteinExperiment.

Class description

See SilacProteinExperiment-class for details.

Accessors

See SilacProteinPeptideExperiment-accessors for details.

Examples

```
## assays
assays_protein <- list(expression = matrix(1:9, ncol = 3))

## colData
colData <- data.frame(sample = c('A1', 'A2', 'A3'),
                      condition = c('A', 'A', 'A'),
                    time = c(1, 2, 3))
## rowData
rowData_protein <- data.frame(prot_id = LETTERS[1:3])

## construct the ProteinExperiment
protExp <- SilacProteinExperiment(assays = assays_protein,
                                  rowData = rowData_protein,
                                  colData = colData,
                                  conditionCol = 'condition',
                                  timeCol = 'time')
```

SilacProteinPeptideExperiment-accessors

Accessors for the SilacProteinExperiment and SilacPeptideExperiment classes

---

Description

All the accessors, dimension, subsetting, merging and coercers that work on `SilacProteinExperiment` and `SilacPeptideExperiment` objects. Functions that work on `SummarizedExperiment` objects should also work on these two objects. Detailed examples of these functions can be found in the vignette of this package.

Usage

```
## S4 replacement method for signature 'SilacProteinExperiment,ANY'
assays(x) <- value

## S4 replacement method for signature 'SilacPeptideExperiment,ANY'
assays(x) <- value

## S4 method for signature 'SilacProteinExperiment'
cbind(..., deparse.level = 1)

## S4 method for signature 'SilacPeptideExperiment'
cbind(..., deparse.level = 1)
```

```
## S4 replacement method for signature 'SilacProteinExperiment,ANY'
colData(x, ...) <- value

## S4 replacement method for signature 'SilacPeptideExperiment,ANY'
colData(x, ...) <- value

## S4 method for signature 'SilacProteinExperiment'
metaoptions(x)

## S4 replacement method for signature 'SilacProteinExperiment'
metaoptions(x) <- value

## S4 method for signature 'SilacPeptideExperiment'
metaoptions(x)

## S4 replacement method for signature 'SilacPeptideExperiment'
metaoptions(x) <- value

## S4 method for signature 'SilacProteinExperiment'
rbind(..., deparse.level = 1)

## S4 method for signature 'SilacPeptideExperiment'
rbind(..., deparse.level = 1)

## S4 replacement method for signature 'SilacProteinExperiment'
rowData(x, ...) <- value

## S4 replacement method for signature 'SilacPeptideExperiment'
rowData(x, ...) <- value

## S4 method for signature 'SilacProteinExperiment'
subset(x, ...)

## S4 method for signature 'SilacPeptideExperiment'
subset(x, ...)
```

Arguments

x	A SilacProteinExperiment or a SilacPeptideExperiment object.
value	An object of class specified in the S4 method signature or as described in the following sections.
...	For rbind and cbind are SilacProteinExperiment or SilacPeptideExperiment objects to be joined together. For subset it is a logical comparison using a column name from the respective rowData data.frame. Otherwise unused.
deparse.level	Unused.

Value

Elements from `SilacProteinExperiment` or `SilacPeptideExperiment` objects.

Accessors

The following functions can be used to access the data in the class slots

`assays`: Access the assays (list of matrices) of the object. Value should be a matrix or list of matrices.

`assayNames` Access the assay names of the object. Value should be a character vector.

`rowData` Access the protein/peptide feature `data.frame` of the object. Value should be `data.frame` with as many rows as proteins/peptides.

`colData`: Access the samples `data.frame` of the object. Value should be a `data.frame` with as many rows as samples.

`metadata`: Access the metadata `list` of the object. Value should be a `list`.

`metaoptions`: Access the metaoptions `list` of the object. Value should be a `list`.

Dimensions

The following functions can be used to get the number of proteins/peptides and number of samples:

`nrow`: Gives how many proteins and/or peptides the object has.

`ncol`: Gives how many samples the object has.

`dim`: Gives the number of proteins/peptides and the number of samples the object has.

`length`: Gives how many proteins and/or peptides the object has.

Subsetting

The following functions can be used to subset the different classes:

`$`: Gives a column from colData by name.

`'['`: Can be used to subset by row and column.

`subset`: Allows to subset based on a logical comparison using a column name from the rowData `data.frame`.

Merging

The following functions can be used to aggregate objects of the same class together:

`cbind`: Joins two or more objects horizontally (adding samples). Must have the same proteins/peptides and in the same order.

`rbind`: Joins two or more objects vertically (adding proteins/peptides). Must have the same samples and in the same order.

`merge`: Joins two objects by adding new samples and tries to merge the proteins/peptide rowData data.frames.

Merge methods are explained in detail in merge.

Coercers

The folloing functions can be used to transform a SilacProteinExperiment or a SilacPeptideExperiment into a SummarizedExperiment or a data.frame.

as(x, 'SummarizedEpriment'): Transforms the object into an object of class SummarizedExperiment.

as(x, 'data.frame'): Transforms the object into an object of class data.frame.

Examples

```
data('wormsPE')
protPE <- ProtExp(wormsPE)

# Accessors
## assays
assays(protPE)

## assaysNames
assayNames(protPE)

## colData
colData(protPE)

## rowData
rowData(protPE)

## metadata
metadata(protPE)

## metaoptions
#metaoptions(protPE)


# Dimensions
nrow(protPE)
ncol(protPE)
dim(protPE)
length(protPE)

# Subsetting
protPE$line
protPE[1,1]
subset(protPE, protein_id == 'AC3.2')

# Merging
rbind(protPE[1:10, ], protPE[11:20, ])
cbind(protPE[,1:2], protPE[,3:4])
#merge(protPE[1:10, 1:3], protPE[3:10, 4:5])

# Coercers
as(protPE, 'SummarizedExperiment')
as(protPE, 'data.frame')
```

SilacProteomicsExperiment-accessors
Accessors for the SilacProteomicsExperiment class

Description

All the accessors, dimension, subsetting, merging and coercers that work on `SilacProteomicsExperiment` objects. Since the `SilacProteomicsExperiment` object has both protein and peptide level data, most of the functions have a 'Prot' or 'Pept' suffix to indicate which level should be used. If the non-suffix function is used, then a list with both protein and peptide data is returned. These functions also work on `SilacProteinExperiment` and `SilacPeptideExperiment` objects.

Usage

```
PeptExp(x)

ProtExp(x)

## S4 method for signature 'SilacProteomicsExperiment'
assayNames(x, ..., withDimnames)

## S4 method for signature 'SilacProteinExperiment'
assayNamesProt(x)

## S4 replacement method for signature 'SilacProteinExperiment'
assayNamesProt(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
assayNamesProt(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
assayNamesProt(x) <- value

## S4 method for signature 'SilacPeptideExperiment'
assayNamesPept(x)

## S4 replacement method for signature 'SilacPeptideExperiment'
assayNamesPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
assayNamesPept(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
assayNamesPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
assays(x, withDimnames = TRUE, ...)
```

```
## S4 method for signature 'SilacProteinExperiment'
assaysProt(x)

## S4 replacement method for signature 'SilacProteinExperiment'
assaysProt(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
assaysProt(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
assaysProt(x) <- value

## S4 method for signature 'SilacPeptideExperiment'
assaysPept(x)

## S4 replacement method for signature 'SilacPeptideExperiment'
assaysPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
assaysPept(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
assaysPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
cbind(..., deparse.level = 1)

## S4 method for signature 'SilacProteomicsExperiment'
colData(x, ...)

## S4 replacement method for signature 'SilacProteomicsExperiment,ANY'
colData(x, ...) <- value

## S4 method for signature 'SilacProteomicsExperiment'
colnames(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
colnames(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
dim(x)

## S4 method for signature 'SilacProteomicsExperiment'
x$name

## S4 replacement method for signature 'SilacProteomicsExperiment'
x$name <- value
```

```
## S4 method for signature 'SilacProteomicsExperiment'
length(x)

## S4 method for signature 'SilacProteomicsExperiment'
linkerDf(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
linkerDf(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
metadata(x, ...)

## S4 replacement method for signature 'SilacProteomicsExperiment'
metadata(x, ...) <- value

## S4 method for signature 'SilacProteomicsExperiment'
metaoptions(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
metaoptions(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
ncol(x)

## S4 method for signature 'SilacProteomicsExperiment'
rbind(..., deparse.level = 1)

## S4 method for signature 'SilacProteomicsExperiment'
rowData(x, use.names = TRUE, ...)

## S4 replacement method for signature 'SilacProteomicsExperiment'
rowData(x, ...) <- value

## S4 method for signature 'SilacProteinExperiment'
rowDataProt(x)

## S4 replacement method for signature 'SilacProteinExperiment'
rowDataProt(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
rowDataProt(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
rowDataProt(x) <- value

## S4 method for signature 'SilacPeptideExperiment'
rowDataPept(x)
```

```
## S4 replacement method for signature 'SilacPeptideExperiment'
rowDataPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
rowDataPept(x)

## S4 method for signature 'SilacProteomicsExperiment'
rownamesProt(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
rownamesProt(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
rownamesPept(x)

## S4 replacement method for signature 'SilacProteomicsExperiment'
rownamesPept(x) <- value

## S4 method for signature 'SilacProteomicsExperiment'
subset(x, ...)

## S4 method for signature 'SilacProteinExperiment'
subsetProt(x, ...)

## S4 method for signature 'SilacProteomicsExperiment'
subsetProt(x, ...)

## S4 method for signature 'SilacPeptideExperiment'
subsetPept(x, ...)

## S4 method for signature 'SilacProteomicsExperiment'
subsetPept(x, ...)

## S4 method for signature 'SilacProteomicsExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

x	A SilacSilacProteomicsExperiment object.
...	For rbind and cbind are SilacProteinExperiment or SilacPeptideExperiment objects to be joined together. For subset it is a logical comparison using a column name from the respective rowData data.frame. Otherwise unused.
withDimnames	Unused.
value	An object of class specified in the S4 method signature or as described in the following sections.
deparse.level	Unused.

name	Column name of colData.
use.names	Unused.
i, j	For `[`, i, j are subscripts that can act to subset the rows and columns of x.
drop	A logical indicating if dimensions should be lowered if possible when subsetting.

Value

Elements from a SilacProteomicsExperiment object.

Accessors

The following functions can be used to access the data in the class slots

assays, assaysProt, assaysPept: Access the assays (list of matrices) of the object. Value should be a matrix or list of matrices.

assayNames, assayNamesProt, assayNamesPept: Access the assay names of the object. Value should be a character vector.

rowData, rowDataProt, rowDataPept: Access the protein/peptide feature data.frame of the object. Value should be a data.frame with as many rows as proteins/peptides.

colData: Access the samples data.frame of the object. Value should be a data.frame with as many rows as samples.

metadata: Access the metadata list of the object. Value should be a list.

metaoptions: Access the metaoptions list of the object. Value should be a list.

linkerDf: Access the linker data.frame of the object (only ProteomicsExperiment). Value should be a data.frame output from buildLinkerDf.

ProtExp **and** PeptExp: Access the experiment objects in a SilacProteomicsExperiment.

Dimensions

The following functions can be used to get the number of proteins/peptides and number of samples:

nrow: Gives how many proteins and peptides the object has.

ncol: Gives how many samples the object has.

dim: Gives both the number of proteins andpeptides and the number of samples the object has.

length: Gives how many proteins and or peptides the object has.

Subsetting

The following functions can be used to subset the different classes:

$: Gives a column from colData by name.

'[': Can be used to subset by row and column.

subset, subsetProt **and** subsetPept: Allows to subset based on a logical comparison using a column name from the rowData data.frame.

The ProteomicsExperiment class is a bit more complex since there are two levels at which the subset can be done and these two levels can be linked or not.

If the metaoption 'linkedSubset' is TRUE, then when subsetting on one level, the proteins/peptide linked to such level are also subsetted. Otherwise, one of the levels remains unmodified.

`subsetProt` can be used to apply `subset` at the rowData `data.frame` of the protein level. `subsetProt` can be used to apply `subset` at the rowData `data.frame` of the peptide level. If `subset` is used, then `subsetProt` or `subsetPept` will be used depending on the metaoption 'subsetMode'.

'[' acts in the same manner as calling `subset`. In this case numerics are used and samples can also be selected.

The vignette offers a detailed set of simple examples with all the possible cases.

Merging

The following functions can be used to aggregate objects of the same class together:

`cbind`**:** Joins two or more objects horizontally (adding samples). Must have the same proteins/peptides and in the same order.

`rbind`**:** Joins two or more objects vertically (adding proteins/peptides). Must have the same samples and in the same order.

`merge`**:** Joins two objects by adding new samples and tries to merge the proteins/peptide rowData data.frames and recalculate the linkerDf `data.frame` for the `SilacProteomicsExperiment` class.

Merge methods are explained in detail in merge.

Examples

```
# Accessors
## assays
data('wormsPE')
assays(wormsPE)
assaysProt(wormsPE)
assaysPept(wormsPE)

## assaysNames
assayNames(wormsPE)
assayNamesProt(wormsPE)
assayNamesPept(wormsPE)

## colData
colData(wormsPE)

## rowData
rowData(wormsPE)
rowDataProt(wormsPE)
rowDataPept(wormsPE)

## metadata
metadata(wormsPE)
```

```
## metaoptions
#metaoptions(wormsPE)

## linkerDf
linkerDf(wormsPE)

# Dimensions and dimensions names
nrow(wormsPE)
ncol(wormsPE)
dim(wormsPE)
length(wormsPE)
colnames(wormsPE)
rownamesProt(wormsPE)
rownamesPept(wormsPE)

# Subsetting
wormsPE$line
wormsPE[1,1]
subsetProt(wormsPE, protein_id == 'AC3.2')
subsetPept(wormsPE, Sequence == 'AIQEISDYHFLIK')

# Merging
rbind(wormsPE[1:10, ], wormsPE[11:20, ])
cbind(wormsPE[,1:2], wormsPE[,3:4])
#merge(wormsPE[1:10, 1:3], wormsPE[3:10, 4:5])
```

`SilacProteomicsExperiment-class` *SilacProteomicsExperiment class*

Description

S4 class that contains a `SilacProteinExperiment` object and a `SilacPeptideExperiment` object. The two objects are linked by a `data.frame` (linkerDf). This class can be used to manage both protein and peptide data at the same time.

Details

The `SilacProteomicsExperiment` object is just a `SilacProteinExperiment` object and a `SilacPeptideExperiment` object together.

The rows of the `SilacProteinExperiment` object represent proteins. The rows of the `SilacPeptideExperiment` object represent peptides.

The columns of the `SilacProteomicsExperiment` object represent samples. Samples are shared at both protein and peptide levels.

Experiment-wide information can be stored in the `metadata` slot, which is accessed with the `metadata` function. This contains a `list` object in which each item is left to the discretion of the user. Some possible examples could be: data of the experiment, author, machine used, etc.

SilacProteomicsExperiment options are stored in the `metadata` slot. This contains a `list` with some parameters that are automatically initialized by the constructor. Some parameters are mandatory for certain functions or operations. The user can add or remove items at their discretion. These parameters are meant to help automate certain pipeline or data analysis steps. These metaoptions are: These metaoptions are:

conditionCol `character` indicating the column name of `colData(x)` that defines the different experiment conditions.

timeCol `character` indicating the column name of `colData(x)` that defines the different time-points of the experiment.

idColProt A `character` indicating which column from the rowData (protein) should be used as ids.

idColPept A `character` indicating which column from the rowData (peptides) should be used as ids.

linkedSubset A `logical` if subsetting should be linked between proteins and peptide.

subsetMode A `character`, either 'protein' or 'peptide' indicating which level should be used first when subsetting.

Slots

`SilacProteinExperiment` Contains a `SilacProteinExperiment` object.

`SilacPeptideExperiment` Contains a `SilacPeptideExperiment` object.

`colData` Contains a `data.frame` with sample information like conditions, replicates, etc.

`linkerDf` Contains a `data.frame` that has been created with `buildLinkerDf`. It contains the relationships between proteins and peptides.

`metadata` Contains a `list` to store any kind of experiment-wide data and the metaoptions.

Constructor

See SilacProteomicsExperiment-constructor for details.

Accessors

See SilacProteomicsExperiment-accessors for details.

See Also

`SilacProteomicsExperiment-constructor`, `SilacProteomicsExperiment-accessors`, `SilacProteinExperiment`, `SilacPeptideExperiment`

SilacProteomicsExperiment-constructor
SilacProteomicsExperiment constructor

Description

Constructor function for the ProteomicsExperiment class object. It requires at minimum a `SilacProteinExperiment` and a `SilacPeptideExperiment`. If the colData, metadata and metaoptions have been already defined in those it is not necessary to give them again.

Usage

```
SilacProteomicsExperiment(
  SilacProteinExperiment,
  SilacPeptideExperiment,
  colData,
  linkerDf,
  metadata,
  idColProt = NA,
  idColPept = NA,
  linkedSubset = TRUE,
  subsetMode = "protein",
  conditionCol = NA,
  timeCol = NA,
  proteinCol = NA
)
```

Arguments

`SilacProteinExperiment`	A `SilacProteinExperiment` object.
`SilacPeptideExperiment`	A `SilacPeptideExperiment` object.
`colData`	A `data.frame` with sample information like conditions, replicates, etc. If not provided uses the colData slot from the `SilacProteinExperiment` and `SilacPeptideExperiment`.
`linkerDf`	A `data.frame` output from `buildLinkerDf`.
`metadata`	A `list` to store any kind of experiment-wide data; like authors, dates, machines used... If not provided uses the metadata from the `SilacProteinExperiment` and `SilacPeptideExperiment`.
`idColProt`	A `character` indicating which column from the rowData (protein) should be used as ids. Should be the same used in `buildLinkerDf`.
`idColPept`	A `character` indicating which column from the rowData (peptide) should be used as ids. Should be the same used in `buildLinkerDf`.
`linkedSubset`	A `logical` if subsetting should be linked between proteins and peptide.

subsetMode	A character, either 'protein' or 'peptide' indicating which level should be used first when subsetting.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
timeCol	A character, which indicates the column name in colData(x) that defines the different experiment timepoints.
proteinCol	A character, which indicates the column name in rowData(x) that defines to which protein a peptide is assigned.

Value

An object of class SilacProteomicsExperiment.

Class description

See SilacProteomicsExperiment-class for details.

Accessors

See SilacProteomicsExperiment-accessors for details.

Examples

```
## assays
assays_protein <- list(expression = matrix(1:9, ncol = 3))

## colData
colData <- data.frame(sample = c('A1', 'A2', 'A3'),
                      condition = c('A', 'A', 'A'),
                     time = c(1, 2, 3))
## rowData
rowData_protein <- data.frame(prot_id = LETTERS[1:3])

## construct the ProteinExperiment
protExp <- SilacProteinExperiment(assays = assays_protein,
                                  rowData = rowData_protein,
                                  colData = colData,
                                  conditionCol = 'condition',
                                  timeCol = 'time')

## assays
assays_peptide <- list(expression = matrix(1:15, ncol = 3))

## colData
colData <- data.frame(sample = c('A1', 'A2', 'A3'),
                      condition = c('A', 'A', 'A'),
                      time = c(1, 2, 3))
## rowData
rowData_peptide <- data.frame(pept_id = letters[1:5],
                              prot_id = c('A', 'A', 'B', 'C', 'C'))
## construct the ProteinExperiment
```

```
peptExp <- SilacPeptideExperiment(assays = assays_peptide,
                                  rowData = rowData_peptide,
                                  colData = colData,
                                  conditionCol = 'condition',
                                  timeCol = 'time')

## list with the relationships
protein_to_peptide <- list(A = c('a', 'b'), B = c('c'), C = c('d', 'e'))
## function to build the data.frame
linkerDf <- buildLinkerDf(protIDs = LETTERS[1:3],
                          pepIDs  = letters[1:5],
                          protToPep = protein_to_peptide)

ProteomicsExp <- SilacProteomicsExperiment(SilacProteinExperiment = protExp,
                                           SilacPeptideExperiment = peptExp,
                                           linkerDf = linkerDf)
```

---

upsetTimeCoverage *Number of detected features per sample*

Description

How many proteins/peptides are detected in each sample. Anything else than NA is considered detected.

Usage

```
upsetTimeCoverage(x, ...)

## S4 method for signature 'SilacProteinExperiment'
upsetTimeCoverage(
  x,
  assayName,
  conditionCol,
  maxMissing = 0,
  returnList = FALSE,
  ...
)

## S4 method for signature 'SilacPeptideExperiment'
upsetTimeCoverage(
  x,
  assayName,
  maxMissing = 0,
  conditionCol,
  returnList = FALSE,
  ...
```

```
)

## S4 method for signature 'SilacProteomicsExperiment'
upsetTimeCoverage(
  x,
  assayName,
  maxMissing = 0,
  conditionCol,
  returnList = FALSE,
  ...
)
```

Arguments

x	A SilacProteinExperiment, SilacPeptideExperiment or a SilacProteomicsExperiment object.
...	Further arguments passed to upset().
assayName	Name of the assay to use in the plot.
conditionCol	A character, which indicates the column name in colData(x) that defines the different experiment conditions.
maxMissing	A numerical indicating how many timepoints can a protein/peptide miss.
returnList	A logical indicating if the list used for the plot should be returned instead.

Value

A barplot or a data.frame.

Examples

```
data('wormsPE')
upsetTimeCoverage(x = ProtExp(wormsPE),
                assayName = 'ratio',
                maxMissing = 2)
```

---

wormsPE *ProteomicsExperiment with pulsed silac data from C. elegans strains*

Description

A pre-built SilacProteomicsExperiment object with data from a pulsed silac experiment done in *C. elegans* by Visscher et al. 2016. It only contains the data from the first 250 priteins and two old worms strains (OW40 and OW450).

Usage

```
data(wormsPE)
```

Format

A `SilacProteomicsExperiment` object with 250 proteins and 3574 peptides in a total of 14 samples.

colData A `DataFrame` with the design of the experiment: samples, timepoints, replicates...

assaysProt A list of matrices with quantification data at protein level: total intensity (int_total), light isotope intensity (int_light), heavy isotope intensity (int_heavy) and heavy/light isotope intensty ratio (ratio).

rowDataProt A `DataFrame` with 22 columns that contains general protein information: ids, gene names, molecular weight...

assaysPep A list of matrices with quantification data at peptide level: total intensity (int_total), light isotope intensity (int_light), heavy isotope intensity (int_heavy) and heavy/light isotope intensty ratio (ratio).

rowDataPept A `DataFrame` with 46 columns that contains general protein information: ids, amino acids counts, length...

linkerDf A `data.frame` with 3574 rows and 4 columns. It contains the relationships between proteins and peptides in the ProteomicsExperiment object.

Details

It is used as example in the pulsed silac vignette to illustrate the main data analysis functions and in the examples of the documentation.

References

https://www.ncbi.nlm.nih.gov/pubmed/28679685

Index