Package ‘SWATH2stats’

June 19, 2018

Type Package

Title Transform and Filter SWATH Data for Statistical Packages

Version 1.11.3

Date 2018-06-12

Author Peter Blattmann, Moritz Heusel and Ruedi Aebersold

Maintainer Peter Blattmann <blattmann@imsb.biol.ethz.ch>

Description This package is intended to transform SWATH data from the OpenSWATH software into a format readable by other statistics packages while performing filtering, annotation and FDR estimation.

License GPL-3

Depends R(>= 2.10.0)

Imports data.table, reshape2, ggplot2, stats, grDevices, graphics, utils, biomaRt

Suggests testthat, aLFQ, knitr, PECA

Enhances imsbInfer, MSstats

biocViews Proteomics, Annotation, ExperimentalDesign, Preprocessing, MassSpectrometry

NeedsCompilation no

VignetteBuilder knitr

RoxygenNote 6.0.1

R topics documented:

add_genesymbol 2
assess_decoy_rate 3
assess_fdr_byrun 4
assess_fdr_overall 5
base_size 7
convert_aLFQ 7
convert_mapDIA 8
convert_MSstats 9

convert_PECA . . . 10
convert_protein_ids . . . 11
convert_python . . . 12
count_analytes . . . 13
disaggregate . . . 14
filter_all_peptides . . . 15
filter_mscore . . . 16
filter_mscore_condition . . . 17
filter_mscore_fdr . . . 18
filter_mscore_freqobs . . . 19
filter_on_max_peptides . . . 20
filter_on_min_peptides . . . 21
filter_proteotypic_peptides . . . 22
import_data . . . 23
load_mart . . . 24
mscore4assayfdr . . . 25
mscore4pepfdr . . . 26
mscore4protfdr . . . 27
MSstats_data . . . 28
OpenSWATH_data . . . 29
plot_correlation_between_samples . . . 29
plot_fdr_cube . . . 30
plot_fdr_table . . . 31
plot_variation . . . 32
plot_variation_vs_total . . . 33
reduce_OpenSWATH_output . . . 34
removeDecoyProteins . . . 35
rmDecoyProt . . . 36
sample_annotation . . . 36
Spyogenes . . . 38
Study_design . . . 38
SWATH2stats . . . 39
transform_MSstats_OpenSWATH . . . 39
unifyProteinGroupLabels . . . 40
write_matrix_peptides . . . 41
write_matrix_proteins . . . 42

Index **43**

`add_genesymbol` *Gather gene symbols from biomart and add them to a swath2stats data set.*

Description

Gather gene symbols from biomart and add them to a swath2stats data set.

Usage

```
add_genesymbol(data_table, gene.ID.table, column.name = "protein",
  ID1 = "uniprotswissprot", ID2 = "hgnc_symbol", id.separator = "/",
  copy_nonconverted = TRUE)
```

Arguments

`data_table`	A data frame or file name.
`gene.ID.table`	A table to match against
`column.name`	The column name where the original protein identifiers are present.
`ID1`	The type of the original protein identifiers (e.g. "uniprotswissprot", "ensembl_peptide_id").
`ID2`	The type of the converted protein identifiers (e.g. "hgnc_symbol", "mgi_symbol", "external_gene_name").
`id.separator`	Separator between protein identifiers of shared peptides.
`copy_nonconverted`	Option defining if the identifiers that cannot be converted should be copied.

Value

some gene symbols

Note

Protein identifiers from shared peptides should be separated by a forward slash. The host of archived ensembl databases can be introduced as well (e.g. "dec2017.archive.ensembl.org")

Author(s)

Peter Blattmann

`assess_decoy_rate` *Query a swath2stats data set for decoy hit rates.*

Description

This looks at the swath2stats table and compares the length of the detected decoy peptides vs non-decoy peptides.

Usage

```
assess_decoy_rate(data)
```

Arguments

`data`	A data frame that contains at least a column named "FullPeptideName" and "decoy".

Details

A printout is generated to indicate the number of non-decoy, decoy peptides and the rate of decoy vs non-decoy peptides. Unique peptides are counted, so a precursor with different charge states is counted as one peptide. In the column "decoy" the values need to be 1,0 or TRUE and FALSE.

Value

list containing the number of decoys, non-decoys, and the ratio.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data <- OpenSWATH_data
 assess_decoy_rate(data)

## End(Not run)
```

`assess_fdr_byrun` *Assess assay, peptide and protein level FDR by run (for each MS_injection separately) in OpenSWATH output table*

Description

This function estimates the assay, peptide and protein FDR by run in an OpenSWATH result table in dependence of a range of m_score cutoffs. The results can be visualized and summarized by the associated method plot.fdr_table(). It counts target and decoy assays (unique transition_group_id), peptides (unique FullPeptideName) and proteins (unique ProteinName) in the OpenSWATH output table in dependence of m-score cutoff, the useful m_score cutoff range is evaluated for each dataset individually on the fly. To arrive from decoy counts at an estimation of the false discovery rate (false positives among the targets remaining at a given mscore cutoff) the ratio of false positives to true negatives (decoys) (FFT) must be supplied. It is estimated for each run individually by pyProphet and contained in the pyProphet statistics [Injection_name]_full_stat.csv. As an approximation, the FFTs of multiple runs are averaged and supplied as argument FFT. For further details see the Vignette Section 1.3 and 4.1. To assess fdr over the entire dataset, please refer to function assess_fdr_overall. FDR is calculated as FDR = (TN*FFT/T); TN=decoys, T=targets, FFT=see above.

Usage

```
assess_fdr_byrun(data, FFT = 1, n_range = 20, output = "pdf_csv",
  plot = TRUE, filename = "FDR_report_byrun",
  output_mscore_levels = c(0.01, 0.001))
```

Arguments

`data`	Annotated OpenSWATH/pyProphet output table. Refer to function sample_annotation from this package for further information.
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target).
`n_range`	Option to set the number of magnitude for which the m_score threshold is decreased (e.g. n.range = 10, m-score from 0.1 until 10^-10)^.
`output`	Choose output type. "pdf_csv" creates the output as files in the working directory, "Rconsole" triggers delivery of the output to the console enabling further computation or custom plotting / output.
`plot`	Logical, whether or not to create plots from the results (using the associated method plot.fdr_cube()
`filename`	Modify the basename of the result files if set.
`output_mscore_levels`	Define m-score levels to plot and write the estimated FDR results.

Value

Returns an array of target/decoy identification numbers and calculated FDR values at different m-score cutoffs.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 assess_fdr_byrun(data, FFT=0.7, output="pdf_csv", plot=TRUE,
                  filename="Testoutput_assess_fdr_byrun")

## End(Not run)
```

`assess_fdr_overall` *Assess overall FDR in annotated OpenSWATH/pyProphet output table in dependence of m_score cutoff*

Description

This function estimates the assay, peptide and protein FDR over a multi-run OpenSWATH/pyProphet output table. It counts target and decoy assays (unique transition_group_id), peptides (unique FullPeptideName) and proteins (unique ProteinName) in dependence of the m-score cutoff (1e-2 to 1e-20). To arrive from decoy counts at an estimation of the false discovery rate (false positives among the targets remaining at a given mscore cutoff) the ratio of false positives to true negatives (decoys) (FFT) must be supplied. It is estimated for each run individually by pyProphet and contained in the pyProphet statistics [Injection_name]_full_stat.csv. As an approximation, the FFTs of multiple runs are averaged and supplied as argument FFT. For further details see the Vignette Section 1.3 and 4.1. Protein FDR control on peak group quality level is a very strict filter and should be handled with caution. FDR is calculated as FDR = (TN*FFT/T); TN=decoys, T=targets, FFT=see above

Usage

```
assess_fdr_overall(data, FFT = 1, n_range = 20, output = "pdf_csv",
  plot = TRUE, filename = "FDR_report_overall")
```

Arguments

`data`	Data table that is produced by the OpenSWATH/pyProphet workflow
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target).
`n_range`	I am also not certain what this is, nor why 20 is the optimal default value, but I think the idea is to set up a series of mscore thresholds.
`output`	Choose output type. "pdf_csv" creates the output as files in the working directory, "Rconsole" triggers delivery of the output to the console enabling further computation or custom plotting / output.
`plot`	Logical, whether or not to create plots from the results (using the associated method plot.fdr_table()
`filename`	Optional, modifying the basename of the result files if applicable.

Value

Returns a list of class "fdr_table". If output "pdf_csv" and plot = TRUE were chosen, report files are written to the working folder.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 assess_fdr_overall(data, FFT=0.7, output="Rconsole", plot=TRUE,
                    filename="Testoutput_assess_fdr_overall")

## End(Not run)
```

---

base_size — *The following is used to set the ggplot2 default text size.*

Description

The following is used to set the ggplot2 default text size.

Usage

```
base_size
```

Format

An object of class `numeric` of length 1.

---

convert_aLFQ — *Convert table into the format for aLFQ*

Description

This functions selects the columns necessary for the aLFQ R package.

Usage

```
convert_aLFQ(data, annotation = TRUE, check_transitions = TRUE)
```

Arguments

`data`	A data frame containing the SWATH data in transition-level format
`annotation`	Option to indicate if the data has been annotated, i.e. if the columns Condition, Replicate, Run are present. If option is set to true it will write a new run_id as a string of the combination of these three columns.
`check_transitions`	if number of transitions should be checked. As input only transition-level data should be used and therefore this is checked. However, this makes the function slow and herewith be omitted.

Value

Returns a data frame in the appropriate format for aLFQ.

Author(s)

Peter Blattmann

References

Rosenberger G, Ludwig C, Rost HL, Aebersold R, Malmstrom L. aLFQ: an R-package for estimating absolute protein quantities from label-free LC-MS/MS proteomics data. Bioinformatics. 2014 Sep 1;30(17):2511-3. doi: 10.1093/bioinformatics/btu200.

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 raw <- disaggregate(data.filtered.decoy)
 data.aLFQ <- convert_aLFQ(raw)

## End(Not run)
```

`convert_mapDIA` *Convert table into the format for mapDIA*

Description

This functions selects the columns necessary for mapDIA.

Usage

```
convert_mapDIA(data, RT = FALSE)
```

Arguments

`data`	A data frame containing SWATH data.
`RT`	Option to export the retention times.

Value

Returns a data frame in the appropriate format for mapDIA.

Note

The table must not contain any technical replica, the intensity of technical replica is averaged. This function requires the package reshape2.

Author(s)

Peter Blattmann

References

Teo, G., et al. (2015). "mapDIA: Preprocessing and statistical analysis of quantitative proteomics data from data independent acquisition mass spectrometry." J Proteomics 129: 108-120.

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 raw <- disaggregate(data.filtered.decoy)
 data.mapDIA <- convert_mapDIA(raw, RT=TRUE)

## End(Not run)
```

`convert_MSstats` *Take data from the swath2stats format and get it ready for use by MSstats.*

Description

Though the two tools use very similar formats, some coercion is required to Convert table into the format for MSstats.

Usage

```
convert_MSstats(data, replace_values = TRUE, replace_colnames = TRUE,
  replace_unimod = TRUE)
```

Arguments

`data`	A data frame containing SWATH data.
`replace_values`	Option to indicate if negative and 0 values should be replaced with NA.
`replace_colnames`	Option to indicate if column names should be renamed and columns reduced to the necessary columns for MSstats.
`replace_unimod`	Option to indicate if Unimod Identifier should be replaced from ":" to "_".

Details

This functions selects the columns necessary for MSstats and renames them if necessary.

The necessary columns are selected and three columns renamed: FullPeptideName -> PeptideSequence Charge -> PrecursorCharge filename -> File

Value

Returns a data frame in the appropriate format for MSstats.

Author(s)

Peter Blattmann

References

Choi M, Chang CY, Clough T, Broudy D, Killeen T, MacLean B, Vitek O. MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments.Bioinformatics. 2014 Sep 1;30(17):2524-6. doi: 10.1093/bioinformatics/btu305.

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 raw <- disaggregate(data.filtered.decoy)
 data.mapDIA <- convert4MSstats(raw)

## End(Not run)
```

---

convert_PECA *Convert table into the format for ROPECA*

Description

This functions selects the columns necessary for ROPECA.

Usage

```
convert_PECA(data)
```

Arguments

data	A data frame containing SWATH data.

Value

Returns a data frame in the appropriate format for ROPECA.

Note

The table must not contain any technical replica, the intensity of technical replica is averaged. This function requires the package reshape2.

Author(s)

Peter Blattmann

References

Suomi, T. and Elo L.L. (2017). "Enhanced differential expression statistics for data-independent acquisition proteomics" Scientific Reports 7, Article number: 5869.doi:10.1038/s41598-017-05949-y

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 data.PECA <- convert_PECA(data.filtered.decoy)

## End(Not run)
```

---

convert_protein_ids *Convert protein ids*

Description

This function renames protein ids in a data frame or file

Usage

```
convert_protein_ids(data_table, column.name = "Protein",
  species = "hsapiens_gene_ensembl", host = "www.ensembl.org",
  mart = "ENSEMBL_MART_ENSEMBL", ID1 = "uniprotswissprot",
  ID2 = "hgnc_symbol", id.separator = "/", copy_nonconverted = TRUE,
  verbose = FALSE)
```

Arguments

`data_table`	A data frame or file name.
`column.name`	The column name where the original protein identifiers are present.
`species`	The species of the protein identifiers in the term used by biomaRt (e.g. "hsapiens_gene_ensembl", "mmusculus_gene_ensembl", "drerio_gene_ensembl", etc.)
`host`	Path of the biomaRt database (e.g. "www.ensembl.org", "dec2017.archive.ensembl.org").
`mart`	The type of mart (e.g. "ENSEMBL_MART_ENSEMBL", etc.)
`ID1`	The type of the original protein identifiers (e.g. "uniprotswissprot", "ensembl_peptide_id").
`ID2`	The type of the converted protein identifiers (e.g. "hgnc_symbol", "mgi_symbol", "external_gene_name").
`id.separator`	Separator between protein identifiers of shared peptides.
`copy_nonconverted`	Option defining if the identifiers that cannot be converted should be copied.
`verbose`	Option to write a file containing the version of the database used.

Value

The data frame with an added column of the converted protein identifiers.

Note

Protein identifiers from shared peptides should be separated by a forward slash. The host of archived ensembl databases can be introduced as well (e.g. "dec2017.archive.ensembl.org")

Author(s)

Peter Blattmann

Examples

```
 ## Not run:
  data_table <- data.frame(
       "Protein" = c("Q01581", "P49327", "2/P63261/P60709"),
       "Abundance" = c(100, 3390, 43423))
  convert_protein_ids(data_table)

## End(Not run)
```

---

convert_python *Convert data into the format for running a python script.*

Description

This functions selects the columns suggested to run a python script to change the data from peptide-level to transition-level.

Usage

```
convert_python(data, replace.Unimod = TRUE)
```

Arguments

`data`	A data frame containing SWATH data.
`replace.Unimod`	Option to indicate if Unimod Identifier should be replaced form ":"" to "_".

Details

The necessary columns are selected and the run column is renamed to filename for the script. The intensities are taken from the column aggr_Peak_Area and therefore the Intensity column is not exported.

Value

Returns a data frame in the appropriate format to be used by a custom python script stored in the scripts folder.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data,0.01)
 data.pythonscript <- convert4pythonscript(data.filtered.decoy)

## End(Not run)
```

`count_analytes` *Counts analytes in different injections*

Description

This functions counts the number of different peakgroups, peptides and proteins in different injections.

Usage

```
count_analytes(data, column_levels = c("transition_group_id",
  "fullpeptidename", "proteinname"), column_by = "run_id", rm_decoy = TRUE)
```

Arguments

`data`	A data frame containing SWATH data.
`column_levels`	Columns in which different identifiers should be counted.
`column_by`	Column for which the different identifiers should be counted for, e.g. for the different injections.
`rm_decoy`	Option to not remove decoy before counting.

Value

Returns a data frame with the count of the different identifiers per e.g. injection.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 count_analytes(data)

## End(Not run)
```

`disaggregate` *Transforms the SWATH data from a peptide- to a transition-level table.*

Description

If the SWATH data should be analyzed on transition-level the data needs to be tranformed from peptide-level table to a transition-level table (one row per transition instead of one row per peptide). The columns "aggr_Fragment_Annotation" and "aggr_Peak_Area" are disaggregated into the new columns "FragmentIon" and "Intensity". The following columns are renamed if they exist: FullPeptideName -> PeptideSequence, Charge -> PrecursorCharge, Area -> Intensity, Fragment -> FragmentIon, Sequence -> NakedSequence.

Usage

```
disaggregate(data, all.columns = FALSE)
```

Arguments

`data`	A data frame containing SWATH data.
`all.columns`	Option that all columns are processed. Otherwise only the typical columns needed for downstream analysis are processed.

Value

Returns a data frame containing the SWATH data in a transition-level table.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 raw <- disaggregate(data.filtered.decoy)

## End(Not run)
```

`filter_all_peptides` *Select all proteins that are supported by peptides.*

Description

This functions counts all proteins that are supported by peptides (including non proteo-typic peptides). All peptides (incl. non proteotypic peptides are selected. For the proteins supproted by proteotypic peptide the "1/" in front of the identifier is removed to facilitate further data processing.

Usage

```
filter_all_peptides(data, column = "proteinname", n = 6)
```

Arguments

`data`	A data frame containing SWATH data.
`column`	Which column contains the data to modify?
`n`	How many new IDs should we print to show if this worked?

Value

Returns a data frame with the data from both proteotypic and non-proteotypic peptides.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 data.all <- filter_all_peptides(data.filtered.decoy)

## End(Not run)
```

`filter_mscore` *Filter openSWATH output table according to mscore.*

Description

This function filters the SWATH data according to the m_score value, as well as to the number of occurence in the data (requant) and within a condition (condition).

Usage

```
filter_mscore(data, mscore = 1, rm.decoy = TRUE)
```

Arguments

`data`	A data frame containing SWATH data.
`mscore`	Value that defines the mscore threshold according to which the data will be filtered.
`rm.decoy`	Drop decoys from the data?

Value

A copy of the data which has been mscore-filtered.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- filter_mscore(data, 0.01)
 data.filtered <- filter_mscore_freqobs(data, 0.01, 0.8)
 data.filtered <- filter_mscore_condition(data, 0.01, 3)

## End(Not run)
```

`filter_mscore_condition` *Filter openSWATH output table according to mscore.*

Description

This function filters the SWATH data according to the m_score value, as well as to the number of occurence in the data (requant) and within a condition (condition).

Usage

```
filter_mscore_condition(data, mscore = 1, n.replica, rm.decoy = TRUE)
```

Arguments

`data`	A data frame containing SWATH data.
`mscore`	Value that defines the mscore threshold according to which the data will be filtered.
`n.replica`	Number of measurements within at least one condition that have to pass the mscore threshold for this transition.
`rm.decoy`	Drop decoys from the data?

Value

A copy of the data which has been mscore-filtered.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- filter_mscore(data, 0.01)
 data.filtered <- filter_mscore_freqobs(data, 0.01, 0.8)
 data.filtered <- filter_mscore_condition(data, 0.01, 3)

## End(Not run)
```

filter_mscore_fdr *Filter annotated OpenSWATH/pyProphet output table to achieve a high FDR quality data matrix with controlled overall protein FDR and quantitative values for all peptides mapping to these high-confidence proteins (up to a desired overall peptide level FDR quality).*

Description

This function controls the protein FDR over a multi-run OpenSWATH/pyProphet output table and filters all quantitative values to a desired overall/global peptide FDR level. It first finds a suitable m-score cutoff to minimally achieve a desired global FDR quality on a protein master list based on the function mscore4protfdr. It then finds a suitable m-score cutoff to minimally achieve a desired global FDR quality on peptide level based on the function mscore4pepfdr. Finally, it reports all the peptide quantities derived based on the peptide level cutoff for only those peptides mapping to the protein master list. It further summarizes the protein and peptide numbers remaining after the filtering and evaluates the individual run FDR qualities of the peptides (and quantitation events) selected.

Usage

```
filter_mscore_fdr(data, FFT = 1, overall_protein_fdr_target = 0.02,
  mscore_limit = 0.01, upper_overall_peptide_fdr_limit = 0.05,
  rm.decoy = TRUE)
```

Arguments

`data`	Annotated OpenSWATH/pyProphet data table.
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target). For further details see the Vignette Section 1.3 and 4.1.
`overall_protein_fdr_target`	FDR target for the protein master list for which quantitative values down to the less strict peptide_fdr criterion will be kept/reported. Defaults to 0.02.
`mscore_limit`	FDR target for the quantitative values kept/reported for all peptides mapping to the high-confidence protein master list. Defaults to 0.05. If all values up to m_score 0.01 shall be kept, set = 1.
`upper_overall_peptide_fdr_limit`	Option to relax or tighten the false discovery rate limit.
`rm.decoy`	Logical T/F, whether decoy entries should be removed after the analysis. Defaults to TRUE. Can be useful to disable to track the influence on decoy fraction by further filtering steps such as requiring 2 peptides per protein.

Value

Returns a data frame with the filtered data.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.fdr.filtered<-filter_mscore_fdr(data, FFT=0.7, overall_protein_fdr_target=0.02,
                                      upper_overall_peptide_fdr_limit=0.1)

## End(Not run)
```

`filter_mscore_freqobs` *Filter openSWATH output table according to mscore.*

Description

This function filters the SWATH data according to the m_score value, as well as to the number of occurence in the data (requant) and within a condition (condition)

Usage

```
filter_mscore_freqobs(data, mscore = 0.01, percentage = NULL,
  rm.decoy = TRUE, file_column = "filename")
```

Arguments

`data`	A data frame containing SWATH data.
`mscore`	Value that defines the mscore threshold according to which the data will be filtered.
`percentage`	Percentage in which replicas the transition has to reach the mscore threshold. ??
`rm.decoy`	Option to remove the decoys during filtering.
`file_column`	Column in the (raw)data in which the filenames should be found.

Value

Returns a data frame with the filtered data.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- filter_mscore(data, 0.01)
 data.filtered <- filter_mscore_freqobs(data, 0.01, 0.8)
 data.filtered <- filter_mscore_condition(data, 0.01, 3)

## End(Not run)
```

---

filter_on_max_peptides

Filter only for the highest intense peptides

Description

In order to reduce the data, the data is filtered only for the proteins with the highest intensity peptides.

Usage

```
filter_on_max_peptides(data, n_peptides = 6, rm.decoy = TRUE,
  column = "proteinname")
```

Arguments

`data`	A data frame containing SWATH data with the column names: ProteinNames, PeptideSequence, PrecursorCharge, Intensity.
`n_peptides`	Maximum number of highest intense peptides to filter the data on.
`rm.decoy`	Option to remove the decoys during filtering.
`column`	which column to use for filtering?

Value

Returns a data frame of the filtered data.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- filter_mscore_freqobs(data, 0.01,0.8)
 data.max <- filter_on_max_peptides(data.filtered, 5)

## End(Not run)
```

`filter_on_min_peptides` *Filter openSWATH output for proteins that are identified by a minimum of n independent peptides.*

Description

This function removes entries mapping to proteins that are identified by less than n_peptides. Removing single-hit proteins from an analysis can significantly increase the sensitivity under strict protein fdr criteria, as evaluated by e.g. assess_fdr_overall.

Usage

```
filter_on_min_peptides(data, n_peptides = 6, rm.decoy = TRUE,
  column = "proteinname")
```

Arguments

`data`	Data table that is produced by the openSWATH/iPortal workflow.
`n_peptides`	Number of minimal number of peptide IDs associated with a protein ID in order to be kept in the dataset.
`rm.decoy`	Option to remove the decoys during filtering.
`column`	which column to use for filtering?

Value

Returns the filtered data frame with only peptides that map to proteins with >= n_peptides peptides.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- filter_mscore_freqobs(data, 0.01,0.8)
 data.max <- filter_on_max_peptides(data.filtered, 5)
 data.min.max <- filter_on_min_peptides(data.max, 3)

## End(Not run)
```

---

filter_proteotypic_peptides

Filter for proteins that are supported by proteotypic peptides.

Description

Peptides can match to several proteins. With this function proteotypic peptides, peptides that are only contained in one protein are selected. Additionally the number of proteins are counted and printed.

Usage

```
filter_proteotypic_peptides(data, rm.decoy = TRUE, column = "proteinname")
```

Arguments

`data`	A data frame containing SWATH data.
`rm.decoy`	Option to remove the decoys during filtering.
`column`	Which column to query for filtering?

Value

Returns a data frame with only the data supported by proteotypic peptides.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 data.all <- filter_proteotypic_peptides(data.filtered.decoy)

## End(Not run)
```

`import_data` *Transforms the column names from a data frame to the required format.*

Description

This functions transforms the column names from a data frame from another format to a data frame with column names used by the OpenSWATH output and required for these functions. During executing of the function the corresponding columns for each column in the data need to be selected. For columns that do not corresond to a certain column 'not applicable' needs to be selected and the column names are not changed.

Usage

```
import_data(data)
```

Arguments

`data`	A data frame containing the SWATH-MS data (one line per peptide precursor quantified) but with different column names.

Value

Returns the data frame in the appropriate format.

Note

List of column names of the OpenSWATH data: ProteinName: Unique identifier for protein or proteingroup that the peptide maps to. Proteotypic peptides should be indicated by 1/ in order to be recognized as such by the function filter_proteotypic_peptides. FullPeptideName: Unique identifier for the peptide. Charge: Charge of the peptide precursor ion quantified. Sequence: Naked peptide sequence without modifications. aggr_Fragment_Annotation: aggregated annotation for the different Fragments quantified for this peptide. In the OpenSWATH results the different annotation in OpenSWATH are concatenated by a semicolon. aggr_Peak_Area: aggregated Intensity values for the different Fragments quantified for this peptide. In the OpenSWATH results the aggregated Peak Area intensities are concatenated by a semicolon. transition_group_id: A unique identifier for each transition group used. decoy: Indicating with 1 or 0 if this transition group is a decoy. m_score: Column containing the score that is used to estimate FDR or filter. M-score values of identified peak groups are equivalent to a q-value and thus typically are smaller than 0.01, depending on the confidence of identification (the lower the m-score, the higher the confidence). Column containing the score that is used to estimate FDR or filter. RT: Column containing the retention time of the quantified peak. filename: Column containing the filename or a unique identifier for each injection. Intensity: column containing the intensity value for each quantified peptide. Columns needed for FDR estimation and filtering functions: ProteinName, FullPeptideName, transition_group_id, decoy, m_score Columns needed for conversion to transition-level format (needed for MSStats and mapDIA input): aggr_Fragment_Annotation, aggr_Peak_Are

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data('Spyogenes', package = 'SWATH2stats')
 head(data)
 str(data)

## End(Not run)
```

`load_mart` *Gather some data from biomaRt. Convert protein ids*

Description

This function renames protein ids in a data frame or file.

Usage

```
load_mart(species, ensembl.path, mart, verbose = FALSE)
```

Arguments

species	The species of the protein identifiers in the term used by biomaRt (e.g. "hsapiens_gene_ensembl", "mmusculus_gene_ensembl", "drerio_gene_ensembl", etc.)
ensembl.path	ensembl host to query.
mart	The type of mart (e.g. "ENSEMBL_MART_ENSEMBL", etc.)
verbose	print a summary of the ensembl connection.

Value

ensembl connection for performing future queries.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data_table <- data.frame(Protein = c("Q01581", "P49327", "2/P63261/P60709"),
                          Abundance = c(100, 3390, 43423))
 convert_protein_ids(data_table)

## End(Not run)
```

mscore4assayfdr — *Find m_score cutoff to reach a desired FDR on assay level (over the entire OpenSWATH/pyProphet output table)*

Description

This function estimates the m_score cutoff required in a dataset to reach a given overall assay level FDR. It counts target and decoy assays at high resolution across the m_score cutoffs and reports a useful m_score cutoff - assay FDR pair close to the supplied fdr_target level over the entire dataset. The m_score cutoff is returned by the function and can be used in the context of the filtering functions, e.g.: data.assayFDR1pc<-filter_mscore(data, mscore4assayfdr(data, fdr_target=0.01)) To arrive from decoy counts at an estimation of the false discovery rate (false positives among the targets remaining at a given mscore cutoff) the ratio of false positives to true negatives (decoys) (FFT) must be supplied. It is estimated for each run individually by pyProphet and contained in the pyProphet statistics [Injection_name]_full_stat.csv. As an approximation, the FFTs of multiple runs are averaged and supplied as argument FFT. For further details see the Vignette Section 1.3 and 4.1. For FDR evaluations on peptide and protein level, please refer to functions mscore4pepfdr and mscore4protfdr.

Usage

```
mscore4assayfdr(data, FFT = 1, fdr_target = 0.01)
```

Arguments

`data`	Annotated OpenSWATH/pyProphet data table. See function sample_annotation from this package.
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target).
`fdr_target`	Assay FDR target, numeric, defaults to 0.01. An m_score cutoff achieving an FDR < fdr_target will be selected. Calculated as FDR = (TN*FFT/T); TN=decoys, T=targets, FFT=see above.

Value

Returns the m_score cutoff selected to arrive at the desired FDR

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 mscore4assayfdr(data, FFT=0.7, fdr_target=0.01)

## End(Not run)
```

---

mscore4pepfdr — *Find m_score cutoff to reach a desired FDR on peptide level (over the entire OpenSWATH/pyProphet output table)*

Description

This function estimates the m_score cutoff required in a dataset to reach a given overall peptide level FDR. It counts target and decoy peptides (unique FullPeptideName) at high resolution across the m_score cutoffs and reports a useful m_score cutoff - peptide FDR pair close to the supplied fdr_target level over the entire dataset. The m_score cutoff is returned by the function and can be used in the context of the filtering functions, e.g.: data.pepFDR2pc<-filter_mscore(data, mscore4pepfdr(data, fdr_target=0.02)) To arrive from decoy counts at an estimation of the false discovery rate (false positives among the targets remaining at a given mscore cutoff) the ratio of false positives to true negatives (decoys) (FFT) must be supplied. It is estimated for each run individually by pyProphet and contained in the pyProphet statistics [Injection_name]_full_stat.csv. As an approximation, the FFTs of multiple runs are averaged and supplied as argument FFT. For further details see the Vignette Section 1.3 and 4.1. For FDR evaluations on assay and protein level, please refer to functions mscore4assayfdr and mscore4protfdr

Usage

```
mscore4pepfdr(data, FFT = 1, fdr_target = 0.01)
```

Arguments

`data`	Annotated OpenSWATH/pyProphet data table. See function sample_annotation from this package.
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target).
`fdr_target`	FDR target, numeric, defaults to 0.01. An m_score cutoff achieving an FDR < fdr_target will be selected. Calculated as FDR = (TN*FFT/T); TN=decoys, T=targets, FFT=see above.

Value

Returns the m_score cutoff selected to arrive at the desired FDR

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 mscore4pepfdr(data, FFT=0.7, fdr_target=0.01)

## End(Not run)
```

`mscore4protfdr` *Find m_score cutoff to reach a desired FDR on protein level (over the entire OpenSWATH/pyProphet output table)*

Description

This function estimates the m_score cutoff required in a dataset to reach a given overall protein level FDR. This filter is to be used with caution as the resulting quantitative matrix is relatively sparse. It can be filled with quantitative values at a lower FDR quality level. It counts target and decoy peptides (unique ProteinName) at high resolution across the m_score cutoffs and reports a useful m_score cutoff - peptide FDR pair close to the supplied fdr_target level over the entire dataset. The m_score cutoff is returned by the function and can be used in the context of the filtering functions, e.g.: data.protFDR5pc<-filter_mscore(data, mscore4protfdr(data, fdr_target=0.02)) To arrive from decoy counts at an estimation of the false discovery rate (false positives among the targets remaining at a given mscore cutoff) the ratio of false positives to true negatives (decoys) (FFT) must be supplied. It is estimated for each run individually by pyProphet and contained in the pyProphet statistics [Injection_name]_full_stat.csv. As an approximation, the FFTs of multiple runs are averaged and supplied as argument FFT. For further details see the Vignette Section 1.3 and 4.1. For FDR evaluations on assay and peptide level, please refer to functions mscore4assayfdr and mscore4pepfdr.

Usage

```
mscore4protfdr(data, FFT = 1, fdr_target = 0.02)
```

Arguments

`data`	Annotated OpenSWATH/pyProphet data table. See function sample_annotation from this package.
`FFT`	Ratio of false positives to true negatives, q-values from [Injection_name]_full_stat.csv in pyProphet stats output. As an approximation, the q-values of multiple runs are averaged and supplied as argument FFT. Numeric from 0 to 1. Defaults to 1, the most conservative value (1 Decoy indicates 1 False target).

`fdr_target`	FDR target, numeric, defaults to 0.01. An m_score cutoff achieving an FDR < fdr_target will be selected. Calculated as FDR = (TN*FFT/T); TN=decoys, T=targets, FFT=see above.

Value

Returns the m_score cutoff selected to arrive at the desired FDR quality.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 mscore4protfdr(data, FFT=0.7, fdr_target=0.01)

## End(Not run)
```

`MSstats_data` *6 rows of sample MSstats data.*

Description

This provides 6 intensities of putative MSstats data for 3 replicates of HeLa cells.

Usage

```
data(MSstats_data)
```

Format

a data frame of 6 rows and 11 columns.

Examples

```
 data(MSstats_data)
 dim(MSstats_data)
```

`OpenSWATH_data` *Sample data from the OpenMS, pyprophet workflow.*

Description

I am not sure where this data came from, I need to look a little more closely.

Usage

```
data(OpenSWATH_data)
```

Format

a data frame of 2369 rows and 58 columns.

Examples

```
data(OpenSWATH_data)
dim(OpenSWATH_data)
```

`plot_correlation_between_samples` *Plots the correlation between injections.*

Description

This function plots the Pearson's and Spearman correlation between samples. If decoys are present these are removed before plotting.

Usage

```
plot_correlation_between_samples(data, column.values = "intensity",
  size = 6, comparison = transition_group_id ~ condition + bioreplicate,
  fun.aggregate = NULL, label = TRUE, ...)
```

Arguments

`data`	Data frame that is produced by the OpenSWATH/pyProphet workflow.
`column.values`	Indicates the columns for which the correlation is assessed. This can be the Intensity or Signal, but also the retention time.
`size`	How large should the text in the grid be (smaller is better for rmarkdown html reports).
`comparison`	The comparison for assessing the variability. Default is to assess the variability per transition_group_id over the different Condition and Replicates. Comparison is performed using the dcast() function of the reshape2 package.

`fun.aggregate`	If for the comparison values have to be aggregated one needs to provide the function here.
`label`	Option to print correlation value in the plot.
`...`	Further arguments passed to methods.

Value

Plots in Rconsole a correlation heatmap and returns the data frame used to do the plotting.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 plot_correlation_between_samples(data)

## End(Not run)
```

---

plot_fdr_cube *Plot functionality for FDR assessment result arrays as produced by e.g. the function assess_fdr_byrun()*

Description

This function creates standard plots from result arrays as produced by e.g. the function assess_fdr_byrun(), visualizig assay, peptide and protein level FDR for each run at m-score cutoffs 1e-2 and 1e-3. Furthermore, Target and Decoy ID numbers are visualized.

Usage

```
plot_fdr_cube(x, output = "Rconsole", filename = "FDR_report_byrun",
  plot_mscore_levels = c(0.01, 0.001), ...)
```

Arguments

`x`	Array of by-run FDR assessment results as produced e.g. by the function assess_fdr_byrun() from this package.
`output`	Choose output type. "pdf_csv" creates the output as files in the working directory, "Rconsole" triggers delivery of the output to the console enabling further computation and/or custom plotting / output.
`filename`	Basename for output files to be created (if output = "pdf_csv" has been selected).
`plot_mscore_levels`	Define m-score levels to plot the estimated FDR results.
`...`	Extra arguments passed on to functions inside this.

Value

Originally this returned nothing, but now it makes a list of ggplot2 plots which may be passed along and plotted as desired (with that in mind, I would like to remove the explicit plot() calls in this function).

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 x <- assess_fdr_byrun(data, FFT=0.7, output="Rconsole", plot=FALSE)
 plot.fdr_cube(x, output="pdf_csv", filename="Assess_fdr_byrun_testplot",
               plot_mscore_levels=0.01)

## End(Not run)
```

---

plot_fdr_table *Plot functionality for results of class "fdr_table" as produced by e.g. the function assess_fdr_overall()*

Description

This function created standard plots from results of class "fdr_table" as produced by e.g. the function assess_fdr_overall() visualizig ID numbers in dependence of estimated FDR and also estimated FDR in dependence of m_score cutoff.

Usage

```
plot_fdr_table(x, output = "Rconsole", filename = "FDR_report_overall", ...)
```

Arguments

`x`	List of class "fdr_table" as produced e.g. by the function assess_fdr_overall() from this package.
`output`	Choose output type. "pdf_csv" creates the output as files in the working directory, "Rconsole" triggers delivery of the output to the console enabling further computation or custom plotting / output.
`filename`	Basename for output files to be created (if output = "pdf_csv" has been selected).
`...`	Extra arguments passed on to functions inside this.

Value

Originally this returned nothing, but now it makes a list of ggplot2 plots which may be passed along and plotted as desired (with that in mind, I would like to remove the explicit plot() calls in this function).

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 x <- assess_fdr_overall(data, FFT=0.7, output="Rconsole", plot=FALSE)
 plot.fdr_table(x, output="pdf_csv", filename="Assess_fdr_overall_testplot")

## End(Not run)
```

---

plot_variation *Plots the coefficient of variation for different replicates.*

Description

This function plots the coefficient of variation within replicates for a given value. If decoys are present these are removed before plotting.

Usage

```
plot_variation(data, column.values = "intensity",
  comparison = transition_group_id + condition ~ bioreplicate,
  fun.aggregate = NULL, label = TRUE, ...)
```

Arguments

`data`	Data frame that is produced by the OpenSWATH/pyProphet workflow.
`column.values`	Indicates the columns for which the variation is assessed. This can be the Intensity or Signal, but also the retention time.
`comparison`	The comparison for assessing the variability. Default is to assess the variability per transition_group_id and Condition over the different Replicates. Comparison is performed using the dcast() function of the reshape2 package.
`fun.aggregate`	If for the comparison values have to be aggregated one needs to provide the function here.
`label`	Option to print value of median cv.
`...`	further arguments passed to method.

Value

Returns a list with the data and calculated cv and a table that summarizes the mean, median and mode cv per Condition (if Condition is contained in the comparison). In addition it plots in Rconsole a violin plot with the observed coefficient of variations.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 plot_variation(data)

## End(Not run)
```

---

plot_variation_vs_total

Plots the total variation versus variation within replicates

Description

This function plots the total variation and the variation within replicates for a given value. If decoys are present these are removed before plotting.

Usage

```
plot_variation_vs_total(data, column.values = "intensity",
  comparison1 = transition_group_id ~ bioreplicate + condition,
  comparison2 = transition_group_id + condition ~ bioreplicate,
  fun.aggregate = NULL, label = TRUE, ...)
```

Arguments

`data`	Data table that is produced by the OpenSWATH/pyProphet workflow.
`column.values`	Indicates the columns for which the variation is assessed. This can be the Intensity or Signal, but also the retention time.
`comparison1`	The comparison for assessing the total variability. Default is to assess the variability per transition_group_id over the combination of Replicates and different Conditions.
`comparison2`	The comparison for assessing the variability within the replicates. Default is to assess the variability per transition_group_id and Condition over the different Replicates.

fun.aggregate	If depending on the comparison values have to be aggregated one needs to provide the function here. (I think this should be sum, yesno?)
label	Option to print value of median cv.
...	Arguments passed through, currently unused.

Value

Plots in Rconsole a violin plot comparing the total variation with the variation within replicates. In addition it returns the data frame from which the plotting is done and a table with the calculated mean, median and mode of the cv for the total or replicate data.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 plot_variation_vs_total(data)

## End(Not run)
```

---

reduce_OpenSWATH_output *Reduce columns of OpenSWATH data*

Description

This function selects the columns from the standard OpenSWATH output to column needed for MSstats, aLFQ and mapDIA.

Usage

```
reduce_OpenSWATH_output(data, column.names = NULL,
  data_file_column = "filename")
```

Arguments

data	A data frame containing SWATH data.
column.names	A vector of column names that can be selected.
data_file_column	Name of the column containing the files from the (raw)data.

Value

Returns a data frame with the selected columns.

Note

A basic set of columns are defined in the function and are used if no column names are indicated.

The column.names can be omitted and then the following columns are selected that are needed for MSstats and mapDIA analysis: ProteinName, FullPeptideName, Sequence, Charge, aggr_Fragment_Annotation, aggr_Peak_Area, filename, m_score, decoy, Intensity, RT. This function should be ommitted if the data is analyzed afterwards with the aLFQ or imsbInfer package that needs further columns.

Author(s)

Peter Blattmann

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered <- reduce_OpenSWATH_output(data)

## End(Not run)
```

`removeDecoyProteins` *Removes decoy proteins from the protein group label*

Description

There exist peptides annotated as protein groups with 2/ProteinA/DECOY_ProteinB. However these are in principal proteotypic peptides and should be annoated 1/ProteinA. This function changes these labels accordingly. The subfunction rmDecoyProt removes the Decoy protein, calling removeDecoyProteins also changes the nubmer before the protein group accordingly.

Usage

```
removeDecoyProteins(data, column = "proteinname")
```

Arguments

`data`	A data frame containing SWATH data.
`column`	which column to query?

Value

Returns a data frame with changed protein labels

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 data.2 <- removeDecoyProteins(data.filtered.decoy)

## End(Not run)
```

`rmDecoyProt` *What string defines decoys?*

Description

What string defines decoys?

Usage

```
rmDecoyProt(x, pattern = "DECOY")
```

Arguments

`x`	proteinname string to query.
`pattern`	chosen string to seek out.

`sample_annotation` *Annotate the SWATH data with the sample information*

Description

For statistical analysis and filtering the measurements need to be annotated with Filename, Condition, BioReplicate, and Run. This functions takes this information from a txt file containing this meta-data.

Usage

```
sample_annotation(data, sample_annotation, data_type = "OpenSWATH",
  annotation_file_column = "filename", check_files = TRUE,
  data_file_column = "filename", condition_column = "condition",
  replicate_column = "bioreplicate",
  fullpeptidename_column = c("fullpeptidename", "fullunimodpeptidename"),
  run_column = "run", change_run_id = TRUE, verbose = FALSE)
```

Arguments

`data`	A data frame containing SWATH data.
`sample_annotation`	A data frame containing the columns: Filename, Condition, BioReplicate, Run. The values contained in the column filename have to be present in the filename of the SWATH data.
`data_type`	Option to specify the format of the table, if the column names from an OpenSWATH output or MSstats table are used.
`annotation_file_column`	Name of the column containing the output file from the experiment metadata for each sample. In my own sample sheet, I keep columns for the mzXML files, tric outputs, raw files, and osw outputs from OpenSwathWorkFlow; and I cannot be relied upon to remember which is which, ergo this option.
`check_files`	Boolean checking if one wishes to ensure that the files listed in the annotation data are equivalent to the files in the actual data.
`data_file_column`	Option to specify the column name where the injection file is specified. Default is set to "filename".
`condition_column`	Which column annotates the experimental condition in the swath data?
`replicate_column`	Which column annotates the replicate in the data?
`fullpeptidename_column`	Character list of possible column names used to define the full peptide name.
`run_column`	Which column annotates the separate runs?
`change_run_id`	Option to choose if the run_id column shall be reassigned to a unique value combining the values of Condition, BioReplicate and Run. (Option only possible if data is of format "OpenSWATH")
`verbose`	Option to turn on reporting on which filename it is working on.

Details

Given dataframes of TRIC processed data and sample annotations, mash them together into something appropriate for downstream analyses.

This performs some quick sanity checks on the data and annotations and creates the 'Condition', 'BioReplicate', and 'Run' columns along with other columns expected by MSstats/OpenSWATH.

Value

Returns a dataframe with each row annotated for the study design

Author(s)

Peter Blattman

Examples

```
data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- SWATH2stats::sample_annotation(OpenSWATH_data, Study_design, verbose=TRUE)
summary(data)
```

`Spyogenes` *A set of Streptococcus pyogenes intensities from 4 samples.*

Description

To my eyes, this appears to be the result of OpenMS->pyprophet with the exception that the alignment files look to be gzipped Excel files. I bet they are actually tab separated outputs from feature_alignment.py or pyprophet.

Usage

```
data(Spyogenes)
```

Format

a data frame of 32,272 rows and 13 columns.

Examples

```
data(Spyogenes)
data <- Spyogenes
dim(data)
```

`Study_design` *A set of sample annotations which describe an experiment 6 of HeLa control and treated samples (3 of each).*

Description

This provides Filename, Condition, BioReplicate, and Run data for a small example dataset. I assumed at first that these correspond to the data in MSstats_data, but that cannot be, as they have different Run IDs.

Usage

```
data(Study_design)
```

Format

a data frame of 6 rows and 4 columns.

Examples

```
data(Study_design)
dim(Study_design)
```

`SWATH2stats` *SWATH2stats: a suite of tools to filter, plot, and convert DIA data.*

Description

SWATH2stats: a suite of tools to filter, plot, and convert DIA data.

`transform_MSstats_OpenSWATH` *Transforms column names to OpenSWATH column names*

Description

This functions transforms the column names from a data frame in MSstats format to a data frame with column names used by the OpenSWATH output. The original table needs to contain at least the 10 columns defined by MSstats: ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity.)

Usage

```
transform_MSstats_OpenSWATH(data)
```

Arguments

`data`	A data frame containing the SWATH data in the MSstats format

Value

The data frame in the appropriate format.

Author(s)

Peter Blattmann

References

Choi M, Chang CY, Clough T, Broudy D, Killeen T, MacLean B, Vitek O. MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments.Bioinformatics. 2014 Sep 1;30(17):2524-6. doi: 10.1093/bioinformatics/btu305.

Examples

```
## Not run:
 data("MSstats_data", package="SWATH2stats")
 transform_MSstats_OpenSWATH(MSstats_data)

## End(Not run)
```

---

unifyProteinGroupLabels

Unify the protein group labels.

Description

Unify the protein group labels (2/ProteinA/ProteinB and 2/ProteinB/ProteinA) to one common label (e.g. 2/ProteinA/ProteinB)

Usage

```
unifyProteinGroupLabels(data, column = "proteinname", keep_colnames = FALSE)
```

Arguments

`data`	A data frame containing SWATH data.
`column`	Which column to use for unifying the groups.
`keep_colnames`	Return the column names to their case-sensitive precursors.

Value

Returns a data frame with the unififed protein labels.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 data.filtered.decoy <- filter_mscore(data, 0.01)
 data.unified <- unifyProteinGroupLabels(data.filtered.decoy)

## End(Not run)
```

`write_matrix_peptides` *Writes out an overview matrix of peptides mapping to a FDR quality controlled protein master list at controlled global peptide FDR quality.*

Description

Writes out an overview matrix on peptide level of a supplied (unfiltered or prefiltered) OpenSWATH results data frame. The peptide quantification is achieved by summing the areas under all 6 transitions per precursor and summing all precursors per FullPeptideName. In order to keep the peptide-to-protein association, the FullPeptideName is joined with the ProteinName.

Usage

```
write_matrix_peptides(data, write.csv = FALSE, fun.aggregate = sum,
  filename = "SWATH2stats_overview_matrix_peptidelevel.csv",
  rm.decoy = FALSE)
```

Arguments

`data`	A data frame containing annotated OpenSWATH/pyProphet data.
`write.csv`	Option to determine if table should be written automatically into csv file.
`fun.aggregate`	What function to use when aggregating the set of intensities?
`filename`	File base name of the .csv matrix written out to the working folder.
`rm.decoy`	Logical whether decoys will be removed from the data matrix. Defaults to FALSE. It's sometimes useful to know how decoys behave across a dataset and how many you allow into your final table with the current filtering strategy.

Value

the peptides as a matrix! also output .csv matrix is written to the working folder.

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 write_matrix_peptides(data)

## End(Not run)
```

write_matrix_proteins *Writes out an overview matrix of summed signals per protein identifier (lines) over run_id(columns).*

Description

Writes out an overview matrix on protein level of a supplied (unfiltered or filtered) OpenSWATH results data frame. The protein quantification is achieved by summing the areas under all 6 transitions per precursor, summing all precursors per FullPeptideName and all FullPeptideName signals per ProteinName entry. This function does not select consistently quantified or top peptides but sums all signals availabe that may or may not originate from the same set of peptides across different runs. A more detailed overview can be generated using the function write_matrix_peptides(). Peptide selection can be achieved upstream using e.g. the functions filter_mscore_requant(), filter_on_max_peptides() and filter_on_min_peptides().

Usage

```
write_matrix_proteins(data, write.csv = FALSE, fun.aggregate = sum,
  filename = "SWATH2stats_overview_matrix_proteinlevel.csv",
  rm.decoy = FALSE)
```

Arguments

`data`	A data frame containing annotated OpenSWATH/pyProphet data.
`write.csv`	Option to determine if table should be written automatically into csv file.
`fun.aggregate`	What function to use when aggregating the set of intensities?
`filename`	File base name of the .csv matrix written out to the working folder
`rm.decoy`	Logical whether decoys will be removed from the data matrix. Defaults to FALSE. It's sometimes useful to know how decoys behave across a dataset and how many you allow into your final table with the current filtering strategy.

Value

the peptides as a matrix, also output .csv matrix is written to the working folder

Author(s)

Moritz Heusel

Examples

```
## Not run:
 data("OpenSWATH_data", package="SWATH2stats")
 data("Study_design", package="SWATH2stats")
 data <- sample_annotation(OpenSWATH_data, Study_design)
 write_matrix_proteins(data)

## End(Not run)
```

Index

∗Topic **datasets**
    base_size, 7
    MSstats_data, 28
    OpenSWATH_data, 29
    Spyogenes, 38
    Study_design, 38

add_genesymbol, 2
assess_decoy_rate, 3
assess_fdr_byrun, 4
assess_fdr_overall, 5

base_size, 7

convert_aLFQ, 7
convert_mapDIA, 8
convert_MSstats, 9
convert_PECA, 10
convert_protein_ids, 11
convert_python, 12
count_analytes, 13

disaggregate, 14

filter_all_peptides, 15
filter_mscore, 16
filter_mscore_condition, 17
filter_mscore_fdr, 18
filter_mscore_freqobs, 19
filter_on_max_peptides, 20
filter_on_min_peptides, 21
filter_proteotypic_peptides, 22

import_data, 23

load_mart, 24

mscore4assayfdr, 25
mscore4pepfdr, 26
mscore4protfdr, 27
MSstats_data, 28

OpenSWATH_data, 29

plot_correlation_between_samples, 29
plot_fdr_cube, 30
plot_fdr_table, 31
plot_variation, 32
plot_variation_vs_total, 33

reduce_OpenSWATH_output, 34
removeDecoyProteins, 35
rmDecoyProt, 36

sample_annotation, 36
Spyogenes, 38
Study_design, 38
SWATH2stats, 39
SWATH2stats-package *(SWATH2stats)*, 39

transform_MSstats_OpenSWATH, 39

unifyProteinGroupLabels, 40

write_matrix_peptides, 41
write_matrix_proteins, 42