Package ‘mzTabHelper’

January 29, 2020

Title What the Package Does (One Line, Title Case)

Version 0.0.0.9000

Description What the package does (one paragraph).

License What license it uses

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

calculateFoldChange *Determine fold changes and map to finite numbers.*

Description

fc = log2(abundances1/abundances2) Should be NaN-save, i.e. NaN in any of the two abundance vectors results in NaN in the fold change vector. NaN = not quantifiable, hence fc = NaN = not quantifiable

Usage

```
calculateFoldChange(abundances1, abundances2)
```

Arguments

abundances1	first abundance vector
abundances2	second abundance vector

Value

fold change

`checkAccessionFormat` *Check that the protein accession is of the format ..|..|..*

Description

Note that NA returns TRUE.

Usage

```
checkAccessionFormat(accession)
```

Arguments

`accession`	protein accession

Value

Boolean. Is the accession of the format ..|..|.. ?

`countOccurrences` *Count the occurences of character char in string s.*

Description

Count the occurences of character char in string s.

Usage

```
countOccurrences(char, s)
```

Arguments

`char`	character
`s`	string

Value

number of occurences

cutSequence *Reduces the length of the peptide sequence to the first 25 amino acids.*

Description

Reduces the length of the peptide sequence to the first 25 amino acids.

Usage

```
cutSequence(s)
```

Arguments

s	peptide sequence

Value

shorter peptide sequence

findPeptidesOfInterest *Find all peptides of interest.*

Description

We assume a <peptides.of.interest> vector exists.

Usage

```
findPeptidesOfInterest(data)
```

Arguments

data	peptide dataframe

Value

dataframe containing only peptides of interest.

findProteinsOfInterest *Find all proteins of interest.*

Description

We assume a <proteins.of.interest> vector exists.

Usage

```
findProteinsOfInterest(data)
```

Arguments

`data`	peptide dataframe

Value

dataframe containing only peptides which occur in <proteins.of.interst>

getAccession *Extracts the second entry from a string of the form ..|..|..*

Description

Extracts the second entry from a string of the form ..|..|..

Usage

```
getAccession(string)
```

Arguments

`string`	string of the form x|y|z

Value

y

getAverageIntensity *Returns an average peptide intensity over all study variables.*

Description

Returns an average peptide intensity over all study variables.

Usage

```
getAverageIntensity(data)
```

Arguments

`data`	dataframe with columns "peptide_abundance_study_variable[*]"

Value

mean intensity over all channels

getGene *Extracts the third entry from a string of the form ..|..|..*

Description

Extracts the third entry from a string of the form ..|..|..

Usage

```
getGene(string)
```

Arguments

`string`	string of the form x|y|z

Value

z

getModsSummary *Create a summary table of all modifications and their specificities.*

Description

Required input is a dataframe with a "sequence" and "modifications" column in mzTab standard.

Usage

```
getModsSummary(data)
```

Arguments

`data`	peptide dataframe

Value

summary table

getPCA *Calculate the principal component object.*

Description

Calculate the principal component object.

Usage

```
getPCA(data)
```

Arguments

`data`	peptide dataframe

Value

principal component object

getPCAeigenvector *Eigenvectors point in the direction of the principal componets in the high-dimensional peptide abundance space.*

Description

Important peptides (i.e. the ones with a large absolute eigenvector component) contribute most to this principal component. The function returns these peptides i.e. their row index.

Usage

```
getPCAeigenvector(pca, n)
```

Arguments

pca	principal component object, see getPCA()
n	number of principal component

Value

row indices of peptides in original peptide dataframe, see getPCA()

getPeptideQuants *Returns a dataframe containing only the peptide quantification columns.*

Description

Returns a dataframe containing only the peptide quantification columns.

Usage

```
getPeptideQuants(data)
```

Arguments

data	dataframe with columns "peptide_abundance_study_variable[*]"

Value

quants only dataframe

`getQuantSummary` *Create a summary table of all quantifications.*

Description

How many quantifications are finite, zero or NaN in each sample?

Usage

```
getQuantSummary(data)
```

Arguments

`data`	peptide dataframe

Value

summary table

`isEmpty` *Check if the vector/column is empty.*

Description

i.e. all entries are NA or "" etc. or the the vector is of length 0

Usage

```
isEmpty(column)
```

Arguments

`column`	column to be checked

Value

Boolean. Is the entire column empty?

makeModifiedSequenceChargeUnique

Makes the (modified sequence, charge) combination unique by picking the quants with maximum intensity.

Description

Makes the (modified sequence, charge) combination unique by picking the quants with maximum intensity.

Usage

```
makeModifiedSequenceChargeUnique(data)
```

Arguments

`data`	peptide dataframe

Value

subset of input dataframe with double rows removed

numberOfStudyVariables

Returns the number of quantification channels i.e. the number of "peptide_abundance_study_variable[*]" columns.

Description

Returns the number of quantification channels i.e. the number of "peptide_abundance_study_variable[*]" columns.

Usage

```
numberOfStudyVariables(data)
```

Arguments

`data`	dataframe

Value

number of quantification channels

`plotBoxplot` *Plot boxplot of all peptide quantifications.*

Description

Plot boxplot of all peptide quantifications.

Usage

```
plotBoxplot(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotChargeDistribution` *Plot charge distribution.*

Description

We assume a 'charge' column exists and do not check.

Usage

```
plotChargeDistribution(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotCorrelations` *Plot correlation of all peptide quantifications.*

Description

Plot correlation of all peptide quantifications.

Usage

```
plotCorrelations(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

Value

correlation

`plotDeltaFcLogIntensity` *Plot the difference between 2+ and 3+ fold changes of the same (modified) sequence against the logarithm of the average peptide intensity.*

Description

Plot the difference between 2+ and 3+ fold changes of the same (modified) sequence against the logarithm of the average peptide intensity.

Usage

```
plotDeltaFcLogIntensity(data, sample.1, sample.2, pdf.file)
```

Arguments

`data`	data frame
`sample.1`	number of the sample i.e. study variable
`sample.2`	number of the sample i.e. study variable
`pdf.file`	path to output pdf file

`plotDistribution` *Plot distribution.*

Description

Plot distribution.

Usage

```
plotDistribution(vector, label, pdf.file)
```

Arguments

`vector`	quantity for distribution
`label`	label for x-axis
`pdf.file`	path to output pdf file

`plotElutionTimeDistribution` *Plot the distribution of peptide elution times.*

Description

Each peptide reports a minimum/maximum retention time in the retention_time_window column.

Usage

```
plotElutionTimeDistribution(data, pdf.file)
```

Arguments

`data`	dataframe with retention_time_window column
`pdf.file`	path to output pdf file

`plotFcLogIntensity` *Plot fold change vs log intensity.*

Description

Plot fold change vs log intensity.

Usage

```
plotFcLogIntensity(fc.vector, intensity.vector, fc.label, pdf.file)
```

Arguments

`fc.vector`	fold change vector (x-axis)
`intensity.vector`	intensity vector (y-axis)
`fc.label`	label for x-axis
`pdf.file`	path to output pdf file

plotFcLogIntensitySingleProtein *Plot peptide fold change vs log intensity for a single specific protein.*

Description

(same peptide sequence -> same colour)

Usage

```
plotFcLogIntensitySingleProtein(data, protein, sample.1, sample.2,
  pdf.file)
```

Arguments

`data`	data frame
`protein`	protein
`sample.1`	number of the sample i.e. study variable
`sample.2`	number of the sample i.e. study variable
`pdf.file`	path to output pdf file

plotFcSingleProtein *Plot sample (or group) index vs fold change for all peptides of a specific protein.*

Description

The fold change is calculated relative to the sample with the most peptide quantifications.

Usage

```
plotFcSingleProtein(data, protein, pdf.file)
```

Arguments

`data`	data frame
`protein`	protein
`pdf.file`	path to output pdf file

plotFrequencyOfFrequencies

Plot frequency of frequencies of any vector. Take for example a vector of protein accessions.

Description

frequency: How often does a particular protein X occur? frequency of frequencies: How often does a protein occur twice or three times and so on?

Usage

```
plotFrequencyOfFrequencies(vector, pdf.file, xlab = "frequency",
  ylab = "frequency of frequency", log = "y")
```

Arguments

`vector`	any set, such as a vector of protein accessions
`pdf.file`	path to output pdf file

plotKendrick

Plot Kendrick nominal fractional mass plot.

Description

Plot Kendrick nominal fractional mass plot.

Usage

```
plotKendrick(mass, pdf.file)
```

Arguments

`mass`	peptide masses
`pdf.file`	path to output pdf file

`plotMultiplicityFrequency` *Plot (modified sequence, charge) pair multiplicity vs frequency plot.*

Description

Each peptide feature (characterised by a (possibly) modified peptide sequence and a charge state) should ideally occur only once in the analysis. In other words, peptides of multiplicity 1 should have a very high frequency. The plot below should show a significant spike on the left and can be used as QC of the analysis.

Usage

```
plotMultiplicityFrequency(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotPCAcomponents` *Plot the standard deviation of all principal components.*

Description

Plot the standard deviation of all principal components.

Usage

```
plotPCAcomponents(pca, pdf.file)
```

Arguments

`pca`	principal component object, see getPCA()
`pdf.file`	path to output pdf file

plotPCAeigenvector *Plot the coordinates of the nth eigenvector.*

Description

Plot the coordinates of the nth eigenvector.

Usage

```
plotPCAeigenvector(pca, data, n, pdf.file)
```

Arguments

`pca`	principal component object, see getPCA()
`data`	peptide dataframe
`n`	number of principal component
`pdf.file`	path to output pdf file

plotPCAscatter *Plot the scatter plot of the first n.pca principal components.*

Description

Plot the scatter plot of the first n.pca principal components.

Usage

```
plotPCAscatter(pca, pdf.file)
```

Arguments

`pca`	principal component object, see getPCA()
`pdf.file`	path to output pdf file

plotPeptidesOfInterest *Plots the reported peptide abundances of all peptides of interest.*

Description

Plots the reported peptide abundances of all peptides of interest.

Usage

```
plotPeptidesOfInterest(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotPeptidesPerProtein` *Plot quantified peptides per protein vs frequency.*

Description

Plot quantified peptides per protein vs frequency.

Usage

```
plotPeptidesPerProtein(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotProteinsOfInterest` *Plots the reported peptide abundances of all proteins of interest.*

Description

Plots the reported peptide abundances of all proteins of interest.

Usage

```
plotProteinsOfInterest(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`plotQuantFrequency` *Plot (in)complete quantifications.*

Description

Not all peptides need to be quantified in all channels/samples. See for example knock-out or TAILS experiments. Not quantified can mean either NaN or exactly zero. The plot below summarises how many peptides were quantified in x samples. 1 <= x <= number of samples

Usage

```
plotQuantFrequency(quants, pdf.file)
```

Arguments

`quants`	dataframe with peptide quantifications, see getPeptideQuants()
`pdf.file`	path to output pdf file

`plotRetentionTimeShiftDistribution` *Plot the retention time shift distribution.*

Description

Plot the retention time shift distribution.

Usage

```
plotRetentionTimeShiftDistribution(data, pdf.file)
```

Arguments

`data`	peptide dataframe
`pdf.file`	path to output pdf file

`readMzTabPEP` *Read the PEP section of an mzTab file.*

Description

Read the PEP section of an mzTab file.

Usage

```
readMzTabPEP(file)
```

Arguments

`file`	path to mzTab file

Value

dataframe of PEP section

`splitAccession` *Splits fasta protein accession into UniProt accession and gene name.*

Description

Splits fasta protein accession into UniProt accession and gene name.

Usage

```
splitAccession(peptide.data)
```

Arguments

`peptide.data`	dataframe with <accession> column

Value

dataframe with UniProt accession and gene name

`startSection` *Simple Moving Average without leading NA.*

Description

Simple Moving Average without leading NA.

Usage

```
startSection(file, section.identifier)
```

Arguments

`file`	path to mzTab file
`section.identifier`	identifier at the start of the section, either 'PEH', 'PRH' or 'PSH'

Value

first row of the section

`studyVariableExists` *Check if a specific "peptide_abundance_study_variable[n]" column exists.*

Description

Check if a specific "peptide_abundance_study_variable[n]" column exists.

Usage

```
studyVariableExists(data, n)
```

Arguments

`data`	dataframe
`n`	index of study variable

Value

Boolean. Does the column "peptide_abundance_study_variable[n]" exist?

`uniqueColors` *Returns a unique colour for each string.*

Description

Returns a unique colour for each string.

Usage

```
uniqueColors(string.vector)
```

Arguments

`string.vector`	vector of strings

Value

colours

Index