Package 'scp'

February 27, 2021

Title Mass Spectrometry-Based Single-Cell Proteomics Data Analysis

Version 1.0.0

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, S4Vectors, dplyr, magrittr, rlang

Suggests testthat, knitr, BiocStyle, rmarkdown, patchwork, ggplot2, matrixStats, impute, scater, sva, 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_12

git_last_commit 8a942fb

git_last_commit_date 2020-10-27

Date/Publication 2021-02-26

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>

R topics documented:

`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

```
aggregateFeaturesOverAssays(obj, i, fcol, name, fun, ...)
```

Arguments

`obj`	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

See Also

QFeatures::aggregateFeatures

Examples

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

`computeFDR` *Compute FDR from posterior error probabilities PEP*

Description

The functions takes the posterior error probabilities (PEPs) from the given assay's `rowData` and adds a new variable to it (called `.FDR`) that contains the computed false discovery rates (FDRs).

Usage

```
computeFDR(object, i, groupCol, pepCol)
```

Arguments

`object`	A `QFeatures` object
`i`	A `numeric()` or `character()` vector indicating from which assays the `rowData` should be taken.
`groupCol`	A `character(1)` indicating the variable names in the `rowData` that contains the grouping variable. The FDR are usually computed for PSMs grouped by peptide ID.
`pepCol`	A `character(1)` indicating the variable names in the `rowData` that contains the PEPs. Since, PEPs are probabilities, the variable must be contained in (0, 1).

Value

A `QFeatures` object.

Examples

```
data("scp1")
scp1 <- computeFDR(scp1,
                   i = 1,
                   groupCol = "Sequence",
                   pepCol = "dart_PEP")
## Check results
rowDataToDF(scp1, 1, c("dart_PEP", ".FDR"))
```

computeMedianCV *Compute the median coefficient of variation (CV) per cell*

Description

The function computes for each cell the median CV. The expression data is normalized twice. First, cell median expression is used as normalization factor, then, the mean for each batch and peptide. The CV is then computed for each protein in each cell. CV is the standard deviation divided by the mean expression. The CV is computed only if there are more than 5 observations per protein per cell.

Usage

```
computeMedianCV(object, i, peptideCol, proteinCol, batchCol)
```

Arguments

object	A QFeatures object
i	A numeric() or character() vector indicating from which assays the rowData should be taken.
peptideCol	A character(1) indicating the variable name in the rowData that contains the peptide grouping.
proteinCol	A character(1) indicating the variable name in the rowData that contains the protein grouping.
batchCol	A character(1) indicating the variable name in the colData of object that contains the batch names.

Details

A new columns, .medianCV, is added to the colData of the assay i and contains the computed median CVs.

Watch out that peptideCol and proteinCol are feature variables and hence taken from the rowData. batchCol is a sample variable and is taken from the colData of the QFeatures object.

Value

A QFeatures object.

Examples

```
data("scp1")
scp1 <- computeMedianCV(scp1,
                        i = "peptides",
                        proteinCol = "protein",
                        peptideCol = "peptide",
                        batchCol = "Set")
## Check results
hist(scp1[["peptides"]]$.MedianCV)
```

computeSCR *Compute the sample over carrier ratio (SCR)*

Description

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

Usage

```
computeSCR(obj, i, colDataCol, samplePattern, carrierPattern)
```

Arguments

obj	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(obj) 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

Value

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

Examples

```
data("scp1")
scp1 <- computeSCR(scp1,
                   i = 1,
                   colDataCol = "SampleType",
                   carrierPattern = "Carrier",
                   samplePattern = "Blank|Macrophage|Monocyte")
## 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(obj, i, colDataCol, samplePattern = ".", refPattern)
```

Arguments

obj	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(obj) that gives the sample annotation.
samplePattern	A character(1) pattern that matches the sample encoding in colDataCol. By default all samples are devided (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")
```

---

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

- 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
- 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
- Deamidation..N..site.IDs: a numeric 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..NQ..Probabilities: a logical vector
- Deamidation..NQ..Score.Diffs: a logical vector
- Deamidation..NQ.: 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
- 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 1197 rows and 139 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.

---

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(quantTable, metaTable, batchCol, channelCol, verbose = TRUE, ...)
```

Arguments

quantTable	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.
metaTable	A data.frame or any object that can be coerced to a data.frame. metaTable is expected to contains 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 quantTable and metaTable 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 metaTable that contains the column names of the quantitive data in quantTable (see Example).
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

```
## 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(quantTable = mqScpData,
        metaTable = sampleAnnotation,
        batchCol = "Set",
        channelCol = "Channel")
```

`readSingleCellExperiment` *Read SingleCellExperiment from tabular data*

Description

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

Usage

```
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 dislpay 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`

Author(s)

Laurent Gatto, Christophe Vanderaa

See Also

The code relies on QFeatures::readSummarizedExperiment.

Examples

```
## 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(obj, i, vars)
```

Arguments

`obj`	A `QFeatures` object
`i`	A `numeric()` or `character()` vector indicating from which assays the `rowData` should be taken.
`vars`	A `character()` vector indicating which variables from the `rowData` should be extracted.

Details

Along with the reuired `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

```
## 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

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

Format

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

Details

##' 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.

scp1 *Single Cell QFeatures data*

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 in 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
```

---

transferColDataToAssay

Transfer the colData *to an Assay*

Description

The function transfers the colData from a QFeatures object to one of the assays it contains. The transfered data is bound to the existing colData of the target assay.

Usage

```
transferColDataToAssay(obj, i)
```

Arguments

obj	A QFeatures object
i	A numeric(1) or character(1) indicating which assay to transfer the colData to.

Value

A QFeatures object

Examples

```
data("scp1")
colData(scp1[["peptides"]])
scp1 <- transferColDataToAssay(scp1, i = "peptides")
colData(scp1[["peptides"]])
```

Index