Package ‘SWATH2stats’

November 6, 2016

Type Package
Title Transform and Filter SWATH Data for Statistical Packages
Version 1.5.1
Date 2016-10-19
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, grid, ggplot2, stats, grDevices,
graphics, utils
Suggests testthat, aLFQ, knitr
Enhances imsbInfer, MSstats
biocViews Proteomics, Annotation, ExperimentalDesign, Preprocessing,
MassSpectrometry
NeedsCompilation no
VignetteBuilder knitr

R topics documented:

  SWATH2stats-package ..............................................  2
  assess_decoy_rate ..............................................  3
  assess_fdr_byrun .................................................  4
  assess_fdr_overall ..............................................  5
  convert4aLFQ ......................................................  6
  convert4mapDIA ....................................................  7
  convert4MSstats ...................................................  8
  convert4pythonscript ............................................  9
  count_analytes ................................................... 10
  disaggregate ...................................................... 10
  filter_all_peptides ............................................. 11
  filter_mscore ..................................................... 12
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 assessment.

Details

Package: SWATH2stats
Type: Package
Version: 1.3.10
Date: 2016-08-05
License: GPLv3

Author(s)

Peter Blattmann, Moritz Heusel and Ruedi Aebersold

Maintainer: Peter Blattmann <blattmann@imsb.biol.ETHZ.ch>
assess_decoy_rate

References


See Also

aLFQ, MSstats.

assess_decoy_rate

assess_decoy_rate: Assess decoy rate

Description

This function counts the number of 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

Prints the decoy rate.

Author(s)

Peter Blattmann
assess_fdr_byrun

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

Usage

assess_fdr_byrun(data, FFT, n.range = 20, output = "pdf_csv", plot = TRUE, filename = "FDR_report_byrun")

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 Optional, modifying the basename of the result files if applicable.
assess_fdr_overall

Value

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

Author(s)

Moritz Heusel

Examples

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")

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 $FDR = \frac{TN \times FFT}{T}$; TN=decoys, T=target, FFT=see above

Usage

assess_fdr_overall(data, FFT, n.range = 20, output = "pdf_csv", plot = TRUE,
filename="FDR_report_overall")

Arguments

data Data table that is produced by the OpenSWATH/pyProphet workflow

n.range Option to set the number of magnitude for which the m_score threshold is de-
creased (e.g. n.range = 10, m-score from 0.1 until 10^-10).

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).
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

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")

convert4aLFQ

convert4aLFQ: Convert table into the format for aLFQ

Description

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

Usage

convert4aLFQ(data, annotation = 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.

Value

Returns a data frame in the appropriate format for aLFQ.

Author(s)

Peter Blattmann
**References**


**Examples**

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 <- convert4aLFQ(raw)

---

**convert4mapDIA**

*convert4mapDIA: Convert table into the format for mapDIA*

**Description**

This function selects the columns necessary for mapDIA.

**Usage**

`convert4mapDIA(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**

# convert4MSstats

Convert table into the format for MSstats

## Description

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

## Usage

```r
convert4MSstats(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

The necessary columns are selected and three columns renamed: `FullPeptideName` -> `PeptideSequence`, `Charge` -> `PrecursorCharge`, `align_origfilename` -> `File`.

## Value

Returns a data frame in the appropriate format for MSstats.

## Author(s)

Peter Blattmann

## References

\textit{convert4pythonscript}

\textbf{Examples}

\begin{verbatim}
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{verbatim}

\textit{convert4pythonscript} \hspace{1cm} \textit{convert4bashscript}: Convert data into the format for running a bash script

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
convert4pythonscript(data, replace.Unimod = TRUE)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{data} \hspace{1cm} A data frame containing SWATH data.
  \item \texttt{replace.Unimod} \hspace{1cm} Option to indicate if Unimod Identifier should be replaced form ":" to ":".
\end{itemize}

\textbf{Details}

The necessary columns are selected and the run column is renamed to align_origfilename for the script. The intensities are taken from the column \texttt{aggr_Peak_Area} and therefore the Intensity column is not exported.

\textbf{Value}

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

\textbf{Author(s)}

Peter Blattmann

\textbf{Examples}

\begin{verbatim}
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{verbatim}
**count_analytes**  
*count_analytes: Counts analytes in different injections*

**Description**  
This function counts the number of different peakgroups, peptides and proteins in different injections.

**Usage**  
```r  
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**  
```r  
data("OpenSWATH_data", package="SWATH2stats")  
data("Study_design", package="SWATH2stats")  
data <- sample_annotation(OpenSWATH_data, Study_design)  
count_analytes(data)  
```

---

**disaggregate**  
*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 transformed from peptide-level table to a transition-level table (one row per transition instead of one row per peptide). The columns "aggr_Fragment_Annnotation" and "aggr_Peak_Area" are disaggregated into the new columns "FragmentIon" and "Intensity".

**Usage**  
```r  
disaggregate(data)  
```
filter_all_peptides

Arguments

data A data frame containing SWATH data.

Value

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

Author(s)

Peter Blattmann

Examples

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.all <- filter_all_peptides(data.filtered.decoy)

filter_all_peptides Select all proteins that are supported by peptides.

Description

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

Usage

filter_all_peptides(data)

Arguments

data A data frame containing SWATH data.

Value

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

Author(s)

Peter Blattmann

Examples

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)
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 occurrence in the data (requant) and within a condition (condition).

Usage

filter_mscore(data, mscore, rm.decoy=TRUE)
filter_mscore_freqobs(data, mscore, percentage=NULL, rm.decoy = TRUE)
filter_mscore_condition(data, mscore, 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.
percentage Percentage in which replicas the transition has to reach the mscore threshold
rm.decoy Option to remove the decoys during filtering.

Value

Returns a data frame with the filtered data.

Author(s)

Peter Blattmann

Examples

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)
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. It further evaluates the individual run FDR qualities of the peptides (and quantitation events) selected.

Usage

```r
filter_mscore_fdr(data, FFT = 1, overall_protein_fdr_target = 0.02, 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.
- `upper_overall_peptide_fdr_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.
- `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

- `data.filtered`: the filtered data frame

Author(s)

Moritz Heusel
Examples

```r
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)
data.max <- filter_on_max_peptides(data.fdr.filtered, 5)
```

---

**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**

```r
filter_on_max_peptides(data, n_peptides)
```

**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.

**Value**

Returns a data frame of the filtered data

**Author(s)**

Peter Blattmann

**Examples**

```r
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)
```
**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)
```

**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.

**Value**

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

**Author(s)**

Moritz Heusel

**Examples**

```
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)
```

---

**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.
import_data

Usage

filter_proteotypic_peptides(data)

Arguments

data A data frame containing SWATH data.

Value

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

Author(s)

Peter Blattmann

Examples

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)

import_data

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

Description

This function 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 correspond 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.

align_origfilename: 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_Area

Author(s)

Peter Blattmann

Examples

data('Spyogenes', package = 'SWATH2stats')
head(data)
str(data)

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 m_score 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 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

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)
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 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

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)
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 m_score 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 mscore4pepfdr

Usage

mscore4protfdr(data, FFT, fdr_target)

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

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)

Description

A small table with the column names corresponding to the MSstats format. This data is intended only to test functions.

Author(s)

Peter Blattmann

OpenSWATH_data  Testing dataset from OpenSWATH

Description

A small selection of the data obtained from the iPortal pipeline for an experiment with perturbations relating to cholesterol regulation. Protein and Peptides have been anonymized as the data is unpublished. The FDR version of the test data contains modified (lowered) decoy peak group m_scores to simulate FDR behaviour of a large dataset.

Author(s)

Peter Blattmann

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(), visualizing 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

## S3 method for class 'fdr_cube'
plot(x, output = "Rconsole", filename = "FDR_report_byrun", ...)
### 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). Further arguments passed to method.

### Value

Plots in Rconsole or report files.

### Author(s)

Moritz Heusel

### Examples

```r
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")
```

---

### 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

```r
## S3 method for class 'fdr_table'
plot(x, output = "Rconsole", filename = "FDR_report_overall", ...)"n
```n

### 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). Further arguments passed to method.
plot_correlation_between_samples

Value

Plots in Rconsole or report files.

Author(s)

Moritz Heusel

Examples

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")

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",
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.

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 method.

Value

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

Author(s)

Peter Blattmann
plot_variation

Examples

data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
plot_correlation_between_samples(data)

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

data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
plot_variation(data)
**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**

```r
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.
- `label`: Option to print value of median cv.
- `...`: Further arguments passed to method.

**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**

```r
data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
plot_variation_vs_total(data)
```
**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)
```

**Arguments**

- `data` A data frame containing SWATH data.
- `column.names` A vector of column names that can be selected.

**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.

**Note**

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`, `align_origfilename`, `m_score`, `decoy`, `Intensity`, `RT`. This function should be omitted if the data is analyzed afterwards with the aLFQ or imsbInfer package that needs further columns.

**Author(s)**

Peter Blattmann

**Examples**

```r
data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
data.filtered <- reduce_OpenSWATH_output(data)
```
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

```r
sample_annotation(data, sample.annotation, data.type="OpenSWATH", column.file = "align_origfilename", 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.
- `column.file`: Option to specify the column name where the injection file is specified. Default is set to "align_origfilename".
- `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.

Value

Returns a dataframe with each row annotated for the study design

Author(s)

Peter Blattmann

Examples

```r
data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
```
Spyogens  |  S.pyogenes example data

**Description**

A table containing SWATH-MS data from S.pyogenes

**Source**  This table was generated from the original data deposited on PeptideAtlas (PASS00289, file "rawOpenSwathResults_lpcnt_only.tsv") by selecting only the column necessary for the SWATH2stats.


---

**Study design**  |  Study design table

**Description**

A table containing the meta-data defining the study design.

- **Filename**  A unique identifier corresponding to the filename in the SWATH data.
- **Condition**  The Condition explains the perturbation performed on this sample.
- **BioReplicate**  Number indicating the biological replicate of this sample.
- **Run**  A unique number for each MS-injection.

**Author(s)**

Peter Blattmann

**Source**

Peter Blattmann

---

**transform_MSstats_OpenSWATH**

**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.)
write_matrix_peptides

Usage

transform_MSstats_OpenSWATH(data)

Arguments

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

Value

Returns the data frame in the appropriate format.

Author(s)

Peter Blattmann

References


Examples

data("MSstats_data", package="SWATH2stats")
transform_MSstats_OpenSWATH(MSstats_data)

write_matrix_peptides

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,
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.
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.
write_matrix_proteins

Value

No return value, output .csv matrix is written to the working folder.

Author(s)

Moritz Heusel

Examples

data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
write_matrix_peptides(data)

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 available 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, 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.
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

No return value, output .csv matrix is written to the working folder.
Author(s)
Moritz Heusel

Examples

data("OpenSWATH_data", package="SWATH2stats")
data("Study_design", package="SWATH2stats")
data <- sample_annotation(OpenSWATH_data, Study_design)
write_matrix_proteins(data)
Index

*Topic SWATH2stats
   SWATH2stats-package, 2

aLFQ, 3
assess_decoy_rate, 3
assess_fdr_byrun, 4
assess_fdr_overall, 5
convert4aLFQ, 6
convert4mapDIA, 7
convert4MSstats, 8
convert4pythonscript, 9
count_analytics, 10
disaggregate, 10
filter_all_peptides, 11
filter_mscore, 12
filter_mscore_condition
   (filter_mscore), 12
filter_mscore_fdr, 13
filter_mscore_freqobs (filter_mscore), 12
filter_on_max_peptides, 14
filter_on_min_peptides, 15
filter_proteotypic_peptides, 15
import_data, 16
mscore4assayfdr, 17
mscore4pepfdr, 19
mscore4protfdr, 20
MSstats, 3
MSstats_data, 21
OpenSWATH_data, 21
OpenSWATH_data_FDR (OpenSWATH_data), 21
plot.fdr_cube, 21
plot.fdr_table, 22
plot_correlation_between_samples, 23
plot_variation, 24
plot_variation_vs_total, 25
reduce_OpenSWATH_output, 26

sample_annotation, 27
Spyogenes, 28
Study_design, 28
SWATH2stats (SWATH2stats-package), 2
SWATH2stats-package, 2
transform_MSstats_OpenSWATH, 28
write_matrix_peptides, 29
write_matrix_proteins, 30