Package ‘MSstats’

January 19, 2024

Title  Protein Significance Analysis in DDA, SRM and DIA for Label-free or Label-based Proteomics Experiments

Version  4.10.0

Date  2022-06-27

Description  A set of tools for statistical relative protein significance analysis in DDA, SRM and DIA experiments.

License  Artistic-2.0

Depends  R (>= 4.0)

Imports  MSstatsConvert, data.table, checkmate, MASS, limma, lme4, preprocessCore, survival, utils, Rcpp, ggplot2, ggrepel, gplots, marray, stats, grDevices, graphics, methods, statmod

Suggests  BiocStyle, knitr, rmarkdown, tinytest, covr, markdown

VignetteBuilder  knitr

biocViews  ImmunoOncology, MassSpectrometry, Proteomics, Software, Normalization, QualityControl, TimeCourse

LazyData  true

URL  http://msstats.org

BugReports  https://groups.google.com/forum/#!forum/msstats

RoxygenNote  7.2.3

Encoding  UTF-8

NeedsCompilation  no

LinkingTo  Rcpp, RcppArmadillo

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

git_branch  RELEASE_3_18

git_last_commit  c97c6fc

git_last_commit_date  2023-10-24

Repository  Bioconductor 3.18

Date/Publication  2024-01-19
**Author**  Meena Choi [aut, cre],
Mateusz Staniak [aut],
Tsung-Heng Tsai [aut],
Ting Huang [aut],
Olga Vitek [aut]

**Maintainer**  Meena Choi <mnchoi67@gmail.com>

**R topics documented:**

- .addCoverageInfo .................................................. 5
- .addModelInformation .............................................. 5
- .addModelVariances .................................................. 6
- .addNInformativeInfo ............................................... 6
- .addNoisyFlag ....................................................... 7
- .addOutlierCutoff ................................................... 7
- .addOutlierInformation ............................................ 8
- .addSurvivalPredictions .......................................... 8
- .adjustLRuns .......................................................... 9
- .calculateOutlierCutoff ............................................ 9
- .calculatePower ..................................................... 10
- .calculateProteinVariance ....................................... 10
- .checkContrastMatrix .............................................. 11
- .checkDataProcessParams ....................................... 11
- .checkExperimentDesign .......................................... 12
- .checkGCPlotsInput ............................................... 12
- .checkGroupComparisonInput .................................... 13
- .checkSingleFeature .............................................. 13
- .checkSingleLabelProteins ..................................... 14
- .checkSingleSubject ............................................... 14
- .checkTechReplicate ............................................... 15
- .checkUnProcessedDataValidity .................................. 15
- .countInformative .................................................. 16
- .countMissingPercentage ......................................... 16
- .documentFunction ............................................... 17
- .finalizeInput ...................................................... 18
- .finalizeLinear ....................................................... 18
- .finalizeTMP .......................................................... 19
- .fitHuber .............................................................. 19
- .fitLinearModel ..................................................... 20
- .fitModelForGroupComparison .................................. 20
- .fitModelSingleProtein .......................................... 21
- .fitTukey .............................................................. 22
- .flagLowCoverage ................................................... 22
- .flagUninformativeSingleLabel .................................. 23
- .getAllComparisons .............................................. 23
- .getContrast .......................................................... 24
- .getContrastLabels .................................................. 24
R topics documented:

- .getEmptyComparison .......................................................... 25
- .getFeatureVariances .......................................................... 25
- .getMedian ................................................................. 26
- .getMedianSigmaSubject .................................................... 26
- .getMin ................................................................. 26
- .getModelParameters .......................................................... 27
- .getNonMissingFilter .......................................................... 27
- .getNonMissingFilterStats .................................................... 28
- .getNumSample ............................................................... 28
- .getSingleProteinForProfile .................................................. 29
- .getVarComponent ............................................................. 29
- .getWideTable ............................................................... 30
- .getYaxis ................................................................. 30
- .handleEmptyConditions ....................................................... 31
- .handleSingleContrast .......................................................... 31
- .isSummarizable ............................................................... 32
- .logDatasetInformation ......................................................... 32
- .logMissingness ............................................................... 33
- .logSingleLabeledProteins ...................................................... 33
- .logSummaryStatistics .......................................................... 34
- .makeComparison ............................................................... 34
- .makeConditionPlot ............................................................. 35
- .makeFactorColumns ............................................................. 36
- .makeHeatmap ................................................................. 36
- .makeProfilePlot ............................................................... 37
- .makeQCPlot ................................................................. 38
- .makeSummaryProfilePlot ....................................................... 40
- .makeVolcano ................................................................. 41
- .nicePrint ................................................................. 42
- .normalizeGlobalStandards ...................................................... 42
- .normalizeMedian ............................................................. 43
- .normalizeQuantile ............................................................ 43
- .onLoad ................................................................. 43
- .plotComparison ............................................................... 44
- .plotHeatmap ................................................................. 45
- .plotVolcano ................................................................. 46
- .prepareForDataProcess ......................................................... 48
- .prepareLinear ............................................................... 48
- .prepareSingleProteinForGC .................................................. 49
- .prepareSummary ............................................................... 49
- .prepareTMP ................................................................. 50
- .preProcessIntensities .......................................................... 50
- .quantileNormalizationSingleLabel ........................................ 51
- .replaceZerosWithNA ............................................................ 51
- .runTukey ................................................................. 52
- .saveSessionInfo ............................................................... 52
- .saveTable ................................................................. 53
- .selectHighQualityFeatures ................................................... 53
Description
Add coverage information to a data.table

Usage
.addCoverageInfo(input)

Arguments
input data.table

Value
data.table

Description
Add model information

Usage
.addModelInformation(input)

Arguments
input data.table

Value
data.table
**Description**
Add model variances

**Usage**
```
.addModelVariances(input)
```

**Arguments**
- `input` data.table

**Value**
data.table

---

**Description**
Add information about number of informative features

**Usage**
```
.addNInformativeInfo(input, min_feature_count, column)
```

**Arguments**
- `input` data.table
- `min_feature_count` minimum number of quality features to consider
- `column` name of a column used for filtering

**Value**
data.table
.addNoisyFlag  

Add flag for noisy features

Description
Add flag for noisy features

Usage
.addNoisyFlag(input)

Arguments
input  data.table

Value
data.table

.addOutlierCutoff  

Add outlier cutoff

Description
Add outlier cutoff

Usage
.addOutlierCutoff(input, quantile_order = 0.01)

Arguments
input  data.table
quantile_order  quantile used to label outliers

Value
data.table
.addOutlierInformation

*Add flag for outlier*

**Description**

Add flag for outlier

**Usage**

```
.addOutlierInformation(input, tol = 3, keep_run = FALSE)
```

**Arguments**

- `input`: data.table
- `tol`: cutoff for outliers
- `keep_run`: if TRUE, completely missing runs will be kept

**Value**

logical

---

.addSurvivalPredictions

*Get predicted values from a survival model*

**Description**

Get predicted values from a survival model

**Usage**

```
.addSurvivalPredictions(input)
```

**Arguments**

- `input`: data.table

**Value**

numeric vector of predictions
### .adjustLRuns

Adjust summarized abundance based on the heavy channel

**Description**

Adjust summarized abundance based on the heavy channel

**Usage**

```r
.adjustLRuns(input, rename = FALSE)
```

**Arguments**

- **input**: data.table
- **rename**: if TRUE, rename the output column to LogIntensities

**Value**

data.table

---

### .calculateOutlierCutoff

Calculate cutoff to label outliers

**Description**

Calculate cutoff to label outliers

**Usage**

```r
.calculateOutlierCutoff(input, quantile_order = 0.01)
```

**Arguments**

- **input**: data.table
- **quantile_order**: quantile used to label outliers

**Value**

numeric
**.calculatePower**  
*Power calculation*

**Description**

Power calculation

**Usage**

```
calculatePower(
    desiredFC,  
    FDR,       
    delta,     
    median_sigma_error,  
    median_sigma_subject,  
    numSample
)
```

**Arguments**

- **desiredFC**: the range of a desired fold change which includes the lower and upper values of the desired fold change.
- **FDR**: a pre-specified false discovery ratio (FDR) to control the overall false positive rate. Default is 0.05
- **delta**: difference between means (\(\mu\))
- **median_sigma_error**: median of error standard deviation
- **median_sigma_subject**: median standard deviation per subject
- **numSample**: minimal number of biological replicates per condition. TRUE represents you require to calculate the sample size for this category, else you should input the exact number of biological replicates.

**.calculateProteinVariance**  
*Calculate protein variances*

**Description**

Calculate protein variances

**Usage**

```
calculateProteinVariance(input)
```
Arguments
input data.table

Value
list of residuals, degrees of freedom and variances

Description
Check if contrast matrix includes all conditions

Usage
.checkContrastMatrix(contrast_matrix, input)

Arguments
contrast_matrix
contrast matrix
input data.table of summarized data

Description
Check validity of parameters to dataProcess function

Usage
.checkDataProcessParams(
  log_base,
  normalization_method,
  standards_names,
  feature_selection,
  summarization,
  imputation
)
`.checkExperimentDesign`

*Check if a given column exists in the data*

**Description**
Check if a given column exists in the data

**Usage**
```
.checkExperimentDesign(input, column_name)
```

**Arguments**
- `input`: data.table
- `column_name`: chr, name of a column to check

---

`.checkGCPlotsInput`

*Check groupComparisonPlots parameters*

**Description**
Check groupComparisonPlots parameters

**Usage**
```
.checkGCPlotsInput(type, log_base, selected_labels, all_labels)
```

**Arguments**
- `type`: type of a plot: HEATMAP/VOLCANO/COMPARISON
- `log_base`: 2 or 10
- `selected_labels`: character vector of contrast labels
- `all_labels`: character vector of all contrast labels
.checkGroupComparisonInput

Check if groupComparison input was processed by the dataProcess function

Description

Check if groupComparison input was processed by the dataProcess function

Usage

.checkGroupComparisonInput(input)

Arguments

input       data.table

---

.checkSingleFeature

Check if data has less than two features

Description

Check if data has less than two features

Usage

.checkSingleFeature(input)

Arguments

input       data.table

Value

logical
.checkSingleLabelProteins

*Check if there are proteins with a single label in a labeled dataset*

**Description**

Check if there are proteins with a single label in a labeled dataset

**Usage**

`.checkSingleLabelProteins(input)`

**Arguments**

| input  | data.table |

**Value**

TRUE invisibly

---

.checkSingleSubject

*Check if there is only single subject*

**Description**

Check if there is only single subject

**Usage**

`.checkSingleSubject(input)`

**Arguments**

| input  | data.table |

---
.checkTechReplicate

Description

Check if there are technical replicates

Usage

.checkTechReplicate(input)

Arguments

input data.table

.checkUnProcessedDataValidity

Description

Check validity of data that were not processed by MSstats converter

Usage

.checkUnProcessedDataValidity(input, fix_missing, fill_incomplete)

Arguments

input data.table

fix_missing str, optional. Defaults to NULL, which means no action. If not NULL, must be one of the options: "zero_to_na" or "na_to_zero". If "zero_to_na", Intensity values equal exactly to 0 will be converted to NA. If "na_to_zero", missing values will be replaced by zeros.
### .countInformative

**Count informative features**

**Description**

Count informative features

**Usage**

```r
.countInformative(input, column)
```

**Arguments**

- `input`: data.table
- `column`: name of a column used for filtering

**Value**

numeric

---

### .countMissingPercentage

**Count percentage of missing values in given conditions**

**Description**

Count percentage of missing values in given conditions

**Usage**

```r
.countMissingPercentage(
  contrast_matrix,
  summarized,
  result,
  samples_info,
  has_imputed
)
```

**Arguments**

- `contrast_matrix`: contrast matrix
- `summarized`: data.table summarized by the dataProcess function
- `result`: result of groupComparison
- `samples_info`: number of runs per group
- `has_imputed`: if TRUE, missing values have been imputed by dataProcess
A dummy function to store shared documentation items.

Description

A dummy function to store shared documentation items.

Usage

`.documentFunction()`

Arguments

- **removeFewMeasurements**
  - TRUE (default) will remove the features that have 1 or 2 measurements across runs.

- **useUniquePeptide**
  - TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

- **summaryforMultipleRows**
  - max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.

- **removeProtein_with1Feature**
  - TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.

- **removeProtein_with1Peptide**
  - TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

- **removeOxidationMpeptides**
  - TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.

- **removeMpeptides**
  - TRUE will remove the peptides including 'M' sequence. FALSE is default.

- **use_log_file**
  - logical. If TRUE, information about data processing will be saved to a file.

- **append**
  - logical. If TRUE, information about data processing will be added to an existing log file.

- **verbose**
  - logical. If TRUE, information about data processing will be printed to the console.

- **log_file_path**
  - character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.
**.finalizeInput**  
*Add summary statistics to dataProcess output*

**Description**

Add summary statistics to dataProcess output

**Usage**

```r
.finalizeInput(input, summarized, method, impute, censored_symbol)
```

**Arguments**

- `input`: feature-level data
- `summarized`: protein-level data (list)
- `method`: summary method
- `impute`: if TRUE, censored missing values were imputed
- `censored_symbol`: censored missing value indicator

**.finalizeLinear**  
*Summary statistics for linear model-based summarization*

**Description**

Summary statistics for linear model-based summarization

**Usage**

```r
.finalizeLinear(input, censored_symbol)
```

**Arguments**

- `input`: feature-level data
- `censored_symbol`: censored missing value indicator
.finalizeTMP

Summary statistics for output of TMP-based summarization

Description

Summary statistics for output of TMP-based summarization

Usage

.finalizeTMP(input, censored_symbol, impute, summarized)

Arguments

input feature-level data
censored_symbol censored missing value indicator
impute if TRUE, censored missing values were imputed
summarized protein-level data (list)

.fitHuber

Wrapper to fit robust linear model for one protein

Description

Wrapper to fit robust linear model for one protein

Usage

.fitHuber(input)

Value

rlm
.fitLinearModel  
Fit a linear model

Description
Fit a linear model

Usage
.fitLinearModel(input, is_single_feature, is_labeled, equal_variances)

Arguments
input  
data.table
is_single_feature  
logical, if TRUE, data has single feature
is_labeled  
logical, if TRUE, data comes from a labeled experiment
equal_variances  
logical, if TRUE, equal variances are assumed

Value
lm or merMod

---

.fitModelForGroupComparison  
Choose a model type (fixed/mixed effects) and fit it for a single protein

Description
Choose a model type (fixed/mixed effects) and fit it for a single protein

Usage
.fitModelForGroupComparison(input, repeated, is_single_subject, has_tech_replicates)
.fitModelSingleProtein

Arguments

- **input**: data.table of summarized data
- **repeated**: if TRUE, experiment consists of repeated measurements
- **is_single_subject**: if TRUE, experiment consists of a single subject
- **has_tech_replicates**: if TRUE, there are technical replicates

Description

Fit model and perform group comparison for a single protein

Usage

```r
.fitModelSingleProtein(
  input,
  contrast_matrix,
  has_tech_replicates,
  is_single_subject,
  repeated,
  groups,
  samples_info,
  save_fitted_models,
  has_imputed
)
```

Arguments

- **input**: data.table of summarized data
- **contrast_matrix**: contrast matrix
- **has_tech_replicates**: if TRUE, there are technical replicates
- **is_single_subject**: if TRUE, experiment consists of a single subject
- **repeated**: if TRUE, experiment consists of repeated measurements
- **groups**: unique labels for experimental conditions
- **samples_info**: number of runs per group
- **save_fitted_models**: if TRUE, fitted model will be saved. If FALSE, it will be replaced by NULL
- **has_imputed**: if TRUE, missing values have been imputed by dataProcess
.fitTukey  
*Fit tukey median polish for a data matrix*

**Description**
Fit tukey median polish for a data matrix

**Usage**
```
.fitTukey(input)
```

**Arguments**
- `input`  
data.table with data for a single protein

**Value**
data.table

---

(flagLowCoverage)

**Description**
Flag for low coverage features

**Usage**
```
.flagLowCoverage(input)
```

**Arguments**
- `input`  
data.table

**Value**
logical
.flagUninformativeSingleLabel

Flag uninformative features

**Description**
Flag uninformative features

**Usage**
.flagUninformativeSingleLabel(input, min_feature_count = 2)

**Arguments**
- `input`: data.table
- `min_feature_count`: minimum number of quality features to consider

**Value**
data.table

---

.getAllComparisons

Get all comparisons for a single protein and a contrast matrix

**Description**
Get all comparisons for a single protein and a contrast matrix

**Usage**
.getAllComparisons(input, fitted_model, contrast_matrix, groups, protein)

**Arguments**
- `input`: summarized data
- `fitted_model`: model fitted by the .fitModelForGroupComparison function
- `contrast_matrix`: contrast matrix
- `groups`: unique labels of experimental conditions
- `protein`: name of a protein
.getContrast

Create a contrast for a model with only group as a fixed effect

Description

Create a contrast for a model with only group as a fixed effect

Usage

.getContrast(input, contrast, coefs, groups)

Arguments

input summarized data for a single protein
coefs coefficients of a linear model (named vector)
groups unique group labels
contrast_matrix row of a contrast_matrix

.getContrastLabels

Get labels for contrasts

Description

Get labels for contrasts

Usage

.getContrastLabels(contrasts)

Arguments

contrasts list of lists of condition labels
### `.getEmptyComparison`

**Comparison output when there are measurements only in a single condition**

**Description**

Comparison output when there are measurements only in a single condition

**Usage**

```r
.getEmptyComparison(input, contrast_matrix, groups, protein)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>summarized data</td>
</tr>
<tr>
<td>contrast_matrix</td>
<td>contrast matrix</td>
</tr>
<tr>
<td>groups</td>
<td>unique labels of experimental conditions</td>
</tr>
<tr>
<td>protein</td>
<td>name of a protein</td>
</tr>
</tbody>
</table>

### `.getFeatureVariances`

**Calculate variances of features**

**Description**

Calculate variances of features

**Usage**

```r
.getFeatureVariances(input, tolerance = 3)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>data.table</td>
</tr>
<tr>
<td>tolerance</td>
<td>cutoff for outliers</td>
</tr>
</tbody>
</table>

**Value**

numeric
.getMedian

*Get median of protein abundances for a given label*

**Description**
Get median of protein abundances for a given label

**Usage**
```
.getMedian(df, label)
```

**Arguments**
- `df` : 'data.table'
- `label` : "L" for light isotopes, "H" for heavy isotopes.

---

.getMedianSigmaSubject

*Get median per subject or group by subject*

**Description**
Get median per subject or group by subject

**Usage**
```
.getMedianSigmaSubject(var_component)
```

**Arguments**
- `var_component` : data.frame, output of .getVarComponent

---

.getMin

*Utility function: get 0.99 * minimum of non-missing values*

**Description**
Utility function: get 0.99 * minimum of non-missing values

**Usage**
```
.getMin(abundance, nonmissing)
```

**Arguments**
- `abundance` : abundances values
- `nonmissing` : logical vector
Description

Get params (coefficients, covariance matrix, degrees of freedom) from a model

Usage

.getModelParameters(fitted_model)

Arguments

fitted_model object of class lm or lmerMod

Description

Identify non-missing values

Usage

.getNonMissingFilter(input, impute, censored_symbol)

Arguments

input 'data.table' in MSstats format
impute if TRUE, missing values are supposed to be imputed
censored_symbol 'censoredInt' parameter to dataProcess
**.getNumSample**

Get sample size

**Description**

Get sample size

**Usage**

```
.getNumSample(
    desiredFC, power, alpha, delta,
    median_sigma_error, median_sigma_subject
)
```

**Arguments**

- `desiredFC` (numeric): Desired fold change.
- `power` (numeric): Power of the test.
- `alpha` (numeric): Significance level.
- `delta` (numeric): Effect size.
- `median_sigma_error` (numeric): Median sigma error.
- `median_sigma_subject` (numeric): Median sigma subject.

**Value**

data.table

---

**.getNonMissingFilterStats**

Get a logical vector for non-missing values to calculate summary statistics

**Description**

Get a logical vector for non-missing values to calculate summary statistics

**Usage**

```
.getNonMissingFilterStats(input, censored_symbol)
```

**Arguments**

- `input` (data.table): Data.table with data for a single protein.
- `censored_symbol` (character): Missing values are censored or at random. 'NA' (default) assumes that all 'NA's in 'Intensity' column are censored. '0' uses zero intensities as censored intensity. In this case, NA intensities are missing at random. The output from Skyline should use '0'. Null assumes that all NA intensities are randomly missing.

**Value**

data.table
Arguments

- **desiredFC**: the range of a desired fold change which includes the lower and upper values of the desired fold change.
- **power**: a pre-specified statistical power which defined as the probability of detecting a true fold change. TRUE represent you require to calculate the power for this category, else you should input the average of power you expect. Default is 0.9
- **alpha**: significance level
- **delta**: difference between means (?)
- **median_sigma_error**: median of error standard deviation
- **median_sigma_subject**: median standard deviation per subject

**Description**

Get data for a single protein to plot

**Usage**

```
.getSingleProteinForProfile(processed, all_proteins, i)
```

**Arguments**

- **all_proteins**: character, set of protein names
- **i**: integer, index of protein to use
- **dataProcess**: output -> FeatureLevelData

**.getVarComponent**

*Get variances from models fitted by the groupComparison function*

**Description**

Get variances from models fitted by the groupComparison function

**Usage**

```
.getVarComponent(fitted_models)
```

**Arguments**

- **fitted_models**: FittedModels element of groupComparison output
.getWideTable  
Utility function for quantile normalization - get table in wide format

Description
Utility function for quantile normalization - get table in wide format

Usage
.getWideTable(input, runs, label = "L", remove_missing = TRUE)

Arguments
input  'data.table' in MSstats standard format
label  "L" for light isotopes, "H" for heavy isotopes
remove_missing  if TRUE, only non-missing values will be considered
vector  of run labels

.getYaxis  
Get name for y-axis

Description
Get name for y-axis

Usage
.getYaxis(temp)

Arguments
temp  data.table
**Description**

Handle contrast when some of the conditions are missing

**Usage**

```r
.handleEmptyConditions(input, fit, contrast, groups, parameters, protein, empty_conditions, coefs)
```

**Arguments**

- `input`: summarized data
- `contrast`: single row of a contrast matrix
- `groups`: unique labels of experimental conditions
- `parameters`: parameters extracted from the model
- `protein`: name of a protein
- `empty_conditions`: labels of empty conditions
- `coefs`: coefficient of the fitted model

**Description**

Group comparison for a single contrast

**Usage**

```r
.handleSingleContrast(input, fit, contrast, groups, parameters, protein, coefs)
```
Arguments

- **input**: summarized data
- **contrast**: single row of a contrast matrix
- **groups**: unique labels of experimental conditions
- **parameters**: parameters extracted from the model
- **protein**: name of a protein
- **coefs**: coefficient of the fitted model

---

**.isSummarizable**

*Check if a protein can be summarized with TMP*

---

Description

Check if a protein can be summarized with TMP

Usage

```r
.isSummarizable(input, remove50missing)
```

Arguments

- **input**: data.table
- **remove50missing**: if TRUE, proteins with more than 50 in all runs will not be summarized

Value

data.table

---

**.logDatasetInformation**

*Log information about feature-level data*

Description

Log information about feature-level data

Usage

```r
.logDatasetInformation(input)
```

Arguments

- **input**: data.table
**Value**

TRUE invisibly after successful logging

---

**Description**

Log information about missing data

**Usage**

.logMissingness(input)

**Arguments**

- **input**: data.table

**Value**

TRUE invisibly

---

**Description**

Print proteins with a single label to the log file

**Usage**

.logSingleLabeledProteins(input, label)

**Arguments**

- **input**: data.table
- **label**: label ("L" or "H")

**Value**

TRUE invisibly
.logSummaryStatistics  Print summary statistics to the log file

Description
Print summary statistics to the log file

Usage
.logSummaryStatistics(input)

Arguments
input  data.table

Value
TRUE invisibly

.makeComparison  Create comparison plot

Description
Create comparison plot

Usage
.makeComparison(
  input,
  log_base,
  dot.size,
  x.axis.size,
  y.axis.size,
  text.angle,
  hjust,
  vjust,
  y.limdown,
  y.limup
)
.makeConditionPlot

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>data.table</td>
</tr>
<tr>
<td>log_base</td>
<td>2 or 10</td>
</tr>
<tr>
<td>dot.size</td>
<td>size of dots in volcano plot and comparison plot. Default is 3.</td>
</tr>
<tr>
<td>x.axis.size</td>
<td>size of axes labels, e.g. name of the comparisons in heatmap, and in comparison plot. Default is 10.</td>
</tr>
<tr>
<td>y.axis.size</td>
<td>size of axes labels, e.g. name of targeted proteins in heatmap. Default is 10.</td>
</tr>
<tr>
<td>text.angle</td>
<td>angle of x-axis labels represented each comparison at the bottom of graph in comparison plot. Default is 0.</td>
</tr>
</tbody>
</table>

Description

Make condition plot

Usage

.makeConditionPlot(input, scale, single_protein, y.limdown, y.limup, x.axis.size, y.axis.size, text.size, text.angle, legend.size, dot.size.condition, yaxis.name)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>data.table</td>
</tr>
<tr>
<td>scale</td>
<td>for &quot;ConditionPlot&quot; only, FALSE(default) means each conditional level is not scaled at x-axis according to its actual value (equal space at x-axis). TRUE means each conditional level is scaled at x-axis according to its actual value (unequal space at x-axis).</td>
</tr>
<tr>
<td>single_protein</td>
<td>data.table</td>
</tr>
<tr>
<td>x.axis.size</td>
<td>size of x-axis labeling for &quot;Run&quot; in Profile Plot and QC Plot, and &quot;Condition&quot; in Condition Plot. Default is 10.</td>
</tr>
</tbody>
</table>
`.makeHeatmap`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>y.axis.size</code></td>
<td>size of y-axis labels. Default is 10.</td>
</tr>
<tr>
<td><code>text.size</code></td>
<td>size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.</td>
</tr>
<tr>
<td><code>text.angle</code></td>
<td>angle of labels represented each condition at the top of graph in Profile Plot and QC plot or x-axis labeling in Condition plot. Default is 0.</td>
</tr>
<tr>
<td><code>legend.size</code></td>
<td>size of feature legend (transition-level or peptide-level) above graph in Profile Plot. Default is 7.</td>
</tr>
<tr>
<td><code>dot.size.condition</code></td>
<td>size of dots in condition plot. Default is 3.</td>
</tr>
</tbody>
</table>

---

`.makeFactorColumns` *Make factor columns where needed*

**Description**

Make factor columns where needed

**Usage**

`.makeFactorColumns(input)`

**Arguments**

- `input` data.table

---

`.makeHeatmap` *Create heatmap*

**Description**

Create heatmap

**Usage**

`.makeHeatmap(input, my.colors, my.breaks, x.axis.size, y.axis.size)`

**Arguments**

- `input` data.table
- `x.axis.size` size of axes labels, e.g. name of the comparisons in heatmap, and in comparison plot. Default is 10.
- `y.axis.size` size of axes labels, e.g. name of targeted proteins in heatmap. Default is 10.
Description
Create profile plot

Usage
.makeProfilePlot(
  input,
  is_censored,
  featureName,
  y.limdown,
  y.limup,
  x.axis.size,
  y.axis.size,
  text.size,
  text.angle,
  legend.size,
  dot.size.profile,
  ss,
  s,
  cumGroupAxis,
  yaxis.name,
  lineNameAxis,
  groupNametemp,
  dot_colors
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>data.table</td>
</tr>
<tr>
<td>is_censored</td>
<td>TRUE if censored values were imputed</td>
</tr>
<tr>
<td>featureName</td>
<td>for &quot;ProfilePlot&quot; only. &quot;Transition&quot; (default) means printing feature legend in transition-level; &quot;Peptide&quot; means printing feature legend in peptide-level; &quot;NA&quot; means no feature legend printing.</td>
</tr>
<tr>
<td>x.axis.size</td>
<td>size of x-axis labeling for &quot;Run&quot; in Profile Plot and QC Plot, and &quot;Condition&quot; in Condition Plot. Default is 10.</td>
</tr>
<tr>
<td>y.axis.size</td>
<td>size of y-axis labels. Default is 10.</td>
</tr>
<tr>
<td>text.size</td>
<td>size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.</td>
</tr>
<tr>
<td>text.angle</td>
<td>angle of labels represented each condition at the top of graph in Profile Plot and QC plot or x-axis labeling in Condition plot. Default is 0.</td>
</tr>
</tbody>
</table>
**.makeQCPlot**

**Legend**

- `legend.size`: size of feature legend (transition-level or peptide-level) above graph in Profile Plot. Default is 7.
- `dot.size.profile`: size of dots in profile plot. Default is 2.

---

**Description**

To illustrate the quantitative data after data-preprocessing and quality control of MS runs, dataProcessPlots takes the quantitative data from function (`dataProcess`) as input and automatically generate three types of figures in pdf files as output: (1) profile plot (specify "ProfilePlot" in option type), to identify the potential sources of variation for each protein; (2) quality control plot (specify "QCPlot" in option type), to evaluate the systematic bias between MS runs; (3) mean plot for conditions (specify "ConditionPlot" in option type), to illustrate mean and variability of each condition per protein.

**Usage**

```r
.makeQCPlot(
  input,  # data.table
  all_proteins,  # character vector of protein names
  y.limdown,
  y.limup,
  x.axis.size,  # size of x-axis labeling for "Run" in Profile Plot and QC Plot, and "Condition" in Condition Plot. Default is 10.
  y.axis.size,
  text.size,  # size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.
  text.angle,
  legend.size,
  label.color,
  cumGroupAxis,
  groupName,
  lineNameAxis,
  yaxis.name
)
```

**Arguments**

- `input`: data.table
- `all_proteins`: character vector of protein names
- `x.axis.size`: size of x-axis labeling for "Run" in Profile Plot and QC Plot, and "Condition" in Condition Plot. Default is 10.
- `y.axis.size`: size of y-axis labels. Default is 10.
- `text.size`: size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.
text.angle  angle of labels represented each condition at the top of graph in Profile Plot and QC plot or x-axis labeling in Condition plot. Default is 0.
legend.size  size of feature legend (transition-level or peptide-level) above graph in Profile Plot. Default is 7.

Details

- **Profile Plot**: identify the potential sources of variation of each protein. `QuantData$FeatureLevelData` is used for plots. X-axis is run. Y-axis is log-intensities of transitions. Reference/endogenous signals are in the left/right panel. Line colors indicate peptides and line types indicate transitions. In summarization plots, gray dots and lines are the same as original profile plots with `QuantData$FeatureLevelData`. Dark dots and lines are for summarized intensities from `QuantData$ProteinLevelData`.

- **QC Plot**: illustrate the systematic bias between MS runs. After normalization, the reference signals for all proteins should be stable across MS runs. `QuantData$FeatureLevelData` is used for plots. X-axis is run. Y-axis is log-intensities of transition. Reference/endogenous signals are in the left/right panel. The pdf file contains (1) QC plot for all proteins and (2) QC plots for each protein separately.

- **Condition Plot**: illustrate the systematic difference between conditions. Summarized intensities from `QuantData$ProteinLevelData` are used for plots. X-axis is condition. Y-axis is summarized log transformed intensity. If scale is TRUE, the levels of conditions is scaled according to its actual values at x-axis. Red points indicate the mean for each condition. If interval is "CI", blue error bars indicate the confidence interval with 0.95 significant level for each condition. If interval is "SD", blue error bars indicate the standard deviation for each condition. The interval is not related with model-based analysis.

The input of this function is the quantitative data from function `dataProcess`.

Examples

```r
# Consider quantitative data (i.e. QuantData) from a yeast study with ten time points of interests, # three biological replicates, and no technical replicates which is a time-course experiment. # The goal is to provide pre-analysis visualization by automatically generate two types of figures # in two separate pdf files.
# Protein IDHC (gene name IDP2) is differentially expressed in time point 1 and time point 7, # whereas, Protein PMG2 (gene name GPM2) is not.

QuantData<-dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
# Profile plot
dataProcessPlots(data=QuantData, type="ProfilePlot")
# Quality control plot
dataProcessPlots(data=QuantData, type="QCPlot")
# Quantification plot for conditions
dataProcessPlots(data=QuantData, type="ConditionPlot")
```
Description

Make summary profile plot

Usage

```
.makeSummaryProfilePlot(
  input,
  is_censored,  
y.limdown,  
y.limup,  
x.axis.size,  
y.axis.size,  
text.size,  
text.angle,  
legend.size,  
dot.size.profile,  
cumGroupAxis,  
yaxis.name,  
lineNameAxis,  
groupNametemp
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>data.table</td>
</tr>
<tr>
<td>is_censored</td>
<td>TRUE if censored values were imputed</td>
</tr>
<tr>
<td>x.axis.size</td>
<td>size of x-axis labeling for &quot;Run&quot; in Profile Plot and QC Plot, and &quot;Condition&quot; in Condition Plot. Default is 10.</td>
</tr>
<tr>
<td>y.axis.size</td>
<td>size of y-axis labels. Default is 10.</td>
</tr>
<tr>
<td>text.size</td>
<td>size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.</td>
</tr>
<tr>
<td>text.angle</td>
<td>angle of labels represented each condition at the top of graph in Profile Plot and QC plot or x-axis labeling in Condition plot. Default is 0.</td>
</tr>
<tr>
<td>legend.size</td>
<td>size of feature legend (transition-level or peptide-level) above graph in Profile Plot. Default is 7.</td>
</tr>
<tr>
<td>dot.size.profile</td>
<td>size of dots in profile plot. Default is 2.</td>
</tr>
</tbody>
</table>
.makeVolcano

Create a volcano plot

Description

Create a volcano plot

Usage

.makeVolcano(
input,
label_name,
log_base_FC,
log_base_pval,
x.lim,
ProteinName,
dot.size,
y.limdown,
y.limup,
text.size,
FCcutoff,
sig,
x.axis.size,
y.axis.size,
legend.size,
log_adjp
)

Arguments

input data.table
label_name contrast label
log_base_FC 2 or 10
log_base_pval 2 or 10
ProteinName for volcano plot only, whether display protein names or not. TRUE (default)
means protein names, which are significant, are displayed next to the points. FALSE
means no protein names are displayed.
dot.size size of dots in volcano plot and comparison plot. Default is 3.
text.size size of ProteinName label in the graph for Volcano Plot. Default is 4.
FCcutoff for volcano plot or heatmap, whether involve fold change cutoff or not. FALSE
(default) means no fold change cutoff is applied for significance analysis. FC-
cutoff = specific value means specific fold change cutoff is applied.
sig FDR cutoff for the adjusted p-values in heatmap and volcano plot. level of sig-
nificance for comparison plot. 100(1-sig)% confidence interval will be drawn. sig=0.05 is default.
.normalizeGlobalStandards

Description
Normalization based on standards

Usage
.normalizeGlobalStandards(input, peptides_dict, standards)

Arguments
- input: data.table in MSstats format
- peptides_dict: 'data.table' of names of peptides and their corresponding features.
- standards: character vector with names of standards, required if "GLOBALSTANDARDS" method was selected.

.nicePrint
Print a table nicely

Description
Print a table nicely

Usage
.nicePrint(string_vector)

Arguments
- string_vector: character

Value
character
**.normalizeMedian**  
*Median normalization*

**Description**
Median normalization

**Usage**
`.normalizeMedian(input)`

**Arguments**
- `input`  
  'data.table' in standard MSstats format

---

**.normalizeQuantile**  
*Quantile normalization based on the `preprocessCore` package*

**Description**
Quantile normalization based on the `preprocessCore` package

**Usage**
`.normalizeQuantile(input)`

**Arguments**
- `input`  
  'data.table' in MSstats standard format

---

**.onLoad**  
*Set default logging object when package is loaded*

**Description**
Set default logging object when package is loaded

**Usage**
`.onLoad(...)`

**Arguments**
- `...`  
  ignored

**Value**
none, sets options called MSstatsLog and MSstatsMsg
.plotComparison

Preprocess data for comparison plots and create them

Description

Preprocess data for comparison plots and create them

Usage

.plotComparison(
  input,         data.table
  proteins,     
  address,      
  width,        
  height,       
  sig,          
  ylimUp,       
  ylimDown,     
  text.angle,   
  dot.size,     
  x.axis.size,  
  y.axis.size,  
  log_base_FC   
)

Arguments

input   data.table
address the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of "VolcanoPlot.pdf" or "Heatmap.pdf" or "ComparisonPlot.pdf". The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

width   width of the saved file. Default is 10.
height  height of the saved file. Default is 10.
sig     FDR cutoff for the adjusted p-values in heatmap and volcano plot. level of significance for comparison plot. 100(1-sig)% confidence interval will be drawn. sig=0.05 is default.
ylimUp  for all three plots, upper limit for y-axis. FALSE (default) for volcano plot/heatmap use maximum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses maximum of log-fold change + CI.
ylimDown for all three plots, lower limit for y-axis. FALSE (default) for volcano plot/heatmap use minimum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses minimum of log-fold change - CI.
.plotHeatmap

angle of x-axis labels represented each comparison at the bottom of graph in comparison plot. Default is 0.

dot.size

size of dots in volcano plot and comparison plot. Default is 3.

x.axis.size

size of axes labels, e.g. name of the comparisons in heatmap, and in comparison plot. Default is 10.

y.axis.size

size of axes labels, e.g. name of targeted proteins in heatmap. Default is 10.

log_base_FC

log base for log-fold changes - 2 or 10

Description

Prepare data for heatmaps and plot them

Usage

.plotHeatmap(
    input,
    log_base_pval,
    ylimUp,
    FCcutoff,
    sig,
    clustering,
    numProtein,
    colorkey,
    width,
    height,
    log_base_FC,
    x.axis.size,
    y.axis.size,
    address
)

Arguments

input data.table

log_base_pval log base for p-values

ylimUp for all three plots, upper limit for y-axis. FALSE (default) for volcano plot/heatmap use maximum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses maximum of log-fold change + CI.

FCcutoff for volcano plot or heatmap, whether involve fold change cutoff or not. FALSE (default) means no fold change cutoff is applied for significance analysis. FC-cutoff = specific value means specific fold change cutoff is applied.
.plotVolcano

Preprocess data for volcano plots and create them

Description

Preprocess data for volcano plots and create them

Usage

.plotVolcano(
  input,
  which.Comparison,
  address,
  width,
  height,
  log_base_FC,
  x.axis.size,
  y.axis.size,
  address,
  width,
  height,
  log_base_pval,
  ylimUp,
)
Arguments

which.Comparison
list of comparisons to draw plots. List can be labels of comparisons or order numbers of comparisons from levels(data$Label), such as levels(testResultMultiComparisons$ComparisonResult$Label). Default is "all", which generates all plots for each protein.

address
the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of "VolcanoPlot.pdf" or "Heatmap.pdf" or "ComparisonPlot.pdf". The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

width
width of the saved file. Default is 10.

height
height of the saved file. Default is 10.

ylimUp
for all three plots, upper limit for y-axis. FALSE (default) for volcano plot/heatmap use maximum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses maximum of log-fold change + CI.

ylimDown
for all three plots, lower limit for y-axis. FALSE (default) for volcano plot/heatmap use minimum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses minimum of log-fold change - CI.

FCcutoff
for volcano plot or heatmap, whether involve fold change cutoff or not. FALSE (default) means no fold change cutoff is applied for significance analysis. FC-cutoff = specific value means specific fold change cutoff is applied.

sig
FDR cutoff for the adjusted p-values in heatmap and volcano plot. level of significance for comparison plot. 100(1-sig)% confidence interval will be drawn. sig=0.05 is default.

xlinUp
for Volcano plot, the limit for x-axis. FALSE (default) for use maximum for absolute value of log-fold change or 3 as default if maximum for absolute value of log-fold change is less than 3.

ProteinName
for volcano plot only, whether display protein names or not. TRUE (default) means protein names, which are significant, are displayed next to the points. FALSE means no protein names are displayed.
dot.size  size of dots in volcano plot and comparison plot. Default is 3.
text.size  size of ProteinName label in the graph for Volcano Plot. Default is 4.
legend.size  size of legend for color at the bottom of volcano plot. Default is 7.
x.axis.size  size of axes labels, e.g. name of the comparisons in heatmap, and in comparison plot. Default is 10.
y.axis.size  size of axes labels, e.g. name of targeted proteins in heatmap. Default is 10.

---

**.prepareForDataProcess**

_Check validity of data already processed by MSstats converter_

**Description**

Check validity of data already processed by MSstats converter

**Usage**

`.prepareForDataProcess(input, ...)`

**Arguments**

- **input**  data.frame of class ‘MSstatsValidated’
- **..**  additional parameters, currently ignored

---

**.prepareLinear**  _Prepare feature-level data for linear summarization_

**Description**

Prepare feature-level data for linear summarization

**Usage**

`.prepareLinear(input, impute, censored_symbol)`

**Arguments**

- **input**  data.table
- **impute**  logical
- **censored_symbol**  "0"/"NA"

**Value**

data.table
.prepareSingleProteinForGC

Prepare data for a single protein for group comparison

Description

Prepare data for a single protein for group comparison

Usage

.prepareSingleProteinForGC(single_protein)

Arguments

single_protein  data.table

.prepareSummary

Prepare feature-level data for summarization

Description

Prepare feature-level data for summarization

Usage

.prepareSummary(input, method, impute, censored_symbol)

Arguments

input  data.table
method  "TMP" / "linear"
impute  logical
censored_symbol  "0"/"NA"

Value

data.table
.prepareTMP

Prepare feature-level data for TMP summarization

Description

Prepare feature-level data for TMP summarization

Usage

.prepareTMP(input, impute, censored_symbol)

Arguments

input       data.table
impute      logical
censored_symbol  "0"/"NA"

Value

data.table

_.preProcessIntensities

Create ABUNDANCE column and log-transform intensities

Description

Create ABUNDANCE column and log-transform intensities

Usage

_.preProcessIntensities(input, log_base)

Arguments

input       data.table
log_base    base of the logarithm
**Quantile normalization for a single label**

**Description**

Quantile normalization for a single label

**Usage**

```
.quantileNormalizationSingleLabel(input, runs, label = "L")
```

**Arguments**

- `input` : 'data.table' in MSstats standard format
- `runs` : run labels
- `label` : "L" for light isotopes, "H" for heavy isotopes

---

**Utility function for normalization: replace 0s by NA**

**Description**

Utility function for normalization: replace 0s by NA

**Usage**

```
.replaceZerosWithNA(vec)
```

**Arguments**

- `vec` : vector
.runTukey

*Fit Tukey median polish*

### Description

Fit Tukey median polish

### Usage

```
.runTukey(input, is_labeled, censored_symbol, remove50missing)
```

### Arguments

- **input**: data.table with data for a single protein
- **is_labeled**: logical, if TRUE, data is coming from an SRM experiment
- **censored_symbol**: Missing values are censored or at random. 'NA' (default) assumes that all 'NA's in 'Intensity' column are censored. '0' uses zero intensities as censored intensity. In this case, NA intensities are missing at random. The output from Skyline should use '0'. Null assumes that all NA intensities are randomly missing.
- **remove50missing**: only for summaryMethod = "TMP". TRUE removes the runs which have more than 50% missing values. FALSE is default.

### Value

data.table

---

.saveSessionInfo

*Save information about R session to sessionInfo.txt file.*

### Description

Save information about R session to sessionInfo.txt file.

### Usage

```
.saveSessionInfo()
```
### .saveTable

**Save a data table to a file**

**Description**

Save a data table to a file

**Usage**

```r
.saveTable(input, name_base, file_name)
```

**Arguments**

- **input**
  - data.table
- **name_base**
  - path to a folder (or "" for working directory)
- **file_name**
  - name of a file to save. If this file already exists, an integer will be appended to this name

### .selectHighQualityFeatures

**Select features of high quality**

**Description**

Select features of high quality

**Usage**

```r
.selectHighQualityFeatures(input, min_feature_count)
```

**Arguments**

- **input**
  - data.table
- **min_feature_count**
  - minimum number of quality features to consider

**Value**

- data.table
.selectTopFeatures  
Select features with highest average abundance

Description
Select features with highest average abundance

Usage
.selectTopFeatures(input, top_n)

Arguments
input  
data.table
top_n  
number of top features to select

Value
data.table

.setCensoredByThreshold
Set censored values based on minimum in run/feature/run or feature

Description
Set censored values based on minimum in run/feature/run or feature

Usage
.setCensoredByThreshold(input, censored_symbol, remove50missing)

Arguments
input  
‘data.table’ in MSstats format
censored_symbol  
censoredInt parameter to ‘dataProcess’
remove50missing  
if TRUE, features with at least 50 will be removed
.updateColumnsForProcessing

*Create columns for data processing*

**Description**

Create columns for data processing

**Usage**

`.updateColumnsForProcessing(input)`

**Arguments**

- `input`: data.table

---

.updateUnequalVariances

*Adjust model for unequal variances*

**Description**

Adjust model for unequal variances

**Usage**

`.updateUnequalVariances(input, fit, num_iter)`

**Arguments**

- `input`: data.table
- `fit`: lm
- `num_iter`: number of iterations

**Value**

merMod
checkRepeatedDesign \hspace{1cm} \textit{Check if data represents repeated measurements design}

\textbf{Description}

Check if data represents repeated measurements design

\textbf{Usage}

\begin{verbatim}
checkRepeatedDesign(summarization_output)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{summarization_output}
    \begin{itemize}
      \item output of the dataProcess function
    \end{itemize}
\end{itemize}

\textbf{Details}

This extracts information required by the group comparison workflow

\textbf{Value}

\begin{itemize}
  \item logical, TRUE if data represent repeated measurements design
\end{itemize}

\textbf{Examples}

\begin{verbatim}
QuantData1 <- dataProcess(SRMRawData, use_log_file = FALSE)
checkRepeatedDesign(QuantData1)
\end{verbatim}

\textbf{dataProcess} \hspace{1cm} \textit{Process MS data: clean, normalize and summarize before differential analysis}

\textbf{Description}

Process MS data: clean, normalize and summarize before differential analysis
**dataProcess**

**Usage**

```r
dataProcess(
  raw,
  logTrans = 2,
  normalization = "equalizeMedians",
  nameStandards = NULL,
  featureSubset = "all",
  remove_uninformative_feature_outlier = FALSE,
  min_feature_count = 2,
  n_top_feature = 3,
  summaryMethod = "TMP",
  equalFeatureVar = TRUE,
  censoredInt = "NA",
  MBimpute = TRUE,
  remove50missing = FALSE,
  fix_missing = NULL,
  maxQuantileforCensored = 0.999,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL
)
```

**Arguments**

- **raw**
  - name of the raw (input) data set.
- **logTrans**
  - base of logarithm transformation: 2 (default) or 10.
- **normalization**
  - normalization to remove systematic bias between MS runs. There are three different normalizations supported: 'equalizeMedians' (default) represents constant normalization (equalizing the medians) based on reference signals is performed. 'quantile' represents quantile normalization based on reference signals. 'globalStandards' represents normalization with global standards proteins. If FALSE, no normalization is performed.
- **nameStandards**
  - optional vector of global standard peptide names. Required only for normalization with global standard peptides.
- **featureSubset**
  - "all" (default) uses all features that the data set has. "top3" uses top 3 features which have highest average of log-intensity across runs. "topN" uses top N features which has highest average of log-intensity across runs. It needs the input for n_top_feature option. "highQuality" flags uninformative feature and outliers.
- **remove_uninformative_feature_outlier**
  - optional. Only required if featureSubset = "highQuality". TRUE allows to remove 1) noisy features (flagged in the column feature_quality with "Uninformative"), 2) outliers (flagged in the column, is_outlier with TRUE, before run-level summarization. FALSE (default) uses all features and intensities for run-level summarization.
min_feature_count
optional. Only required if featureSubset = "highQuality". Defines a minimum number of informative features a protein needs to be considered in the feature selection algorithm.

n_top_feature
optional. Only required if featureSubset = 'topN'. It that case, it specifies number of top features that will be used. Default is 3, which means to use top 3 features.

summaryMethod
"TMP" (default) means Tukey’s median polish, which is robust estimation method. "linear" uses linear mixed model.

equalFeatureVar
only for summaryMethod = "linear". default is TRUE. Logical variable for whether the model should account for heterogeneous variation among intensities from different features. Default is TRUE, which assume equal variance among intensities from features. FALSE means that we cannot assume equal variance among intensities from features, then we will account for heterogeneous variation from different features.

censoredInt
Missing values are censored or at random. 'NA' (default) assumes that all NA's in 'Intensity' column are censored. '0' uses zero intensities as censored intensity. In this case, NA intensities are missing at random. The output from Skyline should use '0'. Null assumes that all NA intensites are randomly missing.

MBimpute
only for summaryMethod = "TMP" and censoredInt = 'NA' or '0'. TRUE (default) imputes NA or '0' (depending on censoredInt option) by Accelerated failure model. FALSE uses the values assigned by cutoffCensored.

remove50missing
only for summaryMethod = "TMP". TRUE removes the runs which have more than 50% missing values. FALSE is default.

fix_missing
Optional, same as the 'fix_missing' parameter in MSstatsConvert::MSstatsBalancedDesign function.

maxQuantileforCensored
Maximum quantile for deciding censored missing values, default is 0.999

use_log_file
logical. If TRUE, information about data processing will be saved to a file.

append
logical. If TRUE, information about data processing will be added to an existing log file.

verbose
logical. If TRUE, information about data processing will be printed to the console.

log_file_path
character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.

Examples

# Consider a raw data (i.e. SRMRawData) for a label-based SRM experiment from a yeast study
# with ten time points (T1-T10) of interests and three biological replicates.
# It is a time course experiment. The goal is to detect protein abundance changes
# across time points.
head(SRMRawData)
# Log2 transformation and normalization are applied (default)
QuantData<-dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)

# Log10 transformation and normalization are applied
QuantData1<-dataProcess(SRMRawData, logTrans=10, use_log_file = FALSE)
head(QuantData1$FeatureLevelData)

# Log2 transformation and no normalization are applied
QuantData2<-dataProcess(SRMRawData, normalization=FALSE, use_log_file = FALSE)
head(QuantData2$FeatureLevelData)

---

dataProcessPlots  Visualization for explanatory data analysis

Description

To illustrate the quantitative data after data-preprocessing and quality control of MS runs, dataProcessPlots takes the quantitative data from function (dataProcess) as input and automatically generate three types of figures in pdf files as output: (1) profile plot (specify "ProfilePlot" in option type), to identify the potential sources of variation for each protein; (2) quality control plot (specify "QCPlot" in option type), to evaluate the systematic bias between MS runs; (3) mean plot for conditions (specify "ConditionPlot" in option type), to illustrate mean and variability of each condition per protein.

Usage

dataProcessPlots(
  data,
  type,
  featureName = "Transition",
  ylimUp = FALSE,
  ylimDown = FALSE,
  scale = FALSE,
  interval = "CI",
  x.axis.size = 10,
  y.axis.size = 10,
  text.size = 4,
  text.angle = 0,
  legend.size = 7,
  dot.size.profile = 2,
  dot.size.condition = 3,
  width = 10,
  height = 10,
  which.Protein = "all",
  originalPlot = TRUE,
  summaryPlot = TRUE,
  save_condition_plot_result = FALSE,
  remove_uninformative_feature_outlier = FALSE,
address = ""
)

Arguments

data name of the (output of dataProcess function) data set.
type choice of visualization. "ProfilePlot" represents profile plot of log intensities across MS runs. "QCPlot" represents quality control plot of log intensities across MS runs. "ConditionPlot" represents mean plot of log ratios (Light/Heavy) across conditions.

featureName for "ProfilePlot" only, "Transition" (default) means printing feature legend in transition-level; "Peptide" means printing feature legend in peptide-level; "NA" means no feature legend printing.

ylimUp upper limit for y-axis in the log scale. FALSE(Default) for Profile Plot and QC Plot use the upper limit as rounded off maximum of log2(intensities) after normalization + 3. FALSE(Default) for Condition Plot is maximum of log ratio + SD or CI.

ylimDown lower limit for y-axis in the log scale. FALSE(Default) for Profile Plot and QC Plot is 0. FALSE(Default) for Condition Plot is minimum of log ratio - SD or CI.

scale for "ConditionPlot" only, FALSE(default) means each conditional level is not scaled at x-axis according to its actual value (equal space at x-axis). TRUE means each conditional level is scaled at x-axis according to its actual value (unequal space at x-axis).

interval for "ConditionPlot" only, "CI"(default) uses confidence interval with 0.95 significant level for the width of error bar. "SD" uses standard deviation for the width of error bar.

x.axis.size size of x-axis labeling for "Run" in Profile Plot and QC Plot, and "Condition" in Condition Plot. Default is 10.

y.axis.size size of y-axis labels. Default is 10.
text.size size of labels represented each condition at the top of graph in Profile Plot and QC plot. Default is 4.
text.angle angle of labels represented each condition at the top of graph in Profile Plot and QC plot or x-axis labeling in Condition plot. Default is 0.

legend.size size of feature legend (transition-level or peptide-level) above graph in Profile Plot. Default is 7.

dot.size.profile size of dots in profile plot. Default is 2.

dot.size.condition size of dots in condition plot. Default is 3.

width width of the saved file. Default is 10.

height height of the saved file. Default is 10.

which.Protein Protein list to draw plots. List can be names of Proteins or order numbers of Proteins from levels(data$FeatureLevelData$PROTEIN). Default is "all", which generates all plots for each protein. For QC plot, "allonly" will generate one QC plot with all proteins.
originalPlot TRUE (default) draws original profile plots.

summaryPlot TRUE (default) draws profile plots with summarization for run levels.

save_condition_plot_result
  TRUE saves the table with values using condition plots. Default is FALSE.

remove_uninformative_feature_outlier
  It only works after users used featureSubset="highQuality" in dataProcess. TRUE allows to remove 1) the features are flagged in the column, feature_quality="Uninformative" which are features with bad quality, 2) outliers that are flagged in the column, is_outlier=TRUE in Profile plots. FALSE (default) shows all features and intensities in profile plots.

address
  the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of "ProfilePlot.pdf" or "QCplot.pdf" or "ConditionPlot.pdf" or "ConditionPlot_value.csv". The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Details

- **Profile Plot**: identify the potential sources of variation of each protein. QuantData$FeatureLevelData is used for plots. X-axis is run. Y-axis is log-intensities of transitions. Reference/endogenous signals are in the left/right panel. Line colors indicate peptides and line types indicate transitions. In summarization plots, gray dots and lines are the same as original profile plots with QuantData$FeatureLevelData. Dark dots and lines are for summarized intensities from QuantData$ProteinLevelData.

- **QC Plot**: illustrate the systematic bias between MS runs. After normalization, the reference signals for all proteins should be stable across MS runs. QuantData$FeatureLevelData is used for plots. X-axis is run. Y-axis is log-intensities of transition. Reference/endogenous signals are in the left/right panel. The pdf file contains (1) QC plot for all proteins and (2) QC plots for each protein separately.

- **Condition Plot**: illustrate the systematic difference between conditions. Summarized intensities from QuantData$ProteinLevelData are used for plots. X-axis is condition. Y-axis is summarized log transformed intensity. If scale is TRUE, the levels of conditions is scaled according to its actual values at x-axis. Red points indicate the mean for each condition. If interval is "CI", blue error bars indicate the confidence interval with 0.95 significant level for each condition. If interval is "SD", blue error bars indicate the standard deviation for each condition. The interval is not related with model-based analysis.

The input of this function is the quantitative data from function dataProcess.

Examples

# Consider quantitative data (i.e. QuantData) from a yeast study with ten time points of interests, three biological replicates, and no technical replicates which is a time-course experiment.
# The goal is to provide pre-analysis visualization by automatically generate two types of figures in two separate pdf files.
# Protein IDHC (gene name IDP2) is differentially expressed in time point 1 and time point 7,
```r
# whereas, Protein PMG2 (gene name GPM2) is not.

QuantData<-dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
# Profile plot
dataProcessPlots(data=QuantData,type="ProfilePlot")
# Quality control plot
dataProcessPlots(data=QuantData,type="QCPlot")
# Quantification plot for conditions
dataProcessPlots(data=QuantData,type="ConditionPlot")
```

---

**DDARawData**

*Example dataset from a label-free DDA, a controlled spike-in experiment.*

**Description**

This is a data set obtained from a published study (Mueller, et. al, 2007). A controlled spike-in experiment, where 6 proteins, (horse myoglobin, bovine carbonic anhydrase, horse Cytochrome C, chicken lysozyme, yeast alcohol dehydrogenase, rabbit aldolase A) were spiked into a complex background in known concentrations in a latin square design. The experiment contained 6 mixtures, and each mixture was analyzed in label-free LC-MS mode with 3 technical replicates (resulting in the total of 18 runs). Each protein was represented by 7-21 peptides, and each peptide was represented by 1-5 transition.

**Usage**

DDARawData

**Format**

data.frame

**Details**

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed.

If the information of one or more columns is not available for the original raw data, please retain the column variables and type in fixed value. For example, the original raw data does not contain the information of PrecursorCharge and ProductCharge, we retain the column PrecursorCharge and ProductCharge and then type in NA for all transitions in RawData.

Variable Intensity is required to be original signal without any log transformation and can be specified as the peak of height or the peak of area under curve.

**Value**

data.frame with the required format of MSstats.


**Author(s)**

Meena Choi, Olga Vitek.

Maintainer: Meena Choi (<mnchoi67@gmail.com>)

**References**


**Examples**

```r
head(DDARawData)
```

**Description**

This is a data set obtained from a published study (Mueller, et. al, 2007). A controlled spike-in experiment, where 6 proteins, (horse myoglobin, bovine carbonic anhydrase, horse Cytochrome C, chicken lysozyme, yeast alcohol dehydrogenase, rabbit aldolase A) were spiked into a complex background in known concentrations in a latin square design. The experiment contained 6 mixtures, and each mixture was analyzed in label-free LC-MS mode with 3 technical replicates (resulting in the total of 18 runs). Each protein was represented by 7-21 peptides, and each peptide was represented by 1-5 transition. Skyline is used for processing.

**Usage**

```r
DDARawData.Skyline
```

**Format**

`data.frame`
**Details**

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed.

This is 'MSstats input' format from Skyline used by 'MSstats_report.skyr'. The column names, 'FileName' and 'Area', should be changed to 'Run' and 'Intensity'. There are two extra columns called 'StandardType' and 'Truncated'. 'StandardType' column can be used for normalization='globalStandard' in `dataProcess`. 'Truncated' columns can be used to remove the truncated peaks with `skylineReport=TRUE` in `dataProcess`.

If the information of one or more columns is not available for the original raw data, please retain the column variables and type in fixed value. For example, the original raw data does not contain the information of PrecursorCharge and ProductCharge, we retain the column PrecursorCharge and ProductCharge and then type in NA for all transitions in RawData.

Variable Intensity is required to be original signal without any log transformation and can be specified as the peak of height or the peak of area under curve.

**Value**

data.frame with the required format of MSstats.

**Author(s)**

Meena Choi, Olga Vitek.

Maintainer: Meena Choi (<mnchoi67@gmail.com>)

**References**


**Examples**

```
head(DDARawData.Skyline)
```
designSampleSize

Description

Calculate sample size for future experiments of a Selected Reaction Monitoring (SRM), Data-Dependent Acquisition (DDA or shotgun), and Data-Independent Acquisition (DIA or SWATH-MS) experiment based on intensity-based linear model. Two options of the calculation: (1) number of biological replicates per condition, (2) power.

Usage

designSampleSize(
  data, 
  desiredFC, 
  FDR = 0.05, 
  numSample = TRUE, 
  power = 0.9, 
  use_log_file = TRUE, 
  append = FALSE, 
  verbose = TRUE, 
  log_file_path = NULL 
)

Arguments

data 'FittedModel' in testing output from function groupComparison.
desiredFC the range of a desired fold change which includes the lower and upper values of the desired fold change.
FDR a pre-specified false discovery ratio (FDR) to control the overall false positive rate. Default is 0.05
numSample minimal number of biological replicates per condition. TRUE represents you require to calculate the sample size for this category, else you should input the exact number of biological replicates.
power a pre-specified statistical power which defined as the probability of detecting a true fold change. TRUE represent you require to calculate the power for this category, else you should input the average of power you expect. Default is 0.9
use_log_file logical. If TRUE, information about data processing will be saved to a file.
append logical. If TRUE, information about data processing will be added to an existing log file.
verbose logical. If TRUE, information about data processing will be printed to the console.
log_file_path character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

Details

The function fits the model and uses variance components to calculate sample size. The underlying model fitting with intensity-based linear model with technical MS run replication. Estimated sample
size is rounded to 0 decimal. The function can only obtain either one of the categories of the sample size calculation (numSample, numPep, numTran, power) at the same time.

Value
data.frame - sample size calculation results including variables: desiredFC, numSample, FDR, and power.

Author(s)
Meena Choi, Ching-Yun Chang, Olga Vitek.

Examples

# Consider quantitative data (i.e. QuantData) from yeast study.
# A time course study with ten time points of interests and three biological replicates.
QuantData <- dataProcess(SRMRawData)
head(QuantData$FeatureLevelData)
## based on multiple comparisons (T1 vs T3; T1 vs T7; T1 vs T9)
comparison1<-matrix(c(-1,0,1,0,0,0,0,0,0,0),nrow=1)
comparison2<-matrix(c(-1,0,0,0,0,1,0,0,0,0),nrow=1)
comparison3<-matrix(c(-1,0,0,0,0,0,0,1,0,0),nrow=1)
comparison<-rbind(comparison1,comparison2, comparison3)
row.names(comparison)<-c("T3-T1","T7-T1","T9-T1")
colnames(comparison)<-unique(QuantData$ProteinLevelData$GROUP)

testResultMultiComparisons<-groupComparison(contrast.matrix=comparison,data=QuantData)
## Calculate sample size for future experiments:
#(1) Minimal number of biological replicates per condition
designSampleSize(data=testResultMultiComparisons$FittedModel, numSample=TRUE,
  desiredFC=c(1.25,1.75), FDR=0.05, power=0.8)
#(2) Power calculation
designSampleSize(data=testResultMultiComparisons$FittedModel, numSample=2,
  desiredFC=c(1.25,1.75), FDR=0.05, power=TRUE)

designSampleSizePlots

Description
To illustrate the relationship of desired fold change and the calculated minimal number sample size which are (1) number of biological replicates per condition, (2) number of peptides per protein, (3) number of transitions per peptide, and (4) power. The input is the result from function designSampleSize.

Usage
designSampleSizePlots(data)
**Arguments**

- **data**
  
  Output from function `designSampleSize`.

**Details**

Data in the example is based on the results of sample size calculation from function `designSampleSize`.

**Value**

Plot for estimated sample size with assigned variable.

**Author(s)**

Meena Choi, Ching-Yun Chang, Olga Vitek.

**Examples**

```r
# Based on the results of sample size calculation from function designSampleSize,
# we generate a series of sample size plots for number of biological replicates, or peptides,
# or transitions or power plot.
QuantData<-dataProcess(SRMRawData)
head(QuantData$ProcessedData)
## based on multiple comparisons (T1 vs T3; T1 vs T7; T1 vs T9)
comparison1<-matrix(c(-1,0,1,0,0,0,0,0,0,0),nrow=1)
comparison2<-matrix(c(-1,0,0,0,0,1,0,0,0,0),nrow=1)
comparison3<-matrix(c(-1,0,0,0,0,0,0,1,0,0),nrow=1)
comparison<-rbind(comparison1,comparison2, comparison3)
row.names(comparison)<-c("T3-T1","T7-T1","T9-T1")
colnames(comparison)<-unique(QuantData$ProteinLevelData$GROUP)

testResultMultiComparisons<-groupComparison(contrast.matrix=comparison, data=QuantData)
# plot the calculated sample sizes for future experiments:
# (1) Minimal number of biological replicates per condition
result.sample<-designSampleSize(data=testResultMultiComparisons$FittedModel, numSample=TRUE, 
  desiredFC=c(1.25,1.75), FDR=0.05, power=0.8)
designSampleSizePlots(data=result.sample)
# (2) Power
result.power<-designSampleSize(data=testResultMultiComparisons$FittedModel, numSample=2, 
  desiredFC=c(1.25,1.75), FDR=0.05, power=TRUE)
designSampleSizePlots(data=result.power)
```

**Description**

Import Diann files
Usage

DIANNtoMSstatsFormat(
  input,
  annotation = NULL,
  global_qvalue_cutoff = 0.01,
  qvalue_cutoff = 0.01,
  pg_qvalue_cutoff = 0.01,
  useUniquePeptide = TRUE,
  removeFewMeasurements = TRUE,
  removeOxidationMpeptides = TRUE,
  removeProtein_with1Feature = TRUE,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  MBR = TRUE,
  ...
)

Arguments

  input           name of MSstats input report from Diann, which includes feature-level data.
  annotation      name of 'annotation.txt' data which includes Condition, BioReplicate, Run.
  global_qvalue_cutoff
                        The global qvalue cutoff
  qvalue_cutoff    local qvalue cutoff for library
  pg_qvalue_cutoff local qvalue cutoff for protein groups Run should be the same as filename.
  useUniquePeptide should unique peptides be removed
  removeFewMeasurements should proteins with few measurements be removed
  removeOxidationMpeptides should peptides with oxidation be removed
  removeProtein_with1Feature should proteins with a single feature be removed
  use_log_file     logical. If TRUE, information about data processing will be saved to a file.
  append           logical. If TRUE, information about data processing will be added to an existing log file.
  verbose          logical. If TRUE, information about data processing will be printed to the console.
  log_file_path    character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.
  MBR              True if analysis was done with match between runs
  ...              additional parameters to ‘data.table::fread’.
Value

data.frame in the MSstats required format.

Author(s)

Elijah Willie

Examples

## Not run:
input = fread('diann_pooled_report.tsv')
annot = fread('Annotation.csv')
colnames(annot) = c('Condition', 'Run', 'BioReplicate')
input = DIANNtoMSstatsFormat(input, annotation = annot, MBR = F)
head(input)

## End(Not run)

---

**DIARawData**  
Example dataset from a label-free DIA, a group comparison study of *S. Pyogenes*.

Description

This example dataset was obtained from a group comparison study of S. Pyogenes. Two conditions, *S. Pyogenes* with 0% and 10% of human plasma added (denoted Strep 0% and Strep 10%), were profiled in two replicates, in the label-free mode, with a SWATH-MS-enabled AB SCIEX TripleTOF 5600 System. The identification and quantification of spectral peaks was assisted by a spectral library, and was performed using OpenSWATH software (http://proteomics.ethz.ch/openswath.html). For reasons of space, the example dataset only contains two proteins from this study. Protein FabG shows strong evidence of differential abundance, while protein Probable RNA helicase exp9 only shows moderate evidence of differential abundance between conditions.

Usage

DIARawData

Format

data.frame

Details

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed.

If the information of one or more columns is not available for the original raw data, please retain the column variables and type in fixed value. For example, the original raw data does not contain
the information of PrecursorCharge and ProductCharge, we retain the column PrecursorCharge and ProductCharge and then type in NA for all transitions in RawData.

Variable Intensity is required to be original signal without any log transformation and can be specified as the peak of height or the peak of area under curve.

Value
data.frame with the required format of MSstats.

Author(s)
Meena Choi, Olga Vitek.
Maintainer: Meena Choi (<mchoi67@gmail.com>)

Examples
head(DIARawData)
DIAUmpiretoMSstatsFormat

Arguments

- `raw.frag`: name of FragSummary_date.xls data, which includes feature-level data.
- `raw.pep`: name of PeptideSummary_date.xls data, which includes selected fragments information.
- `raw.pro`: name of ProteinSummary_date.xls data, which includes selected peptides information.
- `annotation`: name of annotation data which includes Condition, BioReplicate, Run information.
- `useSelectedFrag`: TRUE will use the selected fragment for each peptide. 'Selected_fragments' column is required.
- `useSelectedPep`: TRUE will use the selected peptide for each protein. 'Selected_peptides' column is required.
- `removeFewMeasurements`: TRUE (default) will remove the features that have 1 or 2 measurements across runs.
- `removeProtein_with1Feature`: TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.
- `summaryforMultipleRows`: max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.
- `use_log_file`: logical. If TRUE, information about data processing will be saved to a file.
- `append`: logical. If TRUE, information about data processing will be added to an existing log file.
- `verbose`: logical. If TRUE, information about data processing will be printed to the console.
- `log_file_path`: character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.
- `...`: additional parameters to 'data.table::.fread'.

Value

data.frame in the MSstats required format.

Author(s)

Meena Choi, Olga Vitek

Examples

diau_frag = system.file("tinytest/raw_data/DIAUmpire/dia_frag.csv", package = "MSstatsConvert")
diau_pept = system.file("tinytest/raw_data/DIAUmpire/dia_pept.csv", package = "MSstatsConvert")
extractSDRF

**Extract experimental design from MSstats format into SDRF format**

**Description**

Extract experimental design from MSstats format into SDRF format

**Usage**

```r
extractSDRF(
  data,
  run_name = "comment[data file]",
  condition_name = "characteristics[disease]",
  biological_replicate = "characteristics[biological replicate]",
  fraction = NULL,
  meta_data = NULL
)
```

**Arguments**

- **data** MSstats formatted data that is the output of a dedicated converter, such as `MaxQtoMSstatsFormat`, `SkylinetoMSstatsFormat`, etc.
- **run_name** Run column name in SDRF data
- **condition_name** Condition column name in SDRF data
- **biological_replicate** Biological replicate column name in SDRF data
- **fraction** Fraction column name in SDRF data (if applicable). Default is `NULL`. If there are no fractions keep `NULL`.
- **meta_data** A data.frame including any additional meta data for the SDRF file that is not included in MSstats. This meta data will be added into the final SDRF file. Please ensure the run names in the meta data matches the run names in the MSstats data.
**FragPipetoMSstatsFormat**

*Import FragPipe files*

**Description**

Import FragPipe files

**Usage**

FragPipetoMSstatsFormat(
  input,
  useUniquePeptide = TRUE,
  removeFewMeasurements = TRUE,
  removeProtein_with1Feature = FALSE,
  summaryforMultipleRows = max,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

**Arguments**

- **input**: name of FragPipe msstats.csv export. ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity are required.

- **useUniquePeptide**: TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

- **removeFewMeasurements**: TRUE (default) will remove the features that have 1 or 2 measurements across runs.

**Examples**

```r
mq_ev = data.table::fread(system.file("tinytest/raw_data/MaxQuant/mq_ev.csv", package = "MSstatsConvert"))
mq_pg = data.table::fread(system.file("tinytest/raw_data/MaxQuant/mq_pg.csv", package = "MSstatsConvert"))
annot = data.table::fread(system.file("tinytest/raw_data/MaxQuant/annotation.csv", package = "MSstatsConvert"))
maxq_imported = MaxQtoMSstatsFormat(mq_ev, annot, mq_pg, use_log_file = FALSE)
head(maxq_imported)
SDRF_file = extractSDRF(maxq_imported)
```
getProcessed

removeProtein_with1Feature
TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.

summaryforMultipleRows
max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.

use_log_file
logical. If TRUE, information about data processing will be saved to a file.

append
logical. If TRUE, information about data processing will be added to an existing log file.

verbose
logical. If TRUE, information about data processing wil be printed to the console.

log_file_path
character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

additional parameters to ‘data.table::fread’.

Value

data.frame in the MSstats required format.

Author(s)

Devon Kohler

Examples

fragpipe_raw = system.file("tinytest/raw_data/FragPipe/fragpipe_input.csv", package = "MSstatsConvert")
fragpipe_raw = data.table::fread(fragpipe_raw)
fragpipe_imported = FragPipetoMSstatsFormat(fragpipe_raw, use_log_file = FALSE)
head(fragpipe_imported)

generated table

getProcessed

Description

Get feature-level data to be used in the MSstatsSummarizationOutput function

Usage

generated table

Arguments

input
data.table processed by dataProcess subfunctions
getSamplesInfo

Value

data.table processed by dataProcess subfunctions

Examples

raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANS")
input = MSstatsMergeFractions(input)
input = MSstatsHandleMissing(input, "TMP", TRUE, "NA", 0.999)
input.all = MSstatsSelectFeatures(input, "all") # all features
input.5 = MSstatsSelectFeatures(data.table::copy(input),
  "topN", top_n = 5) # top 5 features

proc1 = getProcessed(input.all)
proc2 = getProcessed(input.5)

proc1
proc2

getSamplesInfo

Get information about number of measurements for each group

Description

Get information about number of measurements for each group

Usage

getSamplesInfo(summarization_output)

Arguments

summarization_output
  output of the dataProcess function

Details

This function extracts information required to compute percentages of missing and imputed values in group comparison.

Value

data.table
Examples

QuantData <- dataProcess(DDARawData, use_log_file = FALSE)
samples_info <- getSamplesInfo(QuantData)
samples_info

getSelectedProteins  Get proteins based on names or integer IDs

Description

Get proteins based on names or integer IDs

Usage

getSelectedProteins(chosen_proteins, all_proteins)

Arguments

chosen_proteins  protein names or integers IDs
all_proteins all unique proteins

Value

character

groupComparison  Whole plot testing

Description

Whole plot testing

Usage

groupComparison(
  contrast.matrix,
  data,
  save_fitted_models = TRUE,
  log_base = 2,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL
)
Arguments

contrast.matrix  
comparison between conditions of interests.

data  
name of the (output of dataProcess function) data set.

save_fitted_models  
logical, if TRUE, fitted models will be added to the output.

log_base  
base of the logarithm used in dataProcess.

use_log_file  
logical. If TRUE, information about data processing will be saved to a file.

append  
logical. If TRUE, information about data processing will be added to an existing log file.

verbose  
logical. If TRUE, information about data processing will be printed to the console.

log_file_path  
character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

Details

contrast.matrix : comparison of interest. Based on the levels of conditions, specify 1 or -1 to the conditions of interests and 0 otherwise. The levels of conditions are sorted alphabetically. Command levels(QuantData$FeatureLevelData$GROUP) can illustrate the actual order of the levels of conditions. The underlying model fitting functions are lm and lmer for the fixed effects model and mixed effects model, respectively. The input of this function is the quantitative data from function (dataProcess).

Value

list that consists of three elements: "ComparisonResult" - data.frame with results of statistical testing, "ModelQC" - data.frame with data used to fit models for group comparison and "FittedModel" - list of fitted models.

Examples

# Consider quantitative data (i.e. QuantData) from yeast study with ten time points of interests,  
# three biological replicates, and no technical replicates.  
# It is a time-course experiment and we attempt to compare differential abundance  
# between time 1 and 7 in a set of targeted proteins.  
# In this label-based SRM experiment, MSstats uses the fitted model with expanded scope of  
# Biological replication.
QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
levels(QuantData$ProteinLevelData$GROUP)
comparision <- matrix(c(-1,0,0,0,0,0,1,0,0,0),nrow=1)
row.names(comparision) <- "T7-T1"
groups = levels(QuantData$ProteinLevelData$GROUP)
colnames(comparision) <- groups[order(as.numeric(groups))]
# Tests for differentially abundant proteins with models:  
# label-based SRM experiment with expanded scope of biological replication.
testResultOneComparison <- groupComparison(contrast.matrix=comparison, data=QuantData,
use_log_file = FALSE)

# table for result
testResultOneComparison$ComparisonResult

groupComparisonPlots <- Visualization for model-based analysis and summarizing differentially abundant proteins

Description

To summarize the results of log-fold changes and adjusted p-values for differentially abundant proteins, groupComparisonPlots takes testing results from function (groupComparison) as input and automatically generate three types of figures in pdf files as output: (1) volcano plot (specify "VolcanoPlot" in option type) for each comparison separately; (2) heatmap (specify "Heatmap" in option type) for multiple comparisons; (3) comparison plot (specify "ComparisonPlot" in option type) for multiple comparisons per protein.

Usage

groupComparisonPlots(
data,
type,
sig = 0.05,
FCcutoff = FALSE,
logBase.pvalue = 10,
ylimUp = FALSE,
ylimDown = FALSE,
xlimUp = FALSE,
xaxis.size = 10,
yaxis.size = 10,
dot.size = 3,
text.size = 4,
text.angle = 0,
legend.size = 13,
ProteinName = TRUE,
colorkey = TRUE,
umProtein = 100,
clustering = "both",
which.Comparison = "all",
which.Protein = "all",
address = ""
)
**Arguments**

- **data**
  - 'ComparisonResult' in testing output from function groupComparison.

- **type**
  - choice of visualization. "VolcanoPlot" represents volcano plot of log fold changes and adjusted p-values for each comparison separately. "Heatmap" represents heatmap of adjusted p-values for multiple comparisons. "ComparisonPlot" represents comparison plot of log fold changes for multiple comparisons per protein.

- **sig**
  - FDR cutoff for the adjusted p-values in heatmap and volcano plot. level of significance for comparison plot. 100(1-sig)% confidence interval will be drawn. sig=0.05 is default.

- **FCcutoff**
  - for volcano plot or heatmap, whether involve fold change cutoff or not. FALSE (default) means no fold change cutoff is applied for significance analysis. FC-cutoff = specific value means specific fold change cutoff is applied.

- **logBase.pvalue**
  - for volcano plot or heatmap, (-) logarithm transformation of adjusted p-value with base 2 or 10 (default).

- **ylimUp**
  - for all three plots, upper limit for y-axis. FALSE (default) for volcano plot/heatmap use maximum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses maximum of log-fold change + CI.

- **ylimDown**
  - for all three plots, lower limit for y-axis. FALSE (default) for volcano plot/heatmap use minimum of -log2 (adjusted p-value) or -log10 (adjusted p-value). FALSE (default) for comparison plot uses minimum of log-fold change - CI.

- **xlimUp**
  - for Volcano plot, the limit for x-axis. FALSE (default) for use maximum for absolute value of log-fold change or 3 as default if maximum for absolute value of log-fold change is less than 3.

- **x.axis.size**
  - size of axes labels, e.g. name of the comparisons in heatmap, and in comparison plot. Default is 10.

- **y.axis.size**
  - size of axes labels, e.g. name of targeted proteins in heatmap. Default is 10.

- **dot.size**
  - size of dots in volcano plot and comparison plot. Default is 3.

- **text.size**
  - size of ProteinName label in the graph for Volcano Plot. Default is 4.

- **text.angle**
  - angle of x-axis labels represented each comparison at the bottom of graph in comparison plot. Default is 0.

- **legend.size**
  - size of legend for color at the bottom of volcano plot. Default is 7.

- **ProteinName**
  - for volcano plot only, whether display protein names or not. TRUE (default) means protein names, which are significant, are displayed next to the points. FALSE means no protein names are displayed.

- **colorkey**
  - TRUE (default) shows colorkey.

- **numProtein**
  - The number of proteins which will be presented in each heatmap. Default is 100. Maximum possible number of protein for one heatmap is 180.

- **clustering**
  - Determines how to order proteins and comparisons. Hierarchical cluster analysis with Ward method (minimum variance) is performed. 'protein' means that protein dendrogram is computed and reordered based on protein means (the order of row is changed). 'comparison' means comparison dendrogram is computed and reordered based on comparison means (the order of comparison is
groupComparisonPlots

changed). 'both' means to reorder both protein and comparison. Default is 'protein'.

width width of the saved file. Default is 10.

height height of the saved file. Default is 10.

which.Comparison list of comparisons to draw plots. List can be labels of comparisons or order numbers of comparisons from levels(data$Label), such as levels(testResultMultiComparisons$ComparisonResult$Label). Default is "all", which generates all plots for each protein.

which.Protein Protein list to draw comparison plots. List can be names of Proteins or order numbers of Proteins from levels(testResultMultiComparisons$ComparisonResult$Protein). Default is "all", which generates all comparison plots for each protein.

address the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of "VolcanoPlot.pdf" or "Heatmap.pdf" or "ComparisonPlot.pdf". The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Details

• Volcano plot: illustrate actual log-fold changes and adjusted p-values for each comparison separately with all proteins. The x-axis is the log fold change. The base of logarithm transformation is the same as specified in "logTrans" from dataProcess. The y-axis is the negative log2 or log10 adjusted p-values. The horizontal dashed line represents the FDR cutoff. The points below the FDR cutoff line are non-significantly abundant proteins (colored in black). The points above the FDR cutoff line are significantly abundant proteins (colored in red/blue for up-/down-regulated). If fold change cutoff is specified (FCcutoff = specific value), the points above the FDR cutoff line but within the FC cutoff line are non-significantly abundant proteins (colored in black)/

• Heatmap: illustrate up-/down-regulated proteins for multiple comparisons with all proteins. Each column represents each comparison of interest. Each row represents each protein. Color red/blue represents proteins in that specific comparison are significantly up-regulated/down-regulated proteins with FDR cutoff and/or FC cutoff. The color scheme shows the evidences of significance. The darker color it is, the stronger evidence of significance it has. Color gold represents proteins are not significantly different in abundance.

• Comparison plot: illustrate log-fold change and its variation of multiple comparisons for single protein. X-axis is comparison of interest. Y-axis is the log fold change. The red points are the estimated log fold change from the model. The blue error bars are the confidence interval with 0.95 significant level for log fold change. This interval is only based on the standard error, which is estimated from the model.

Examples

QuantData<-dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
## based on multiple comparisons (T1 vs T3; T1 vs T7; T1 vs T9)
**makePeptidesDictionary**

Prepare a peptides dictionary for global standards normalization

**Description**

Prepare a peptides dictionary for global standards normalization

**Usage**

```r
makePeptidesDictionary(input, normalization)
```

**Arguments**

- **input** `‘data.table’` in MSstats standard format
- **normalization** normalization method
Details

This function extracts information required to perform normalization with global standards. It is useful for running the summarization workflow outside of the dataProcess function.

Examples

```r
input = data.table::as.data.table(ddArData)
peptides_dict = makePeptidesDictionary(input, "GLOBALSTANDARDS")
head(peptides_dict) # ready to be passed to the MSstatsNormalize function
```

MaxQtoMSstatsFormat

Import MaxQuant files

Description

Import MaxQuant files

Usage

```r
MaxQtoMSstatsFormat(
  evidence,
  annotation,
  proteinGroups,
  proteinID = "Proteins",
  useUniquePeptide = TRUE,
  summaryforMultipleRows = max,
  removeFewMeasurements = TRUE,
  removeMpeptides = FALSE,
  removeOxidationMpeptides = FALSE,
  removeProtein_with1Peptide = FALSE,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)
```

Arguments

- **evidence**: name of 'evidence.txt' data, which includes feature-level data.
- **annotation**: name of 'annotation.txt' data which includes Raw.file, Condition, BioReplicate, Run, IsotopeLabelType information.
- **proteinGroups**: name of 'proteinGroups.txt' data. It needs to matching protein group ID. If proteinGroups=NULL, use 'Proteins' column in 'evidence.txt'.
- **proteinID**: 'Proteins'(default) or 'Leading.razor.protein' for Protein ID.
MaxQtoMSstatsFormat

useUniquePeptide
TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

summaryforMultipleRows
max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.

removeFewMeasurements
TRUE (default) will remove the features that have 1 or 2 measurements across runs.

removeMpeptides
TRUE will remove the peptides including 'M' sequence. FALSE is default.

removeOxidationMpeptides
TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.

removeProtein_with1Peptide
TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.

use_log_file
logical. If TRUE, information about data processing will be saved to a file.

append
logical. If TRUE, information about data processing will be added to an existing log file.

verbose
logical. If TRUE, information about data processing will be printed to the console.

log_file_path
character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.

... additional parameters to 'data.table::fread'.

Value
data.frame in the MSstats required format.

Note
Warning: MSstats does not support for metabolic labeling or iTRAQ experiments.

Author(s)
Meena Choi, Olga Vitek.

Examples
mq_ev = data.table::fread(system.file("tinytest/raw_data/MaxQuant/mq_ev.csv", package = "MSstatsConvert"))
mq_pg = data.table::fread(system.file("tinytest/raw_data/MaxQuant/mq_pg.csv", package = "MSstatsConvert"))
annot = data.table::fread(system.file("tinytest/raw_data/MaxQuant/annotation.csv", package = "MSstatsConvert"))
maxq_imported = MaxQtoMSstatsFormat(mq_ev, annot, mq_pg, use_log_file = FALSE)
modelBasedQCPlots

Visualization for model-based quality control in fitting model

Description

To check the assumption of linear model for whole plot inference, modelBasedQCPlots takes the results after fitting models from function (groupComparison) as input and automatically generate two types of figures in pdf files as output: (1) normal quantile-quantile plot (specify "QQPlot" in option type) for checking normally distributed errors.; (2) residual plot (specify "ResidualPlot" in option type).

Usage

modelBasedQCPlots(
  data,
  type,
  axis.size = 10,
  dot.size = 3,
  width = 10,
  height = 10,
  which.Protein = "all",
  address = ""
)

Arguments

data output from function groupComparison.
type choice of visualization. "QQPlots" represents normal quantile-quantile plot for each protein after fitting models. "ResidualPlots" represents a plot of residuals versus fitted values for each protein in the dataset.
axis.size size of axes labels. Default is 10.
dot.size size of points in the graph for residual plots and QQ plots. Default is 3.
width width of the saved file. Default is 10.
height height of the saved file. Default is 10.
which.Protein Protein list to draw plots. List can be names of Proteins or order numbers of Proteins from levels(testResultOneComparison$ComparisonResult$Protein). Default is "all", which generates all plots for each protein.
address name that will serve as a prefix to the name of output file.
Results based on statistical models for whole plot level inference are accurate as long as the assumptions of the model are met. The model assumes that the measurement errors are normally distributed with mean 0 and constant variance. The assumption of a constant variance can be checked by examining the residuals from the model.

- **QQPlots**: A normal quantile-quantile plot for each protein is generated in order to check whether the errors are well approximated by a normal distribution. If points fall approximately along a straight line, then the assumption is appropriate for that protein. Only large deviations from the line are problematic.

- **ResidualPlots**: The plots of residuals against predicted(fitted) values. If it shows a random scatter, then the assumption is appropriate.

**Value**

produce a pdf file

**Examples**

```
QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
levels(QuantData$FeatureLevelData$GROUP)
comparison <- matrix(c(-1,0,0,0,0,0,1,0,0,0),nrow=1)
row.names(comparison) <- "T7-T1"
colnames(comparison) <- unique(QuantData$ProteinLevelData$GROUP)
# Tests for differentially abundant proteins with models:
# label-based SRM experiment with expanded scope of biological replication.
testResultOneComparison <- groupComparison(contrast.matrix=comparison, data=QuantData,
   use_log_file = FALSE)
# normal quantile-quantile plots
modelBasedQCPlots(data=testResultOneComparison, type="QQPlots", address="")
# residual plots
modelBasedQCPlots(data=testResultOneComparison, type="ResidualPlots", address="")
```
Arguments

contrasts One of the following: i) list of lists. Each sub-list consists of two vectors that name conditions that will be compared. See the details section for more information ii) matrix. In this case, it's correctness will be checked iii) "pairwise". In this case, pairwise comparison matrix will be generated iv) data.frame. In this case, input will be converted to matrix

conditions unique condition labels

labels labels for contrasts (row.names of the contrast matrix)

Description

Group comparison

Usage

MSstatsGroupComparison(
  summarized_list,
  contrast_matrix,
  save_fitted_models,
  repeated,
  samples_info
)

Arguments

summarized_list output of MSstatsPrepareForGroupComparison

contrast_matrix contrast matrix

save_fitted_models if TRUE, fitted models will be included in the output

repeated logical, output of checkRepeatedDesign function

samples_info data.table, output of getSamplesInfo function

Examples

QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
group_comparison_input = MSstatsPrepareForGroupComparison(QuantData)
levels(QuantData$ProteinLevelData$GROUP)
comparison <- matrix(c(-1,0,0,0,0,1,0,0,0),nrow=1)
row.names(comparison) <- "T7-T1"
groups = levels(QuantData$ProteinLevelData$GROUP)
MSstatsGroupComparisonOutput

Create output of group comparison based on results for individual proteins

Description

Create output of group comparison based on results for individual proteins

Usage

MSstatsGroupComparisonOutput(input, summarization_output, log_base = 2)

Arguments

input output of MSstatsGroupComparison function
summarization_output output of dataProcess function
log_base base of the logarithm used in fold-change calculation

Value

list, same as the output of ‘groupComparison’

Examples

QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
group_comparison_input = MSstatsPrepareForGroupComparison(QuantData)
levels(QuantData$ProteinLevelData$GROUP)
comparison <- matrix(c(-1,0,0,0,0,0,1,0,0,0),nrow=1)
row.names(comparison) <- "T7-T1"
groups = levels(QuantData$ProteinLevelData$GROUP)
colnames(comparison) <- groups[order(as.numeric(groups))]
samples_info = getSamplesInfo(QuantData)
repeated = checkRepeatedDesign(QuantData)
group_comparison = MSstatsGroupComparison(group_comparison_input, comparison,
FALSE, repeated, samples_info)
group_comparison_final = MSstatsGroupComparisonOutput(group_comparison,
Description

Group comparison for a single protein

Usage

MSstatsGroupComparisonSingleProtein(
  single_protein,
  contrast_matrix,
  repeated,
  groups,
  samples_info,
  save_fitted_models,
  has_imputed
)

Arguments

single_protein  data.table with summarized data for a single protein
contrast_matrix  contrast matrix
repeated         if TRUE, repeated measurements will be modeled
groups          unique labels of experimental conditions
samples_info    number of runs per group
save_fitted_models  if TRUE, fitted model will be saved. If not, it will be replaced with NULL
has_imputed     TRUE if missing values have been imputed

Examples

QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
group_comparison_input <- MSstatsPrepareForGroupComparison(QuantData)
levels(QuantData$ProteinLevelData$GROUP)
comparison <- matrix(c(-1,0,0,0,0,0,1,0,0,0),nrow=1)
row.names(comparison) <- "T7-T1"
groups = levels(QuantData$ProteinLevelData$GROUP)
colnames(comparison) <- groups[order(as.numeric(groups))]
samples_info <- getSamplesInfo(QuantData)
repeated <- checkRepeatedDesign(QuantData)
single_output <- MSstatsGroupComparisonSingleProtein(
  group_comparison_input[[1]], comparison, repeated, groups, samples_info,
  FALSE, TRUE)
single_output # same as a single element of MSstatsGroupComparison output

---

MSstatsHandleMissing  Handle censored missing values

Description

Handle censored missing values

Usage

MSstatsHandleMissing(
  input,
  summary_method,
  impute,
  missing_symbol,
  censored_cutoff
)

Arguments

- **input**: `data.table` in MSstats data format
- **summary_method**: summarization method (`summaryMethod` parameter to `dataProcess`)
- **impute**: if TRUE, missing values are supposed to be imputed (`MBimpute` parameter to `dataProcess`)
- **missing_symbol**: `censoredInt` parameter to `dataProcess`
- **censored_cutoff**: `maxQuantileforCensored` parameter to `dataProcess`

Value

data.table

Examples

```r
raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANs")
input = MSstatsMergeFractions(input)
```
MSstatsNormalize

Normalize MS data

Description

Normalize MS data

Usage

MSstatsNormalize(input)

Arguments

input 'data.table' in MSstats format

Examples

raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANS")
head(input)

MSstatsMergeFractions  Re-format the data before feature selection

Description

Re-format the data before feature selection

Usage

MSstatsMergeFractions(input)

Arguments

input 'data.table' in MSstats format

Value
data.table

Examples

raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANS")
input = MSstatsMergeFractions(input)
head(input)
**Usage**

```r
MSstatsNormalize(
    input,
    normalization_method,
    peptides_dict = NULL,
    standards = NULL
)
```

**Arguments**

- **input**: data.table in MSstats format
- **normalization_method**: name of a chosen normalization method: "NONE" or "FALSE" for no normalization, "EQUALIZEMEDIANS" for median normalization, "QUANTILE" normalization for quantile normalization from 'preprocessCore' package, "GLOBALSTANDARDS" for normalization based on selected peptides or proteins.
- **peptides_dict**: `data.table` of names of peptides and their corresponding features.
- **standards**: character vector with names of standards, required if "GLOBALSTANDARDS" method was selected.

**Value**

data.table

**Examples**

```r
raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANS") # median normalization
head(input)
```

---

**MSstatsPrepareForDataProcess**

*Prepare data for processing by 'dataProcess' function*

**Description**

Prepare data for processing by ‘dataProcess’ function

**Usage**

```r
MSstatsPrepareForDataProcess(input, log_base, fix_missing)
```
MSstatsPrepareForGroupComparison

Prepare output for dataProcess for group comparison

Description

Prepare output for dataProcess for group comparison

Usage

MSstatsPrepareForGroupComparison(summarization_output)

Arguments

summarization_output
output of dataProcess

Value

list of run-level data for each protein in the input. This list has a "has_imputed" attribute that indicates if missing values were imputed in the input dataset.
Examples

QuantData <- dataProcess(SRMRawData, use_log_file = FALSE)
group_comparison_input = MSstatsPrepareForGroupComparison(QuantData)
length(group_comparison_input) # list of length equal to number of proteins
# in protein-level data of QuantData
head(group_comparison_input[[1]])

MSstatsPrepareForSummarization

Prepare feature-level data for protein-level summarization

Description

Prepare feature-level data for protein-level summarization

Usage

MSstatsPrepareForSummarization(
  input,
  method,
  impute,
  censored_symbol,
  remove_uninformative_feature_outlier
)

Arguments

input feature-level data processed by dataProcess subfunctions
method summarization method - ‘summaryMethod’ parameter of the dataProcess function
impute if TRUE, censored missing values will be imputed - ‘MBimpute’ parameter of the dataProcess function
censored_symbol censored missing value indicator - ‘censoredInt’ parameter of the dataProcess function
remove_uninformative_feature_outlier if TRUE, features labeled as outlier of uninformative by the MSstatsSelectFeatures function will not be used in summarization

Value
data.table
MSstatsSelectFeatures  Feature selection before feature-level data summarization

Description

Feature selection before feature-level data summarization

Usage

MSstatsSelectFeatures(input, method, top_n = 3, min_feature_count = 2)

Arguments

- **input**: data.table
- **method**: "all" / "highQuality", "topN"
- **top_n**: number of features to use for "topN" method
- **min_feature_count**: number of quality features for "highQuality" method

Value

data.table

Examples

```r
raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
head(input)

input_all = MSstatsSelectFeatures(input, "all") # all features
input_5 = MSstatsSelectFeatures(data.table::copy(input), "topN", top_n = 5) # top 5 features
input_informative = MSstatsSelectFeatures(input, "highQuality") # feature selection
```
**MSstatsSummarizationOutput**

Post-processing output from MSstats summarization

**Description**

Post-processing output from MSstats summarization

**Usage**

```r
MSstatsSummarizationOutput(
  input,
  summarized,
  processed,
  method,
  impute,
  censored_symbol
)
```

**Arguments**

- **input**: 'data.table' in MSstats format
- **summarized**: output of the `MSstatsSummarize` function
- **processed**: output of `MSstatsSelectFeatures`
- **method**: name of the summarization method (`summaryMethod` parameter to `dataProcess`)  
- **impute**: if TRUE, censored missing values were imputed (`MBimpute` parameter to `dataProcess`) 
- **censored_symbol**: censored missing value indicator (`censoredInt` parameter to `dataProcess`) 

**Value**

list that consists of the following elements:

- FeatureLevelData - feature-level data after processing
- ProteinLevelData - protein-level (summarized) data
- SummaryMethod (string) - name of summarization method that was used
Examples

raw = DDARawData
method = "TMP"
cens = "NA"
impute = TRUE
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZEMEDIANS")
input = MSstatsMergeFractions(input)
input = MSstatsHandleMissing(input, "TMP", TRUE, "NA", 0.999)
input = MSstatsSelectFeatures(input, "all")
processed = getProcessed(input)
input = MSstatsPrepareForSummarization(input, method, impute, cens, FALSE)
input_split = split(input, input$PROTEIN)
summarized = MSstatsSummarize(input_split, method, impute, cens, FALSE, TRUE)
output = output = MSstatsSummarizationOutput(input, summarized, processed,
method, impute, cens)

MSstatsSummarize  Feature-level data summarization

Description

Feature-level data summarization

Usage

MSstatsSummarize(
proteins_list,
method,
impute,
censored_symbol,
remove50missing,
equal_variance
)

Arguments

proteins_list list of processed feature-level data
method summarization method: "linear" or "TMP"
impute only for summaryMethod = "TMP" and censoredInt = 'NA' or '0'. TRUE (default) imputes 'NA' or '0' (depending on censoredInt option) by Accelerated failure model. FALSE uses the values assigned by cutoffCensored
censored_symbol Missing values are censored or at random. 'NA' (default) assumes that all 'NA's in 'Intensity' column are censored. '0' uses zero intensites as censored intensity. In this case, NA intensites are missing at random. The output from Skyline should use '0'. Null assumes that all NA intensites are randomly missing.
**MSstatsSummarizeSingleLinear**

Linear model-based summarization for a single protein

**Description**

Linear model-based summarization for a single protein

**Usage**

```r
MSstatsSummarizeSingleLinear(single_protein, equal_variances = TRUE)
```
**Arguments**

- **single_protein** feature-level data for a single protein
- **equal_variances**
  - if TRUE, observation are assumed to be homoskedastic

**Value**

list with protein-level data

**Examples**

```r
raw = DDARawData
dc = "linear"
cens = NULL
impute = FALSE
# currently, MSstats only supports MBimpute = FALSE for linear summarization
MSstatsConvert::MSstatsLogsSettings(TRUE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZATION"
input = MSstatsMergeFractions(input)
input = MSstatsHandleMissing(input, "TMP", TRUE, "NA", 0.999)
input = MSstatsSelectFeatures(input, "all"
input = MSstatsPrepareForSummarization(input, dc, impute, cens, FALSE)
input_split = split(input, input$PROTEIN)
single_protein_summary = MSstatsSummarizeSingleLinear(input_split[[1]])
head(single_protein_summary[[1]])
```

---

**MSstatsSummarizeSingleTMP**

*Tukey Median Polish summarization for a single protein*

---

**Description**

Tukey Median Polish summarization for a single protein

**Usage**

```r
MSstatsSummarizeSingleTMP(
  single_protein,
  impute,
  censored_symbol,
  remove50missing
)
```
**OpenMStoMSstatsFormat**

**Arguments**

- `single_protein` feature-level data for a single protein
- `impute` only for `summaryMethod = "TMP"` and `censoredInt = 'NA' or '0'`. TRUE (default) imputes 'NA' or '0' (depending on `censoredInt` option) by Accelerated failure model. FALSE uses the values assigned by `cutoffCensored`
- `censored_symbol` Missing values are censored or at random. 'NA' (default) assumes that all 'NA's in 'Intensity' column are censored. '0' uses zero intensities as censored intensity. In this case, NA intensities are missing at random. The output from Skyline should use '0'. Null assumes that all NA intensites are randomly missing.
- `remove50missing` only for `summaryMethod = "TMP"`. TRUE removes the runs which have more than 50% missing values. FALSE is default.

**Value**

list of two data.tables: one with fitted survival model, the other with protein-level data

**Examples**

```r
raw = DDARawData
tmp_method = "TMP"
tmp_cens = "NA"
# currently, MSstats only supports MBimpute = FALSE for linear summarization
MSstatsConvert::MSstatsLogsSettings(FALSE)
input = MSstatsPrepareForDataProcess(raw, 2, NULL)
input = MSstatsNormalize(input, "EQUALIZE_MEDIANS")
input = MSstatsMergeFractions(input)
input = MSstatsHandleMissing(input, "TMP", TRUE, "NA", 0.999)
input = MSstatsSelectFeatures(input, "all")
input = MSstatsPrepareForSummarization(input, tmp_method, tmp_impute, tmp_cens, FALSE)
input_split = split(input, input$PROTEIN)
single_protein_summary = MSstatsSummarizeSingleTMP(input_split[[1]],
  tmp_impute, tmp_cens, FALSE)
head(single_protein_summary[[1]])
```

**Description**

Import OpenMS files
Usage

OpenMStoMSstatsFormat(
  input, annotation = NULL, useUniquePeptide = TRUE, removeFewMeasurements = TRUE, removeProtein_with1Feature = FALSE, summaryforMultipleRows = max, use_log_file = TRUE, append = FALSE, verbose = TRUE, log_file_path = NULL, ...
)

Arguments

  input       name of MSstats input report from OpenMS, which includes feature(peptide ion)-level data.
  annotation  name of ‘annotation.txt’ data which includes Condition, BioReplicate, Run. Run should be the same as filename.
  useUniquePeptide  TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
  removeFewMeasurements  TRUE (default) will remove the features that have 1 or 2 measurements across runs.
  removeProtein_with1Feature  TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.
  summaryforMultipleRows  max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.
  use_log_file  logical. If TRUE, information about data processing will be saved to a file.
  append  logical. If TRUE, information about data processing will be added to an existing log file.
  verbose  logical. If TRUE, information about data processing will be printed to the console.
  log_file_path  character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.
  ...

Value

data.frame in the MSstats required format.
Author(s)
Meena Choi, Olga Vitek.

Examples
openms_raw = data.table::fread(system.file("tinytest\raw_data/OpenMS/openms_input.csv", package = "MSstatsConvert"))
openms_imported = OpenMStoMSstatsFormat(openms_raw, use_log_file = FALSE)
head(openms_imported)

---

OpenSWATHtoMSstatsFormat

Import OpenSWATH files

Description
Import OpenSWATH files

Usage
OpenSWATHtoMSstatsFormat(
  input,
  annotation,
  filter_with_mscore = TRUE,
  mscore_cutoff = 0.01,
  useUniquePeptide = TRUE,
  removeFewMeasurements = TRUE,
  removeProtein_with1Feature = FALSE,
  summaryforMultipleRows = max,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

Arguments

input name of MSstats input report from OpenSWATH, which includes feature-level data.
annotation name of 'annotation.txt' data which includes Condition, BioReplicate, Run. Run should be the same as filename.
filter_with_mscore TRUE(default) will filter out the features that have greater than mscore_cutoff in m_score column. Those features will be removed.
mscore_cutoff Cutoff for m_score. Default is 0.01.
OpenSWATHtoMSstatsFormat

useUniquePeptide
  TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

removeFewMeasurements
  TRUE (default) will remove the features that have 1 or 2 measurements across runs.

removeProtein_with1Feature
  TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.

summaryforMultipleRows
  max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.

use_log_file
  logical. If TRUE, information about data processing will be saved to a file.

append
  logical. If TRUE, information about data processing will be added to an existing log file.

verbose
  logical. If TRUE, information about data processing wil be printed to the console.

log_file_path
  character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

... additional parameters to ‘data.table::fread’.

Value
data.frame in the MSstats required format.

Author(s)
Meena Choi, Olga Vitek.

Examples

```r
os_raw = system.file("tinytest/raw_data/OpenSWATH/openswath_input.csv",
                      package = "MSstatsConvert")
annot = system.file("tinytest/annotations/annot_os.csv",
                   package = "MSstats")
os_raw = data.table::fread(os_raw)
annot = data.table::fread(annot)
os_imported = OpenSWATHtoMSstatsFormat(os_raw, annot, use_log_file = FALSE)
head(os_imported)
```
PDtoMSstatsFormat  Import Proteome Discoverer files

Description
Import Proteome Discoverer files

Usage
PDtoMSstatsFormat(
  input,
  annotation,
  useNumProteinsColumn = FALSE,
  useUniquePeptide = TRUE,
  summaryforMultipleRows = max,
  removeFewMeasurements = TRUE,
  removeOxidationMpeptides = FALSE,
  removeProtein_with1Peptide = FALSE,
  which.quantification = "Precursor.Area",
  which.proteinid = "Protein.Group.Accessions",
  which.sequence = "Sequence",
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

Arguments
input  PD report or a path to it.
annotation  name of 'annotation.txt' or 'annotation.csv' data which includes Condition, BioReplicate, Run information. 'Run' will be matched with 'Spectrum.File'.
useNumProteinsColumn  TRUE removes peptides which have more than 1 in # Proteins column of PD output.
useUniquePeptide  TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.
summaryforMultipleRows  max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.
removeFewMeasurements  TRUE (default) will remove the features that have 1 or 2 measurements across runs.
removeOxidationMpeptides
   TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.
removeProtein_with1Peptide
   TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.
which.quantification
   Use 'Precursor.Area'(default) column for quantified intensities. 'Intensity' or 'Area' can be used instead.
which.proteinid
   Use 'Protein.Accessions'(default) column for protein name. 'Master.Protein.Accessions' can be used instead.
which.sequence
   Use 'Sequence'(default) column for peptide sequence. 'Annotated.Sequence' can be used instead.
use_log_file
   logical. If TRUE, information about data processing will be saved to a file.
append
   logical. If TRUE, information about data processing will be added to an existing log file.
verbose
   logical. If TRUE, information about data processing will be printed to the console.
log_file_path
   character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.
...
   additional parameters to 'data.table::fread'.

Value
data.frame in the MSstats required format.

Author(s)
Meena Choi, Olga Vitek

Examples

pd_raw = system.file("tinytest/raw_data/PD/pd_input.csv", 
   package = "MSstatsConvert")
annot = system.file("tinytest/annotations/annot_pd.csv", package = "MSstats")
pd_raw = data.table::fread(pd_raw)
annot = data.table::fread(annot)

pd_imported = PDtoMSStatsFormat(pd_raw, annot, use_log_file = FALSE)
head(pd_imported)
ProgenesistoMSstatsFormat

Import Progenesis files

Description

Import Progenesis files

Usage

ProgenesistoMSstatsFormat(
  input,
  annotation,
  useUniquePeptide = TRUE,
  summaryforMultipleRows = max,
  removeFewMeasurements = TRUE,
  removeOxidationMpeptides = FALSE,
  removeProtein_with1Peptide = FALSE,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

Arguments

input name of Progenesis output, which is wide-format. 'Accession', 'Sequence', 'Modification', 'Charge' and one column for each run are required.

annotation name of 'annotation.txt' or 'annotation.csv' data which includes Condition, BioReplicate, Run information. It will be matched with the column name of input for MS runs.

useUniquePeptide TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

summaryforMultipleRows max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.

removeFewMeasurements TRUE (default) will remove the features that have 1 or 2 measurements across runs.

removeOxidationMpeptides TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.

removeProtein_with1Peptide TRUE will remove the proteins which have only 1 peptide and charge. FALSE is default.
quantification

use_log_file logical. If TRUE, information about data processing will be saved to a file.
append logical. If TRUE, information about data processing will be added to an existing log file.
verbose logical. If TRUE, information about data processing will be printed to the console.
log_file_path character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

... additional parameters to 'data.table::fread'.

Value
data.frame in the MSstats required format.

Author(s)
Meena Choi, Olga Vitek, Ulrich Omasits

Examples

progenesis_raw = system.file("tinytest/raw_data/Progenesis/progenesis_input.csv",
                          package = "MSstatsConvert")
annot = system.file("tinytest/raw_data/Progenesis/progenesis_annot.csv",
                   package = "MSstatsConvert")
progenesis_raw = data.table::fread(progenesis_raw)
annot = data.table::fread(annot)

progenesis_imported = ProgenesistoMSstatsFormat(progenesis_raw, annot,
                                             use_log_file = FALSE)
head(progenesis_imported)

quantification Protein sample quantification or group quantification

Description

Model-based quantification for each condition or for each biological sample per protein in a targeted Selected Reaction Monitoring (SRM), Data-Dependent Acquisition (DDA or shotgun), and Data-Independent Acquisition (DIA or SWATH-MS) experiment. Quantification takes the processed data set by dataProcess as input and automatically generate the quantification results (data.frame) in a long or matrix format.
Usage

quantification(
  data, 
  type = "Sample", 
  format = "matrix", 
  use_log_file = TRUE, 
  append = FALSE, 
  verbose = TRUE, 
  log_file_path = NULL
)

Arguments

data name of the (processed) data set.
type choice of quantification. "Sample" or "Group" for protein sample quantification or group quantification.
format choice of returned format. "long" for long format which has the columns named Protein, Condition, LogIntensities (and BioReplicate if it is subject quantification), NumFeature for number of transitions for a protein, and NumPeaks for number of observed peak intensities for a protein. "matrix" for data matrix format which has the rows for Protein and the columns, which are Groups(or Conditions) for group quantification or the combinations of BioReplicate and Condition (labeled by "BioReplicate"."Condition") for sample quantification. Default is "matrix"
use_log_file logical. If TRUE, information about data processing will be saved to a file.
append logical. If TRUE, information about data processing will be added to an existing log file.
verbose logical. If TRUE, information about data processing will be printed to the console.
log_file_path character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If `append = TRUE`, has to be a valid path to a file.

Details

• Sample quantification: individual biological sample quantification for each protein. The label of each biological sample is a combination of the corresponding group and the sample ID. If there are no technical replicates or experimental replicates per sample, sample quantification is the same as run summarization from dataProcess. If there are technical replicates or experimental replicates, sample quantification is median among run quantification corresponding MS runs.

• Group quantification: quantification for individual group or individual condition per protein. It is median among sample quantification.

• The quantification for endogenous samples is based on run summarization from subplot model, with TMP robust estimation.
Value
data.frame as described in details.

Examples

# Consider quantitative data (i.e. QuantData) from a yeast study with ten time points of
# interests, three biological replicates, and no technical replicates which is
# a time-course experiment.
# Sample quantification shows model-based estimation of protein abundance in each biological
# replicate within each time point.
# Group quantification shows model-based estimation of protein abundance in each time point.
QuantData<--dataProcess(SRMRawData, use_log_file = FALSE)
head(QuantData$FeatureLevelData)
# Sample quantification
sampleQuant<--quantification(QuantData, use_log_file = FALSE)
head(sampleQuant)
# Group quantification
groupQuant<--quantification(QuantData, type="Group", use_log_file = FALSE)
head(groupQuant)

savePlot

---

savePlot

Save a plot to pdf file

Description

Save a plot to pdf file

Usage

savePlot(name_base, file_name, width, height)

Arguments

name_base path to a folder (or "" for working directory)
file_name name of a file to save. If this file already exists, an integer will be appended to
this name
width width of a plot
height height of a plot
Convert SDRF experimental design file into an MSstats annotation file

Description
Takes an SDRF file and outputs an MSstats annotation file. Note the information in the SDRF file must be correctly annotated for MSstats so that MSstats can identify the experimental design. In particular the biological replicates must be correctly annotated, with group comparison experiments having a unique ID for each BioReplicate. For more information on this please see the Supplementary of the most recent MSstats paper.

Usage
SDRFtoAnnotation(
  data, 
  run_name = "comment[data file]", 
  condition_name = "characteristics[disease]", 
  biological_replicate = "characteristics[biological replicate]", 
  fraction = NULL 
)

Arguments
- data: SDRF annotation file
- run_name: Column name in SDRF file which contains the name of the MS run. The information in this column must match exactly with the run names in the PSM file
- condition_name: Column name in SDRF file which contains information on the conditions in the data.
- biological_replicate: Column name in SDRF file which contains the identifier for the biological replicate. Note MSstats uses this column to determine if the experiment is a repeated measure design. BioReplicate IDs should only be reused if the replicate was measured multiple times.
- fraction: Column name in SDFT file which contains information on the fractionation in the data. Only required if data contains fractions. Default is ‘NULL’

Examples
head(example_SDRF)

msstats_annotation = SDRFtoAnnotation(example_SDRF)

head(msstats_annotation)
SkylinetoMSstatsFormat

Import Skyline files

Description

Import Skyline files

Usage

SkylinetoMSstatsFormat(
  input,
  annotation = NULL,
  removeiRT = TRUE,
  filter_with_Qvalue = TRUE,
  qvalue_cutoff = 0.01,
  useUniquePeptide = TRUE,
  removeFewMeasurements = TRUE,
  removeOxidationMpeptides = FALSE,
  removeProtein_with1Feature = FALSE,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>name of MSstats input report from Skyline, which includes feature-level data.</td>
</tr>
<tr>
<td>annotation</td>
<td>name of 'annotation.txt' data which includes Condition, BioReplicate, Run. If annotation is already complete in Skyline, use annotation=NULL (default). It will use the annotation information from input.</td>
</tr>
<tr>
<td>removeiRT</td>
<td>TRUE (default) will remove the proteins or peptides which are labeled 'iRT' in 'StandardType' column. FALSE will keep them.</td>
</tr>
<tr>
<td>filter_with_Qvalue</td>
<td>TRUE (default) will filter out the intensities that have greater than qvalue_cutoff in DetectionQValue column. Those intensities will be replaced with zero and will be considered as censored missing values for imputation purpose.</td>
</tr>
<tr>
<td>qvalue_cutoff</td>
<td>Cutoff for DetectionQValue. default is 0.01.</td>
</tr>
<tr>
<td>useUniquePeptide</td>
<td>TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.</td>
</tr>
<tr>
<td>removeFewMeasurements</td>
<td>TRUE (default) will remove the features that have 1 or 2 measurements across runs.</td>
</tr>
</tbody>
</table>
removeOxidationMpeptides
   TRUE will remove the peptides including 'oxidation (M)' in modification. FALSE is default.
removeProtein_with1Feature
   TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.
use_log_file
   logical. If TRUE, information about data processing will be saved to a file.
append
   logical. If TRUE, information about data processing will be added to an existing log file.
verbose
   logical. If TRUE, information about data processing will be printed to the console.
log_file_path
   character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If 'append = TRUE', has to be a valid path to a file.
... additional parameters to 'data.table::fread'.

Value

data.frame in the MSstats required format.

Author(s)

Meena Choi, Olga Vitek

Examples

skyline_raw = system.file("tinytest/raw_data/Skyline/skyline_input.csv", package = "MSstatsConvert")
skyline_raw = data.table::fread(skyline_raw)
skyline_imported = SkylinetoMSstatsFormat(skyline_raw)
head(skyline_imported)
SpectronauttoMSstatsFormat

Usage

SpectronauttoMSstatsFormat(
  input,
  annotation = NULL,
  intensity = "PeakArea",
  filter_with_Qvalue = TRUE,
  qvalue_cutoff = 0.01,
  useUniquePeptide = TRUE,
  removeFewMeasurements = TRUE,
  removeProtein_with1Feature = FALSE,
  summaryforMultipleRows = max,
  use_log_file = TRUE,
  append = FALSE,
  verbose = TRUE,
  log_file_path = NULL,
  ...
)

Arguments

input name of Spectronaut output, which is long-format. ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity, F.ExcludedFromQuantification are required. Rows with F.ExcludedFromQuantification=True will be removed.

annotation name of 'annotation.txt' data which includes Condition, BioReplicate, Run. If annotation is already complete in Spectronaut, use annotation=NULL (default). It will use the annotation information from input.

intensity 'PeakArea'(default) uses not normalized peak area. 'NormalizedPeakArea' uses peak area normalized by Spectronaut.

filter_with_Qvalue TRUE(default) will filter out the intensities that have greater than qvalue_cutoff in EG.Qvalue column. Those intensities will be replaced with zero and will be considered as censored missing values for imputation purpose.

qvalue_cutoff Cutoff for EG.Qvalue. default is 0.01.

useUniquePeptide TRUE (default) removes peptides that are assigned for more than one proteins. We assume to use unique peptide for each protein.

removeFewMeasurements TRUE (default) will remove the features that have 1 or 2 measurements across runs.

removeProtein_with1Feature TRUE will remove the proteins which have only 1 feature, which is the combination of peptide, precursor charge, fragment and charge. FALSE is default.

summaryforMultipleRows max(default) or sum - when there are multiple measurements for certain feature and certain run, use highest or sum of multiple intensities.
use_log_file logical. If TRUE, information about data processing will be saved to a file.
append logical. If TRUE, information about data processing will be added to an existing log file.
verbose logical. If TRUE, information about data processing will be printed to the console.
log_file_path character. Path to a file to which information about data processing will be saved. If not provided, such a file will be created automatically. If ‘append = TRUE’, has to be a valid path to a file.

Value
data.frame in the MSstats required format.

Author(s)
Meena Choi, Olga Vitek

Examples
spectronaut_raw = system.file("tinytest/raw_data/Spectronaut/spectronaut_input.csv", package = "MSstatsConvert")
spectronaut_raw = data.table::fread(spectronaut_raw)
spectronaut_imported = SpectronauttoMSstatsFormat(spectronaut_raw, use_log_file = FALSE)
head(spectronaut_imported)

Description
This is a partial data set obtained from a published study (Picotti, et. al, 2009). The experiment targeted 45 proteins in the glycolysis/gluconeogenesis/TCA cycle/glyoxylate cycle network, which spans the range of protein abundance from less than 128 to 10E6 copies per cell. Three biological replicates were analyzed at ten time points (T1-T10), while yeasts transited through exponential growth in a glucose-rich medium (T1-T4), diauxic shift (T5-T6), post-diauxic phase (T7-T9), and stationary phase (T10). Prior to trypsinization, the samples were mixed with an equal amount of proteins from the same N15-labeled yeast sample, which was used as a reference. Each sample was profiled in a single mass spectrometry run, where each protein was represented by up to two peptides and each peptide by up to three transitions. The goal of this study is to detect significantly change in protein abundance across time points. Transcriptional activity under the same experimental conditions has been previously investigated by (DeRisi et. al., 1997). Genes coding for 29 of the proteins are differentially expressed between conditions similar to those represented by T7 and T1 and could be treated as external sources to validate the proteomics analysis. In this exampled data set, two of the targeted proteins are selected and validated with gene expression study: Protein IDHC (gene name IDP2) is differentially expressed in time point 1 and time point 7, whereas, Protein PMG2 (gene name GPM2) is not. The protein names are based on Swiss Prot Name.
Usage

SRMRawData

Format

data.frame

Details

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed.

If the information of one or more columns is not available for the original raw data, please retain the column variables and type in fixed value. For example, the original raw data does not contain the information of ProductCharge, we retain the column ProductCharge and type in NA for all transitions in RawData.

The column BioReplicate should label with unique patient ID (i.e., same patients should label with the same ID).

Variable Intensity is required to be original signal without any log transformation and can be specified as the peak of height or the peak of area under curve.

Value

data.frame with the required format of MSstats.

Author(s)

Meena Choi, Olga Vitek.

Maintainer: Meena Choi (<mnchoi67@gmail.com>)

References


Examples

head(SRMRawData)
theme_msstats

Theme for MSstats plots

Description

Theme for MSstats plots

Usage

theme_msstats(
  type,
  x.axis.size = 10,
  y.axis.size = 10,
  legend_size = 13,
  strip_background = element_rect(fill = "gray95"),
  strip_text_x = element_text(colour = c("black"), size = 14),
  legend_position = "top",
  legend_box = "vertical",
  text_angle = 0,
  text_hjust = NULL,
  text_vjust = NULL,
  ...
)

Arguments

type type of a plot
x.axis.size size of text on the x axis
y.axis.size size of text on the y axis
legend_size size of the legend
strip_background background of facet
strip_text_x size of text on facets
legend_position position of the legend
legend_box legend.box
text_angle angle of text on the x axis (for condition and comparison plots)
text_hjust hjust parameter for x axis text (for condition and comparison plots)
text_vjust vjust parameter for x axis text (for condition and comparison plots)
... additional parameters passed on to ggplot2::theme()
Index

* internal
  .addCoverageInfo, 5
  .addModelInformation, 5
  .addModelVariances, 6
  .addNInformativeInfo, 6
  .addNoisyFlag, 7
  .addOutlierCutoff, 7
  .addOutlierInformation, 8
  .addSurvivalPredictions, 8
  .adjustLRuns, 9
  .calculateOutlierCutoff, 9
  .calculatePower, 10
  .calculateProteinVariance, 10
  .checkContrastMatrix, 11
  .checkDataProcessParams, 11
  .checkExperimentDesign, 12
  .checkGCPlotsInput, 12
  .checkGroupComparisonInput, 13
  .checkSingleFeature, 13
  .checkSingletonLabelProteins, 14
  .checkSingleSubject, 14
  .checkTechReplicate, 15
  .checkUnProcessedDataValidity, 15
  .countInformative, 16
  .countMissingPercentage, 16
  .documentFunction, 17
  .finalizeInput, 18
  .finalizeLinear, 18
  .finalizeTMP, 19
  .fitHuber, 19
  .fitLinearModel, 20
  .fitModelForGroupComparison, 20
  .fitModelSingleProtein, 21
  .fitTukey, 22
  .flagLowCoverage, 22
  .flagUninformativeSingleLabel, 23
  .getAllComparisons, 23
  .getContrast, 24
  .getContrastLabels, 24
  .getEmptyComparison, 25
  .getFeatureVariances, 25
  .getMedian, 26
  .getMedianSigmaSubject, 26
  .getMin, 26
  .getModelParameters, 27
  .getNonMissingFilter, 27
  .getNonMissingFilterStats, 28
  .getNumSample, 28
  .getSingleProteinForProfile, 29
  .getVarComponent, 29
  .getWideTable, 30
  .getYaxis, 30
  .handleEmptyConditions, 31
  .handleSingleContrast, 31
  .isSummarizable, 32
  .logDatasetInformation, 32
  .logMissingness, 33
  .logSingletonLabeledProteins, 33
  .logSummaryStatistics, 34
  .makeComparison, 34
  .makeConditionPlot, 35
  .makeFactorColumns, 36
  .makeHeatmap, 36
  .makeProfilePlot, 37
  .makeQCPlot, 38
  .makeSummaryProfilePlot, 40
  .makeVolcano, 41
  .nicePrint, 42
  .normalizeGlobalStandards, 42
  .normalizeMedian, 43
  .normalizeQuantile, 43
  .onLoad, 43
  .plotComparison, 44
  .plotHeatmap, 45
  .plotVolcano, 46
  .preProcessIntensities, 50
  .prepareForDataProcess, 48
  .prepareLinear, 48
.prepareSingleProteinForGC, 49
.prepareSummary, 49
.prepareTMP, 50
.quantileNormalizationSingleLabel, 51
.replaceZerosWithNA, 51
.runTukey, 52
.saveSessionInfo, 52
.saveTable, 53
.selectHighQualityFeatures, 53
.selectTopFeatures, 54
.setCensoredByThreshold, 54
.updateColumnsForProcessing, 55
.updateUnequalVariances, 55
.addCoverageInfo, 5
.addModelInformation, 5
.addModelVariances, 6
.addNInformativeInfo, 6
.addNoisyFlag, 7
.addOutlierCutoff, 7
.addOutlierInformation, 8
.addSurvivalPredictions, 8
.adjustLRuns, 9
.calculateOutlierCutoff, 9
.calculatePower, 10
.calculateProteinVariance, 10
.checkContrastMatrix, 11
.checkDataProcessParams, 11
.checkExperimentDesign, 12
.checkGCPlotsInput, 12
.checkGroupComparisonInput, 13
.checkSingleFeature, 13
.checkSingleLabelProteins, 14
.checkSingleSubject, 14
.checkTechReplicate, 15
.checkUnProcessedDataValidity, 15
.countInformative, 16
.countMissingPercentage, 16
.documentFunction, 17
.finalizeInput, 18
.finalizeLinear, 18
.finalizeTMP, 19
.fitHuber, 19
.fitLinearModel, 20
.fitModelForGroupComparison, 20
.fitModelSingleProtein, 21
.fitTukey, 22
.flagLowCoverage, 22
.flagUninformativeSingleLabel, 23
.getAllComparisons, 23
.getContrast, 24
.getContrastLabels, 24
.getEmptyComparison, 25
.getFeatureVariances, 25
.getMedian, 26
.getMedianSigmaSubject, 26
.getMin, 26
.getModelParameters, 27
.getNonMissingFilter, 27
.getNonMissingFilterStats, 28
.getNumSample, 28
.getSingleProteinForProfile, 29
.getVarComponent, 29
.getWideTable, 30
.getYaxis, 30
.handleEmptyConditions, 31
.handleSingleContrast, 31
.isSummarizable, 32
.logDatasetInformation, 32
.logMissingness, 33
.logSingleLabeledProteins, 33
.logSummaryStatistics, 34
.makeComparison, 34
.makeConditionPlot, 35
.makeFactorColumns, 36
.makeHeatmap, 36
.makeProfilePlot, 37
.makeQCPlot, 38
.makeSummaryProfilePlot, 40
.makeVolcano, 41
.nicePrint, 42
.normalizeGlobalStandards, 42
.normalizeMedian, 43
.normalizeQuantile, 43
.onLoad, 43
.plotComparison, 44
.plotHeatmap, 45
.plotVolcano, 46
.preProcessIntensities, 50
.prepareForDataProcess, 48
.prepareLinear, 48
.prepareSingleProteinForGC, 49
.prepareSummary, 49
.prepareTMP, 50
.quantileNormalizationSingleLabel, 51
.replaceZerosWithNA, 51
.runTukey, 52
.saveSessionInfo, 52
.saveTable, 53
.selectHighQualityFeatures, 53
.selectTopFeatures, 54
.setCensoredByThreshold, 54
.updateColumnsForProcessing, 55
.updateUnequalVariances, 55

checkRepeatedDesign, 56
dataProcess, 38, 39, 56, 59, 61, 64, 80, 106
dataProcessPlots, 59
DDARawData, 62
DDARawData.Skyline, 63
designSampleSize, 64, 66, 67
designSampleSizePlots, 66
DIANNtoMSstatsFormat, 67
DIARawData, 69
DIAUmpiretoMSstatsFormat, 70

extractSDRF, 72

FragPipetoMSstatsFormat, 73
generated, 74
getSamplesInfo, 75
getSelectedProteins, 76
groupComparison, 76, 78, 84
groupComparisonPlots, 78

makePeptidesDictionary, 81
MaxQtoMSstatsFormat, 82
modelBasedQCPlots, 84
MSstatsContrastMatrix, 85
MSstatsGroupComparison, 86
MSstatsGroupComparisonOutput, 87
MSstatsGroupComparisonSingleProtein, 88
MSstatsHandleMissing, 89
MSstatsMergeFractions, 90
MSstatsNormalize, 90
MSstatsPrepareForDataProcess, 91
MSstatsPrepareForGroupComparison, 92
MSstatsPrepareForSummarization, 93
MSstatsSelectFeatures, 94
MSstatsSummarizationOutput, 95
MSstatsSummarize, 96
MSstatsSummarizeSingleLinear, 97
MSstatsSummarizeSingleTMP, 98

OpenMStoMSstatsFormat, 99
OpenSWATHtoMSstatsFormat, 101
PDtoMSstatsFormat, 103
ProgenesistoMSstatsFormat, 105
quantification, 106
savePlot, 108
SDRFToAnnotation, 109
SkylinetoMSstatsFormat, 110
SpectronauttoMSstatsFormat, 111
SRMRawData, 113

theme_msstats, 115