Package: PTXQC (via r-universe)

September 8, 2024

Type Package

Title Quality Report Generation for MaxQuant and mzTab Results

Version 1.1.1

Date 2024-03-11

Author Chris Bielow [aut, cre], Juliane Schmachtenberg [ctb], Swenja Wagner [ctb], Patricia Scheil [ctb], Tom Waschischek [ctb], Guido Mastrobuoni [dtc, rev]

Maintainer Chris Bielow <chris.bielow@bsc.fu-berlin.de>

Description Generates Proteomics (PTX) quality control (QC) reports for shotgun LC-MS data analyzed with the MaxQuant software suite (from .txt files) or mzTab files (ideally from OpenMS 'QualityControl' tool). Reports are customizable (target thresholds, subsetting) and available in HTML or PDF format. Published in J. Proteome Res., Proteomics Quality Control: Quality Control Software for MaxQuant Results (2015) <doi:10.1021/acs.jproteome.5b00780>.

SystemRequirements pandoc (http://pandoc.org) for building Vignettes and output reports as HTML

Depends R (>= 3.3.0)

Imports data.table, ggplot2 (>= 2.2), ggdendro, grid, gridExtra, grDevices, gtable, htmlTable, knitr (>= 1.10), magrittr, methods, plyr, R6, R6P, RColorBrewer, reshape2, rmarkdown, rmzqc (>= 0.5.0), seqinr, stats, utils, UpSetR, xml2, yaml

Suggests testthat

VignetteBuilder knitr

License BSD_3_clause + file LICENSE

Encoding UTF-8

Roxygen list()

RoxygenNote 7.3.1

URL https://github.com/cbielow/PTXQC

BugReports https://github.com/cbielow/PTXQC/issues

Repository https://cbielow.r-universe.dev

RemoteUrl https://github.com/cbielow/ptxqc

RemoteRef HEAD

RemoteSha 41ea336a44bbb68c9867e50004a99d6d6f95990e

Contents

`PTXQC-package` *PTXQC: A package for computing Quality Control (QC) metrics for Proteomics (PTX)*

Description

The following sections describe the main components:

Input

Valid input data are either the files from MaxQuant's .txt folder (all versions from MaxQuant >= 1.0 upwards are supported) or a single mzTab file. All mzTab files will work, but most metrics can be obtained from OpenMS' mzTab as produced by the QualityControl TOPP tool (from OpenMS 2.5 onwards).

Important functions

The central function of this package is called `createReport` and it accepts either MaxQuant or mzTab data, along with a configuration (optional). There is a parser for mzTab `MzTabReader` and MaxQuant txt files `MQDataReader`, as well as a plethora of QC metrics derived from a common `qcMetric` class and scoring functions `qual...`, e.g. `qualGaussDev`.

Configuration

The user can modify the behaviour of PTXQC, e.g. to enable/disable certain metrics or change scoring thresholds, via a YAML object/file. By default a Yaml file is written automatically side-by-side to the input files upon running PTXQC for the first time on a particular input. A custom Yaml object can be passed to the main `createReport` function for customization. Use `yaml::yaml.load_file(input = 'myYAML.yaml')` to load an existing file and pass the Yaml object along.

Output

Either a PDF and/or Html report which contains QC plots and a description of the metrics.

See Also

Useful links:

- `https://github.com/cbielow/PTXQC`
- Report bugs at `https://github.com/cbielow/PTXQC/issues`

`alignmentCheck` *Verify an alignment by checking the retention time differences of identical peptides across Raw files*

Description

The input is a data frame containing feature evidence with corrected retention times, e.g. a 'calibrated.retention.time' column.

Usage

```
alignmentCheck(data, referenceFile)
```

Arguments

`data`	A data.frame with columns 'calibrated.retention.time', 'id', 'modified.sequence', 'charge', 'raw.file' and 'fraction' (if present)
`referenceFile`	A raw file name as occuring in data$raw.file, serving as alignment reference (when no fractions are used).

Details

Note that this function must be given real MS/MS identifications only (type "MULTI-MSMS") in order to work correctly!

For each peptide sequence (and charge) in the reference Raw file, this function looks up the already calibrated retention time difference of the same feature in all other files. For every comparison made, we report the RT difference. If alignment worked perfectly, the differences are very small (<1 min).

An 'id' column must be present, to enable mapping the result of this function to the original data frame.

A reference Raw file can be identified using 'findAlignReference()'. If Maxquants experimental design included pre-fractionation, a column named 'fraction' should be given and 'referenceFile' should be empty. This function will pick the one Raw file for each fraction (the first in order) to use as reference. Only the immediately neighbouring fractions will be matched to this reference.

Value

A data.frame containing the RT diff for each feature found in a Raw file and the reference.

---

appendEnv *Add the value of a variable to an environment (fast append)*

Description

The environment must exist, and its name must be given as string literal in 'env_name'! The value of the variable 'v' will be stored under the name given in 'v_name'. If 'v_name' is not given, a variable name will be created by increasing an internal counter and using the its value padded with zeros as name (i.e., "0001", "0002" etc).

Usage

```
appendEnv(env_name, v, v_name = NULL)
```

Arguments

env_name	String of the environment variable
v	Value to be inserted
v_name	String used as variable name. Automatically generated if omitted.

Value

Always TRUE

`assembleMZQC` *Collects all 'mzQC' members from each entry in lst_qcMetrics and stores them in an overall mzQC object, which can be written to disk (see writeMZQC()) or augmented otherwise*

Description

Collects all 'mzQC' members from each entry in lst_qcMetrics and stores them in an overall mzQC object, which can be written to disk (see writeMZQC()) or augmented otherwise

Usage

```
assembleMZQC(lst_qcMetrics, raw_file_mapping)
```

Arguments

`lst_qcMetrics`	A list of qcMetric objects which have their mzQC member populated with "MzQCrunQuality" and/or "MzQCsetQuality" objects
`raw_file_mapping`	A data.frame with cols 'from', to' and maybe 'best.effort' (if shorting was unsuccessful), as e.g. obtained by a FilenameMapper$raw_file_mapping

Value

An MzQCmzQC object (root object of an mzQC document)

`assignBlocks` *Assign set numbers to a vector of values.*

Description

Each set has size set_size (internally optimized using `correctSetSize`), holding values from 'values'. This gives n such sets and the return value is just the set index for each value.

Usage

```
assignBlocks(values, set_size = 5, sort_values = TRUE)
```

Arguments

`values`	Vector of values
`set_size`	Number of distinct values allowed in a set
`sort_values`	Before assigning values to sets, sort the values?

Value

Vector (same length as input) with set numbers

Examples

```
#library(PTXQC)
assignBlocks(c(1:11, 1), set_size = 3, sort_values = FALSE)
## --> 1 1 1 2 2 2 3 3 3 4 4 1
```

---

boxplotCompare *Boxplots - one for each condition (=column) in a data frame.*

Description

Given a data.frame with two/three columns in long format (name, value, [contaminant]; in that order), each group (given from 1st column) is plotted as a bar. Contaminants (if given) are separated and plotted as yellow bars.

Usage

```
boxplotCompare(
  data,
  log2 = TRUE,
  ylab = "intensity",
  mainlab = ylab,
  sublab = "",
  boxes_per_page = 30,
  abline = NA,
  coord_flip = TRUE,
  names = NA
)
```

Arguments

data	Data frame in long format with numerical expression data
log2	Apply log2 to the data (yes/no)
ylab	Label on Y-axis
mainlab	Main title
sublab	Sub title
boxes_per_page	Maximum number of boxplots per plot. Yields multiple plots if more groups are given.
abline	Draw a horizontal green line at the specified y-position (e.g. to indicate target median values)
coord_flip	Exchange Y and X-axis for better readability
names	An optional data.frame(long=.., short=..), giving a renaming scheme (long->short) for the 'name' column

Details

Boxes are shaded: many NA or Inf lead to more transparency. Allows to easily spot sparse groups

Value

List of ggplot objects

`brewer.pal.Safe` *Return color brew palettes, but fail hard if number of requested colors is larger than the palette is holding.*

Description

Internally calls 'brewer.pal(n, palette)', checking 'n' beforehand.

Usage

```
brewer.pal.Safe(n = 3, palette = "Set1")
```

Arguments

`n`	Number of colours
`palette`	Name of palette (e.g. "set1")

Value

character vector of colors

`byX` *Calls FUN on a subset of data in blocks of size 'subset_size' of unique indices.*

Description

One subset consists of 'subset_size' unique groups and thus of all rows which in 'data' which have any of these groups. The last subset might have less groups, if the number of unique groups is not dividable by subset_size.

Usage

```
byX(data, indices, subset_size = 5, FUN, sort_indices = TRUE, ...)
```

Arguments

data	Data.frame whose subsets to use on FUN
indices	Vector of group assignments, same length as nrow(data)
subset_size	Number of groups to use in one subset
FUN	Function applied to subsets of data
sort_indices	Sort groups (by their sorted character(!) names) before building subsets
...	More arguments to FUN

Details

FUN is applied on each subset.

Value

list of function result (one entry for each subset)

Examples

```
byX(data.frame(d=1:10), 1:10, 2, sum)
```

---

byXflex *Same as byX, but with more flexible group size, to avoid that the last group has only a few entries (<50% of desired size).*

Description

The 'subset_size' param is internally optimized using correctSetSize and then byX is called.

Usage

```
byXflex(data, indices, subset_size = 5, FUN, sort_indices = TRUE, ...)
```

Arguments

data	Data.frame whose subset to use on FUN
indices	Vector of group assignments, same length as nrow(data)
subset_size	Ideal number of groups to use in one subset – this can be changed internally, from 75%-150%
FUN	function Applied to subsets of data
sort_indices	Groups are formed by their sorted character(!) names
...	More arguments to FUN

Value

list of function result (one entry for each subset)

Examples

```
stopifnot(
  byXflex(data.frame(d=1:10), 1:10, 2, sum, sort_indices = FALSE) ==
  c(3, 7, 11, 15, 19)
)
```

`checkEnglishLocale` *When MaxQuant is run with a wrong locale (i.e. the decimal separator is not a '.', but a ','), then MaxQuant results are plainly wrong and broken. The can be detected by, e.g. checking for negative charge annotation*

Description

When MaxQuant is run with a wrong locale (i.e. the decimal separator is not a '.', but a ','), then MaxQuant results are plainly wrong and broken. The can be detected by, e.g. checking for negative charge annotation

Usage

```
checkEnglishLocale(df_evd)
```

Arguments

`df_evd`	Evidence table from which we only need the 'charge' column

---

`computeMatchRTFractions` *Combine several data structs into a final picture for segmentation incurred by 'Match-between-runs'.*

Description

qMBRSeg_Dist_inGroup might be empty if there are only singlets (transferred and genuine), but then the scores will be pretty boring as well (100

Usage

```
computeMatchRTFractions(qMBR, qMBRSeg_Dist_inGroup)
```

Arguments

qMBR	A data.frame as computed by peakSegmentation()
qMBRSeg_Dist_inGroup	A data.frame as computed by inMatchWindow()

Value

A data.frame which details the distribution of singlets and pairs (inRT and outRT) for each Raw file and genuine vs. all

---

correctSetSize *Re-estimate a new set size to split a number of items into equally sized sets.*

Description

This is useful for plotting large datasets where multiple pages are needed. E.g. you know that you need 101 barplots, but you only want to fit about 25 per page. Naively one would now do five plots, with the last one only containing a single barplot. Using this function with correctSetSize(101, 25) would tell you to use 26 barplots per page, so you end up with four plots, all roughly equally filled. It also works the other extreme case, where your initial size is chosen slightly too high, e.g. Sets of size 5 for just 8 items is too much, because we can reduce the set size to 4 and still need two sets but now they are much more equally filled (correctSetSize(8, 5) == 4).

Usage

```
correctSetSize(item_count, initial_set_size)
```

Arguments

item_count	Known number of items which need to assigned to sets
initial_set_size	Desired number of items a single set should hold

Details

We allow for up to set sizes of 150% from default, to avoid the last set being sparse (we remove it and distribute to the other bins) 150 Once the number of sets is fixed, we distribute all items equally.

E.g. 6 items & initial_set_size=5, would result in 2 bins (5 items, 1 item), but we'd rather have one bin of 6 items or 8 items & initial_set_size=5, would result in 2 bins (5+3 items), since the last set is more than half full, but we'd rather have 4+4

Value

re-estimated set size which a set should hold in order to avoid underfilled sets

Examples

```
stopifnot(
  correctSetSize(8, 5) == 4
)
stopifnot(
  correctSetSize(101, 25) == 26
)
```

`createReport` *Create a quality control report (in PDF format).*

Description

This is the main function of the package and the only thing you need to call directly if you are just interested in getting a QC report.

Usage

```
createReport(
  txt_folder = NULL,
  mztab_file = NULL,
  yaml_obj = list(),
  report_filenames = NULL,
  enable_log = FALSE
)
```

Arguments

`txt_folder`	Path to txt output folder of MaxQuant (e.g. "c:/data/Hek293/txt")
`mztab_file`	Alternative to **txt_folder**, you can provide a single mzTab file which contains PSM, PEP and PRT tables
`yaml_obj`	A nested list object with configuration parameters for the report. Useful to switch off certain plots or skip entire sections.
`report_filenames`	Optional list with names (as generated by `getReportFilenames`). If not provided, will be created internally by calling `getReportFilenames`.
`enable_log`	If TRUE all console output (including warnings and errors) is logged to the file given in **report_filenames$log_file**. Note: warnings/errors can only be shown in either the log **or** the console, not both!

Details

You need to provide either a) the folder name of the 'txt' output, as generated by MaxQuant or an mzTab file or b) an mzTab file as generated by the OpenMS QualityControl TOPP tool (other mzTab files will probably not work)

Optionally, provide a YAML configuration object, which allows to (de)activate certain plots and holds other parameters. The yaml_obj is complex and best obtained by running this function once using the default (empty list). A full YAML configuration object will be written in the 'txt' folder you provide and can be loaded using `yaml.load`.

The PDF and the config file will be stored in the given txt folder.

Value

List with named filename strings, e.g. $yaml_file, $report_file etc..

Note

You need write access to the txt/mzTab folder!

For updates, bug fixes and feedback please visit https://github.com/cbielow/PTXQC.

`createYaml` *Creates a yaml file storing the parameters that are used for creating the PTXQC report and returns these parameters as well as a list of available qc-Metrics objects.*

Description

Valid parameters are: param_useMQPAR, add_fs_col, id_rate_bad, id_rate_great , pg_ratioLabIncThresh , param_PG_intThresh, param_EV_protThresh , param_EV_intThresh, param_EV_pepThresh , yaml_contaminants, param_EV_MatchingTolerance, param_evd_mbr , param_EV_PrecursorTolPPM, param_EV_PrecursorOutOfCalSD , param_EV_PrecursorTolPPMmainSearch, param_MSMSScans_ionInjThresh, param_OutputFormats and param_PageNumbers

Please provide them as a list() of this format: list$parameter_name

Usage

```
createYaml(
  yc,
  param = list(),
  DEBUG_PTXQC = FALSE,
  txt_files = NULL,
  metrics = NULL
)
```

Arguments

yc	A yaml class object created by YAMLClass$new()
param	list of parameters sorted by names; if empty, will be populated with defaults
DEBUG_PTXQC	print some debugging information; default FALSE
txt_files	list of paths to MaxQuant files; if NULL, it is assumed that the parameters are for mzTab-mode
metrics	list of metric names that should be plotted; if NULL, will be populated with defaults

Value

list of parameters used for creating the report and list of qc-Metrics objects

---

CV *Coefficient of variation (CV)*

Description

Computes sd(x) / mean(x)

Usage

```
CV(x)
```

Arguments

x	Vector of numeric values

Value

CV

---

darken *Make a color (given as name or in RGB) darker by factor x = [0 = black, 1=unchanged]*

Description

Make a color (given as name or in RGB) darker by factor x = [0 = black, 1=unchanged]

Usage

```
darken(color, factor = 0.8)
```

Arguments

color	A color as understood by col2rgb
factor	Between 0 (make black) and 1 (leave color as is)

Value

darkened color

---

del0 *Replace 0 with NA in a vector*

Description

Replace 0 with NA in a vector

Usage

```
del0(x)
```

Arguments

x	A numeric vector

Value

Vector of same size as 'x', with 0's replaced by NA

---

delLCP *Removes the longest common prefix (LCP) from a vector of strings.*

Description

You should provide only unique strings (to increase speed). If only a single string is given, the empty string will be returned unless `minOutputLength` is set.

Usage

```
delLCP(x, min_out_length = 0, add_dots = FALSE)
```

Arguments

x	Vector of strings with common prefix
min_out_length	Minimal length of the shortest element of x after LCP removal [default: 0, i.e. empty string is allowed] . If the output would be shorter, the last part of the LCP is kept.
add_dots	Prepend output with '..' if shortening was done.

Value

Shortened vector of strings

Examples

```
delLCP(c("TK12345_H1"), min_out_length=0)
## ""

delLCP(c("TK12345_H1"), min_out_length=4)
## "5_H1"

delLCP(c("TK12345_H1"), min_out_length=4, add_dots = TRUE)
## "..5_H1"

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=4)
## "5_H1" "5_H2"

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=4, add_dots = TRUE)
## "..5_H1" "..5_H2"

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=8)
## "12345_H1", "12345_H2"

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=8, add_dots = TRUE)
## "TK12345_H1", "TK12345_H2" (unchanged, since '..' would add another two)

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=60)
## "TK12345_H1", "TK12345_H2" (unchanged)

delLCP(c("TK12345_H1", "TK12345_H2"), min_out_length=60, add_dots = TRUE)
## "TK12345_H1", "TK12345_H2" (unchanged)
```

---

delLCS *Removes the longest common suffix (LCS) from a vector of strings.*

Description

Removes the longest common suffix (LCS) from a vector of strings.

Usage

```
delLCS(x)
```

Arguments

x	Vector of strings with common suffix

Value

Shortened vector of strings

Examples

```
delLCS(c("TK12345_H1"))                    ## ""
delLCS(c("TK12345_H1", "TK12345_H2"))      ## "TK12345_H1" "TK12345_H2"
delLCS(c("TK12345_H1", "TK12!45_H1"))      ## "TK123"    "TK12!"
```

`FilenameMapper-class` *Make sure to call $readMappingFile(some_file) if you want to support a user-defined file mapping. Otherwise, calls to $getShortNames() will create/augment the mapping for filenames.*

Description

Make sure to call $readMappingFile(some_file) if you want to support a user-defined file mapping. Otherwise, calls to $getShortNames() will create/augment the mapping for filenames.

Fields

`raw_file_mapping` Data.frame with columns 'from', 'to' and maybe 'best.effort' (if shorting was unsuccessful)

`mapping.creation` how the current mapping was obtained (user or auto)

`external.mapping.file` Filename of user-defined mapping file; only defined if readMapping-File() was called

Methods

`getShortNamesStatic(raw.files, max_len, fallbackStartNr = 1)` Static method: Shorten a set of Raw file names and return a data frame with the mappings. Mapping will have: $from, $to and optionally $best.effort (if shorting was unsuccessful and numbers had to be used)

- `raw.files` Vector of Raw files.
- `max_len` Maximal length of shortening results, before resorting to canonical names (file 1,...).
- `fallbackStartNr` Starting index for canonical names.

Return Value: data.frame with mapping.

Examples

```
a = FilenameMapper$new()
a$readMappingFile('filenamemapping.txt')
```

`findAlignReference` *Return list of raw file names which were reported by MaxQuant as reference point for alignment.*

Description

There is only one reference point which has '0' in 'retention.time.calibration' column in evidence.txt as corrected RT. This is true for most MaxQuant versions and also true for fractions. However, some evidence.txt files show 0.03 as an averaged minimum per Raw file. We use the raw.file with the smallest average as reference.

Usage

```
findAlignReference(data)
```

Arguments

`data`	The data.frame with columns 'retention.time.calibration' and 'raw.file'

Details

Note that MaxQuant uses a guide tree to align the Raw files, so the order of files does not influence the alignment. But the first file will always be used as reference point when reporting delta-RTs. And this file is also used by PTXQC as reference file vs all other files to find the real calibration function (see alignmentCheck()).

This function might return multiple raw file names (if MQ decides to change its mind at some point in the future). In this case the result should be treated with caution or (better) regarded as failure.

Value

List of reference raw files (usually just one)

`fixCalibration` *Detect (and fix) MaxQuant mass recalibration columns, since they sometimes report wrong values.*

Description

Returns a list of items for both diagnostics and possibly a fixed evidence data.frame. Also two strings with messages are returned, which can serve as user message for pre and post calibration status.

Usage

```
fixCalibration(
  df_evd,
  df_idrate = NULL,
  tolerance_sd_PCoutOfCal = 2,
  low_id_rate = 1
)
```

Arguments

df_evd	Evidence data.frame with columns ()
df_idrate	Data.frame from summary.txt, giving ID rates for each raw file (cols: "ms.ms.identified....", "fc.raw.file"). Can also be NULL.
tolerance_sd_PCoutOfCal	Maximal standard deviation allowed before considered 'failed'
low_id_rate	Minimum ID rate in Percent before a Raw file is considered 'failed'

Value

list of data (stats, affected_raw_files, df_evd, recal_message, recal_message_post)

---

flattenList *Flatten lists of lists with irregular depths to just a list of items, i.e. a list of the leaves (if you consider the input as a tree).*

Description

Flatten lists of lists with irregular depths to just a list of items, i.e. a list of the leaves (if you consider the input as a tree).

Usage

```
flattenList(x)
```

Arguments

x	List of 'stuff' (could be lists or items or a mix)

Value

A flat list

getAbundanceClass *Assign a relative abundance class to a set of (log10) abundance values*

Description

Abundances (should be logged already) are grouped into different levels, starting from the smallest values ("low") to the highest values ("high"). Intermediate abundances are either assigned as "mid", or "low-mid". If the range is too large, only "low" and "high" are assigned, the intermediate values are just numbers.

Usage

```
getAbundanceClass(x)
```

Arguments

x	Vector of numeric values (in log10)

Details

Example: getAbundanceClass(c(12.4, 17.1, 14.9, 12.3)) ## –> factor(c("low", "high", "mid", "low"))

Value

Vector of factors corresponding to input with abundance class names (e.g. low, high)

getECDF *Estimate the empirical density and return it*

Description

Estimate the empirical density and return it

Usage

```
getECDF(samples, y_eval = (1:100)/100)
```

Arguments

samples	Vector of input values (samples from the distribution)
y_eval	Vector of points where CDF is evaluated (each percentile by default)

Value

Data.frame with columns 'x', 'y'

Examples

```
plot(getECDF(rnorm(1e4)))
```

`getFileEncoding` *Determine if a file is 'UTF-8' or 'UTF-8-BOM' (as of MQ2.4) or 'UTF-16BE' or 'UTF-16LE'*

Description

Determine if a file is 'UTF-8' or 'UTF-8-BOM' (as of MQ2.4) or 'UTF-16BE' or 'UTF-16LE'

Usage

```
getFileEncoding(filename)
```

Arguments

`filename`	Relative or absolute path to a file

Value

'' if the file does not exist or is not readable

`getFragmentErrors` *Extract fragment mass deviation errors from a data.frame from msms.txt*

Description

Given a data.frame as obtainable from a msms.txt with - a 'mass.analyzer' column which contains only a single value for the whole column - a 'mass.deviations..da.' and (if available) 'mass.deviations..ppm.' - a 'masses' column (only required if 'mass.deviations..ppm.' is unavailable and the mass.analyzer indicates hig-res data)

Usage

```
getFragmentErrors(x, recurse = 0)
```

Arguments

`x`	Data frame in long format with numerical expression data
`recurse`	Internal usage only. Leave at 0 when calling.

Details

Mass deviations are extracted from the columns, e.g. each cell containing values separated by semicolons is split into single values. The appropriate unit is chosen (Da or ppm, depending on ITMS or FTMS data). Also the fragmentation type can be used: CID indicates ITMS, HCD to FTMS. This is not 100

Sometimes, peptides are identified purely based on MS1, i.e. have no fragments. These will be ignored.

If ppm mass deviations are not available, errors in Da will be converted to ppm using the corresponding mass values.

Value

Data frame with mass errors ('msErr') and their 'unit' (Da or ppm) or NULL (if no fragments were given)

---

getHTMLTable *Create an HTML table with an extra header row*

Description

Create an HTML table with an extra header row

Usage

```
getHTMLTable(data, caption = NA)
```

Arguments

data	A data.frame which serves as table
caption	A set of headlines, e.g. c("top line", "bottom line")

Value

table as html character string for cat()'ing into an html document

Examples

```
data = data.frame(raw.file = letters[1:4],
                  id.rate = 3:6)
getHTMLTable(data,
             caption = "some header line")
```

getMaxima — *Find the local maxima in a vector of numbers.*

Description

A vector of booleans is returned with the same length as input which contains TRUE when there is a maximum. Simply sum up the vector to get the number of maxima.

Usage

```
getMaxima(x, thresh_rel = 0.2)
```

Arguments

x	Vector of numbers
thresh_rel	Minimum relative intensity to maximum intensity of 'x' required to be a maximum (i.e., a noise threshold). Default is 20%.

Value

Vector of bool's, where TRUE indicates a local maximum.

Examples

```
r = getMaxima(c(1,0,3,4,5,0))
all(r == c(TRUE,FALSE,FALSE,FALSE,TRUE,FALSE))

getMaxima(c(1, NA, 3, 2, 3, NA, 4, 2, 5))
```

getMetaData — *Extract meta information (orderNr, metric name, category) from a list of Qc metric objects*

Description

Extract meta information (orderNr, metric name, category) from a list of Qc metric objects

Usage

```
getMetaData(lst_qcMetrics)
```

Arguments

lst_qcMetrics	List of qcMetrics

Value

data.frame with columns 'name', 'order' and 'cat' (category)

`getMetricsObjects` *Get all currently available metrics*

Description

Get all currently available metrics

Usage

```
getMetricsObjects(DEBUG_PTXQC = FALSE)
```

Arguments

`DEBUG_PTXQC`	Use qc objects from the package (FALSE) or from environment (TRUE/DEBUG)

Value

List of matric objects

`getMQPARValue` *Retrieve a parameter value from a mqpar.xml file*

Description

If the file has the param, then return it as string. If the file is missing, warning is shown and NULL is returned. If the param (i.e. XML tag) is unknown or cannot be extracted, the program will quit (since this is a hard error). When multiple occurrences of the param are found (usually due to parameter groups), we test if the values are all identical. If so, the value is returned. If the values are different, a warning is emitted and NULL is returned unless 'allow_multiple = TRUE'

Usage

```
getMQPARValue(mqpar_filename, xpath, allow_multiple = FALSE)
```

Arguments

`mqpar_filename`	Filename (incl. absolute or relative path) to the mqpar.xml file
`xpath`	An XPath to extract the content of XML tag(s), e.g. '//firstSearchTol'
`allow_multiple`	If the XPath expression returns more than one value, all values must be identical (not allowing multiple different values) or 'stop()' is called

Details

E.g. calling getMQPARValue("mqpar.xml", "//firstSearchTol") will look up the line <firstSearchTol>20</firstSearchTol> and return "20" (string!).

Value

The stored value as string(!)

`getPCA` *Create a principal component analysis (PCA) plot for the first two dimensions.*

Description

Create a principal component analysis (PCA) plot for the first two dimensions.

Usage

```
getPCA(data, do_plot = TRUE, connect_line_order = NA, gg_layer)
```

Arguments

`data`	Matrix(!) where each row is one high-dimensional point, with ncol dimensions, e.g. a mouse as an array of proteinexpressions rownames(data) give classes for colouring (can be duplicates in matrices, as opposed to data.frames)
`do_plot`	Show PCA plot? if ==2, then shows correlations plot as well
`connect_line_order`	Connect points by lines, the order is given by this vector. Default: NA (no lines)
`gg_layer`	More parameters added to a ggplot object (ggplot(x) + gg_layer)

Value

[invisible] Named list with "PCA": The PCA object as returned by `prcomp`, access $x for PC values and "plots": list of plot objects (one or two)

getPeptideCounts — *Extract the number of peptides observed per Raw file from an evidence table.*

Description

Required columns are "fc.raw.file", "modified.sequence" and "is.transferred".

Usage

```
getPeptideCounts(df_evd)
```

Arguments

df_evd	Data.frame of evidence.txt as read by MQDataReader

Details

If match-between-runs was enabled during the MaxQuant run, the data.frame returned will contain separate values for 'transferred' evidence plus an 'MBRgain' column, which will give the extra MBR evidence in percent.

Value

Data.frame with columns 'fc.raw.file', 'counts', 'category', 'MBRgain'

getProteinCounts — *Extract the number of protein groups observed per Raw file from an evidence table.*

Description

Required columns are "protein.group.ids", "fc.raw.file" and "is.transferred".

Usage

```
getProteinCounts(df_evd)
```

Arguments

df_evd	Data.frame of evidence.txt as read by MQDataReader

Details

If match-between-runs was enabled during the MaxQuant run, the data.frame returned will contain separate values for 'transferred' evidence plus an 'MBRgain' column, which will give the extra MBR evidence in percent.

Value

Data.frame with columns 'fc.raw.file', 'counts', 'category', 'MBRgain'

`getQCHeatMap` *Generate a Heatmap from a list of QC measurements.*

Description

Each list entry is a data.frame with two columns. The first one contains the Raw file name (or the short version). and should be named 'raw.file' (or 'fc.raw.file'). The second column's name must be an expression (see ?plotmath) and contains quality values in the range [0,1]. If values are outside this range, a warning is issued and values are cut to the nearest allowed value (e.g. '1.2' becomes '1'). List entries are merged and columns are ordered by name.

All substrings enclosed by 'X[0-9]*X.' will be removed (can be used for sorting columns). The resulting string is evaluated as an expression. E.g. parse(text = <colname>)

Usage

```
getQCHeatMap(lst_qcMetrics, raw_file_mapping)
```

Arguments

`lst_qcMetrics`	List of QCMetric objects
`raw_file_mapping`	Data.frame with 'from' and 'to' columns for name mapping to unify names from list entries

Details

To judge the overall quality of each raw file a summary column is added, values being the mean of all other columns per row.

Value

A ggplot object for printing

`getReportFilenames` *Assembles a list of output file names, which will be created during reporting.*

Description

You can combine **report_name_has_folder** (and **mzTab_filename** for mzTab files) to obtain report filenames which are even more robust to moving around (since they contain infixes of the mzTab filename and the folder), e.g. '@em 'report_HEK293-study_myProjects.html", where the input was 'mzTab_filename='HEK293-study.mzTab' and 'folder='c:/somePath/myProjects/'.

Usage

```
getReportFilenames(
  folder,
  report_name_has_folder = TRUE,
  mzTab_filename = NULL
)
```

Arguments

`folder`	Directory where the MaxQuant output (txt folder) or the mzTab file resides
`report_name_has_folder`	Boolean: Should the report files (html, pdf) contain the name of the deepest(=last) subdirectory in **txt_folder** which is not 'txt'? Useful for discerning different reports in a PDF viewer. E.g. when flag is FALSE: 'report_v0.91.0.html'; and 'report_v0.91.0_bloodStudy.html' when flag is TRUE (and the txt folder is '.../bloodStudy/txt/' or '...bloodStudy/')
`mzTab_filename`	If input is an mzTab, specify its name, so that the filenames can use its basename as infix E.g. when 'mzTab_filename = 'HEK293-study.mzTab'' then the output will be 'report_HEK293-study.html'. This allows to get reports on multiple mzTabs in the same folder without overwriting report results.

Value

List of output file names (just names, no file is created) with list entries: **yaml_file**, **heatmap_values_file**, **R_plots_file**, **filename_sorting**, **mzQC_file**, **log_file**, **report_file_prefix**, **report_file_PDF**, **report_file_HTML**

getRunQualityTemplate *Get an mzQC runQuality without actual metrics, but with full metadata*

Description

Get an mzQC runQuality without actual metrics, but with full metadata

Usage

```
getRunQualityTemplate(fc.raw.file, raw_file_mapping)
```

Arguments

fc.raw.file	For which run
raw_file_mapping	A data.frame with cols 'from', 'to' and maybe 'best.effort' (if shorting was unsuccessful), as e.g. obtained by a FilenameMapper$raw_file_mapping

Value

An MzQCrunQuality object

ggAxisLabels *Function to thin out the number of labels shown on an axis in GGplot*

Description

By default, 20 labels (or up to 40 see below) are shown. If the number of items is less than twice the number of desired labels, all labels will be shown (to avoid irregular holes for some labels). I.e. if n=20, and x has 22 entries, there would be only two labels removed, giving a very irregular picture. It only becomes somewhat regular if after any label there is at least one blank, i.e. at most half the entries are labeled. #' Example: ## p is any ggplot object p + scale_y_discrete(breaks = ggAxisLabels) ## customize 'n' my.ggAxisLabels = function(x) ggAxisLabels(x, n = 4) p + scale_y_discrete(breaks = my.ggAxisLabels)

Usage

```
ggAxisLabels(x, n = 20)
```

Arguments

x	Vector of labels (passed by GGplot)
n	Number of labels to show

Value

Shortened version of 'x'

ggText *Plot a text as graphic using ggplot2.*

Description

Plot a text as graphic using ggplot2.

Usage

```
ggText(title, text, col = "black")
```

Arguments

title	The title of the plot
text	Centered text, can contain linebreaks
col	Colour of text (excluding the title)

Value

ggplot object

grepv *Grep with values returned instead of indices.*

Description

The parameter 'value' should not be passed to this function since it is passed internally already.

Usage

```
grepv(reg, data, ...)
```

Arguments

reg	regex param
data	container
...	other params forwarded to grep()

Value

values of data which matched the regex

Examples

```
grepv("x", c("abc", "xyz"))
## --> "xyz"
```

`idTransferCheck` *Check how close transferred ID's after alignment are to their genuine IDs within one Raw file.*

Description

The input is a data.frame containing feature evidence with corrected retention times, e.g. a 'calibrated.retention.time' column.

Usage

```
idTransferCheck(df_evd_all)
```

Arguments

`df_evd_all`	A data.frame with columns 'type', 'calibrated.retention.time', 'modified.sequence', 'charge', 'raw.file'

Details

Note that this function must be given MS/MS identifications of type "MULTI-MSMS" and "MSMS-MATCH". It will stop() otherwise.

We compare for each peptide sequence (and charge) the RT difference within groups of either genuine as well as mixed pairs. For every comparison made, we report the RT span If alignment worked perfectly, the span are very small (<1 min), for the mixed group, i.e. the pairs are accidentally split 3D peaks. Alignment performance has no influence on the genuine-only groups.

Note: We found early MaxQuant versions (e.g. 1.2.2.5) to have an empty 'modified.sequence' column for 'MULTI-MATCH' entries. The sequence which SHOULD be present is equal to the immediate upper row. This is what we use to guess the sequence. However, this relies on the data.frame not being subsetted before (we can sort using the 'id' column)!

Value

A data.frame containing the RT diff for each ID-group found in a Raw file (bg = genuine).

`inMatchWindow` *For grouped peaks: separate them into in-width vs. out-width class.*

Description

Looking at groups only: Compute the fraction of 3D-peak pair groups per Raw file which have an acceptable RT difference after alignment using the result from 'idTransferCheck()', i.e. compute the fraction of groups which are within a certain RT tolerance.

Usage

```
inMatchWindow(data, df.allowed.deltaRT)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'rtdiff_mixed', 'rtdiff_genuine'
df.allowed.deltaRT	The allowed matching difference for each Raw file (as data.frame(fc.rawfile, m))

Details

Returned value is between 0 (bad) and 1 (all within tolerance).

Value

A data.frame with one row for each raw.file and columns 'raw.file' and score 'withinRT' (0-1)

---

lcpCount *Count the number of chars of the longest common prefix*

Description

Count the number of chars of the longest common prefix

Usage

```
lcpCount(x)
```

Arguments

x	Vector of strings with common prefix

Value

Length of LCP

LCS *Compute longest common substring of two strings.*

Description

Implementation is very inefficient (dynamic programming in R) –> use only on small instances

Usage

```
LCS(s1, s2)
```

Arguments

s1	String one
s2	String two

Value

String containing the longest common substring

lcsCount *Count the number of chars of the longest common suffix*

Description

Count the number of chars of the longest common suffix

Usage

```
lcsCount(x)
```

Arguments

x	Vector of strings with common suffix

Value

Length of LCS

LCSn *Find longest common substring from 'n' strings.*

Description

Warning: greedy heuristic! This is not guaranteed to find the best solution (or any solution at all), since its done pairwise with the shortest input string as reference.

Usage

```
LCSn(strings, min_LCS_length = 0)
```

Arguments

strings	A vector of strings in which to search for LCS
min_LCS_length	Minimum length expected. Empty string is returned if the result is shorter

Value

longest common substring (or "" if shorter than min_LCS_length)

Examples

```
LCSn(c("1_abcde...",
       "2_abcd...",
       "x_abc..."))  ## --> "_abc"
LCSn(c("16_IMU008_CISPLA_E5_R11",
       "48_IMU008_CISPLA_P4_E7_R31",
       "60_IMU008_CISPLA_E7_R11"), 3) ## -->"_IMU008_CISPLA_"
LCSn(c("AAAAACBBBBB",
       "AAAAADBBBBB",
       "AAAABBBBBEF",
       "AAABBBBBDGH")) ## -->  "BBBBB"
LCSn(c("AAAXXBBB",
       "BBBXXDDD",
       "XXAAADDD")) ## --> fails due to greedy approach; should be "XX"
```

longestCommonPrefix *Get the longest common prefix from a set of strings.*

Description

Input is converted to character (e.g. from factor) first.

Usage

```
longestCommonPrefix(strings)
```

Arguments

strings	Vector of strings

Value

Single string - might be empty ("")

Examples

```
longestCommonPrefix(c("CBA.321", "CBA.77654", ""))    ## ""
longestCommonPrefix(c("CBA.321", "CBA.77654", "CB"))  ## "CB"
longestCommonPrefix(c("ABC.123", "ABC.456"))          ## "ABC."
longestCommonPrefix(c("nothing", "in", "common"))     ## ""
```

---

longestCommonSuffix *Like longestCommonPrefix(), but on the suffix.*

Description

Like longestCommonPrefix(), but on the suffix.

Usage

```
longestCommonSuffix(strings)
```

Arguments

strings	Vector of strings

Value

Single string - might be empty ("")

Examples

```
longestCommonSuffix(c("123.ABC", "45677.ABC", "BC"))  ## "BC"
longestCommonSuffix(c("123.ABC", "", "BC"))           ## ""
longestCommonSuffix(c("123.ABC", "45677.ABC"))        ## ".ABC"
longestCommonSuffix(c("nothing", "in", "common"))     ## ""
```

modsToTable *Convert list of (mixed)modifications to a frequency table*

Description

Convert list of (mixed)modifications to a frequency table

Usage

```
modsToTable(mod_list)
```

Arguments

mod_list	A vector with modifications, each for a specific peptide. Multiple mods per entry are allowed, each separated by comma.

Value

A data.frame with 'modification_names' and 'Freq' (0-100)

Examples

```
modsToTable(c("Ox (M)",
              "Unmodified",
              "Ox (M),Acetyl (Prot N-term)",
              "2 Ox (M)",
              "Unmodified",
              "Unmodified"))
```

modsToTableByRaw *Convert list of (mixed)modifications to a frequency table*

Description

Convert list of (mixed)modifications to a frequency table

Usage

```
modsToTableByRaw(
  df_evd,
  name_unmod = "Unmodified",
  name_unmod_inverse = "Modified (total)"
)
```

Arguments

df_evd	data.frame with 'fc.raw.file' and a 'modifications' column, which contains the modifications for each peptide.
name_unmod	String in 'modifications' which represents an unmodified peptide
name_unmod_inverse	If non-empty, then inverse the frequencies of the 'name_unmod' modifications (i.e. 100-x) IFF they are >=50% on average (across Raw files) and rename them to this string

Value

A data.table with 'fc.raw.file', 'modification_names' (factor), and 'Freq' (0-100)

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B"),
                                    each = 3),
       modifications = c("Oxidation (M)",
                         "Unmodified",
                         "Oxidation (M),Acetyl (Protein N-term)",
                         "2 Oxidation (M)",
                         "Unmodified", "Unmodified"))
modsToTableByRaw(data)
```

---

mosaicize *Prepare a Mosaic plot of two columns in long format.*

Description

Found at http://stackoverflow.com/questions/19233365/how-to-create-a-marimekko-mosaic-plot-in-ggplot2 Modified (e.g. to pass R check)

Usage

```
mosaicize(data)
```

Arguments

data	A data.frame with exactly two columns

Details

Returns a data frame, which can be used for plotting and has the following columns: 'Var1' - marginalized values from 1st input column 'Var2' - marginalized values from 2nd input column 'Freq' - relative frequency of the combination given in [Var1, Var2] 'margin_var1' - frequency of the value given in Var1 'var2_height' - frequency of the value given in Var2, relative to Var1 'var1_center' - X-position when plotting (large sets get a larger share)

Value

Data.frame

Examples

```
data = data.frame(raw.file = c(rep('file A', 100), rep('file B', 40)),
                  charge = c(rep(2, 60), rep(3, 30), rep(4, 10),
                             rep(2, 30), rep(3, 7), rep(4, 3)))
mosaicize(data)
```

`MQDataReader-class` *S5-RefClass to read MaxQuant .txt files*

Description

This class is used to read MQ data tables using `MQDataReader::readMQ()` while holding the internal raw file –> short raw file name mapping (stored in a member called 'fn_map') and updating/using it every time `MQDataReader::readMQ()` is called.

Arguments

`file`	(Relative) path to a MQ txt file.
`filter`	Searched for "C" and "R". If present, [c]ontaminants and [r]everse hits are removed if the respective columns are present. E.g. to filter both, `filter = "C+R"`
`type`	Allowed values are: "pg" (proteinGroups) [default], adds abundance index columns (*AbInd*, replacing 'intensity') "sm" (summary), splits into three row subsets (raw.file, condition, total) "ev" (evidence), will fix empty modified.sequence cells for older MQ versions (when MBR is active) "msms_scans", will fix invalid (negative) scan event numbers Any other value will not add/modify any columns
`col_subset`	A vector of column names as read by read.delim(), e.g., spaces are replaced by dot already. If given, only columns with these names (ignoring lower/uppercase) will be returned (regex allowed) E.g. col_subset=c("^lfq.intensity.", "protein.name")
`add_fs_col`	If TRUE and a column 'raw.file' is present, an additional column 'fc.raw.file' will be added with common prefix AND common substrings removed (`simplifyNames`) E.g. two rawfiles named 'OrbiXL_2014_Hek293_Control', 'OrbiXL_2014_Hek293_Treated' will give 'Control', 'Treated' If `add_fs_col` is a number AND the longest shortname is still longer, the names are discarded and replaced by a running ID of the form 'file <x>', where <x> is a number from 1 to N. If the function is called again and a mapping already exists, this mapping is used. Should some raw.files be unknown (ie the mapping from the previous file is incomplete), they will be augmented
`check_invalid_lines`	After reading the data, check for unusual number of NA's to detect if file was corrupted by Excel or alike

LFQ_action	[For type=='pg' only] An additional custom LFQ column ('cLFQ...') is created where zero values in LFQ columns are replaced by the following method IFF(!) the corresponding raw intensity is >0 (indicating that LFQ is erroneusly 0) "toNA": replace by NA "impute": replace by lowest LFQ value >0 (simulating 'noise')
...	Additional parameters passed on to read.delim()
colname	Name of the column (e.g. 'contaminants') in the mq.data table
valid_entries	Vector of values to be replaced (must contain all values expected in the column – fails otherwise)
replacements	Vector of values inserted with the same length as valid_entries.

Details

Since MaxQuant changes capitalization and sometimes even column names, it seemed convenient to have a function which just reads a txt file and returns unified column names, irrespective of the MQ version. So, it unifies access to columns (e.g. by using lower case for ALL columns) and ensures columns are identically named across MQ versions:

```
alternative term          new term
-----------------------------------------
protease                  enzyme
protein.descriptions      fasta.headers
potential.contaminant     contaminant
mass.deviations           mass.deviations..da.
basepeak.intensity        base.peak.intensity
```

We also correct 'reporter.intensity.*' naming issues to MQ 1.6 convention, when 'reporter.intensity.not.corrected' is present. MQ 1.5 uses: reporter.intensity.X and reporter.intensity.not.corrected.X MQ 1.6 uses: reporter.intensity.X and reporter.intensity.corrected.X

Note: you must find a regex which matches both versions, or explicitly add both terms if you are requesting only a subset of columns!

Fixes for msmsScans.txt: negative Scan Event Numbers in msmsScans.txt are reconstructed by using other columns

Automatically detects UTF8-BOM encoding and deals with it (since MQ2.4).

Example of usage:

```
 mq = MQDataReader$new()
d_evd = mq$readMQ("evidence.txt", type="ev", filter="R", col_subset=c("proteins", "Retention.Length"
```

If the file is empty, this function shows a warning and returns NULL. If the file is present but cannot be read, the program will stop.

Wrapper to read a MQ txt file (e.g. proteinGroups.txt).

Value

A data.frame of the respective file

Replaces values in the mq.data member with (binary) values. Most MQ tables contain columns like 'contaminants' or 'reverse', whose values are either empty strings or "+", which is inconvenient and can be much better represented as TRUE/FALSE. The params `valid_entries` and `replacements` contain the matched pairs, which determine what is replaced with what.

Returns `TRUE` if successful.

Methods

`getInvalidLines()` Detect broken lines (e.g. due to Excel import+export)

When editing a MQ txt file in Microsoft Excel, saving the file can cause it to be corrupted, since Excel has a single cell content limit of 32k characters (see http://office.microsoft.com/en-001/excel-help/excel-specifications-and-limits-HP010342495.aspx) while MQ can easily reach 60k (e.g. in oxidation sites column). Thus, affected cells will trigger a line break, effectively splitting one line into two (or more).

If the table has an 'id' column, we can simply check the numbers are consecutive. If no 'id' column is available, we detect line-breaks by counting the number of NA's per row and finding outliers. The line break then must be in this line (plus the preceeding or following one). Depending on where the break happened we can also detect both lines right away (if both have more NA's than expected).

Currently, we have no good strategy to fix the problem since columns are not aligned any longer, which leads to columns not having the class (e.g. numeric) they should have. (thus one would need to un-do the linebreak and read the whole file again)

[Solution to the problem: try LibreOffice 4.0.x or above – seems not to have this limitation]

@return Returns a vector of indices of broken (i.e. invalid) lines

---

`MzTabReader-class` *Class to read an mzTab file and store the tables internally.*

Description

The 'sections' field is initialized after $readMzTab was called. The 'fn_map' fields should be initialized via ...fn_mapreadMappingFile(...) manually if user-defined filename mappings are desired and is automatically updated/queried when $readMzTab is called.

Fields

`sections` MzTab sections as list. Valid list entries are: "MTD", "PRT", "PEP", "PSM", "SML", "filename" and "comments"

`fn_map` FilenameMapper which can translate raw filenames into something shorter

Methods

`RTUnitCorrection(dt)` Convert all RT columns from seconds (OpenMS default) to minutes (MaxQuant default)

`getEvidence()` Basically the PSM table and additionally columns named 'raw.file' and 'fc.raw.file'.

`getMSMSScans(identified_only = FALSE)` Basically the PSM table (partially renamed columns) and additionally two columns 'raw.file' and 'fc.raw.file'. If identified_only is TRUE, only MS2 scans which were identified (i.e. a PSM) are returned – this is equivalent to msms.txt in MaxQuant.

`getParameters()` Converts internal mzTab metadata section to a two column key-value data.frame similar to MaxQuants parameters.txt.

`getProteins()` Basically the PRT table ...

`getSummary()` Converts internal mzTab metadata section to a two data.frame with columns 'fc.raw.file', 'ms.ms.identified....' similar to MaxQuants summary.txt.

`renameColumns(dt, namelist)` Renames all columns and throws a warning if a column does not exist in the data

---

pasten *paste with newline as separator*

Description

paste with newline as separator

Usage

```
pasten(...)
```

Arguments

...	Arguments forwarded to paste()

Value

return value of paste()

Examples

```
pasten("newline","separated")
## --> "newline\nseparated"
```

pastet *paste with tab as separator*

Description

paste with tab as separator

Usage

```
pastet(...)
```

Arguments

...	Arguments forwarded to paste()

Value

return value of paste()

Examples

```
pastet("tab","separated")
## --> "tab\tseparated"
```

peakSegmentation *Determine fraction of evidence which causes segmentation, i.e. sibling peaks at different RTs confirmed either by genuine or transferred MS/MS.*

Description

Sometimes, MQ splits a feature into 2 or more if the chromatograpic conditions are not optimal and there is a drop in RT intensity. If both features contain successful MS/MS scans, we will find the same peptide twice (with slightly different RT) in the same charge state. This constitutes a natively split peak and is rare (95

Usage

```
peakSegmentation(df_evd_all)
```

Arguments

df_evd_all	A data.frame of evidences containing the above columns

Details

If Match-between-runs is used and the RT alignment is not perfect, then a peptide might be inferred at a wrong RT position, even though this Raw file already contains MS/MS evidence of this peptide. Usually the number of peak duplicates rises drastically (e.g. only 75 In most cases, the RT is too far off to be a split peak. It's rather a lucky hit with accidentally the same mass-to-charge, and thus the intensity is random. To find by how much these peak pairs differ in RT, use idTransferCheck() and inMatchWindow().

Required columns are 'is.transferred', 'fc.raw.file', 'modified.sequence', 'charge', 'type'.

Note that this function must be given MS/MS identifications of type "MULTI-MSMS" and "MSMS-MATCH". It will stop() otherwise.

Value

A data.frame with one row per Raw file and three columns: 1) 2) 3)

---

peakWidthOverTime *Discretize RT peak widths by averaging values per time bin.*

Description

Should be applied for each Raw file individually.

Usage

```
peakWidthOverTime(data, RT_bin_width = 2)
```

Arguments

data	Data.frame with columns 'retention.time' and 'retention.length'
RT_bin_width	Bin size in minutes

Details

Returns a data.frame, where 'bin' gives the index of each bin, 'RT' is the middle of each bin and 'peakWidth' is the averaged peak width per bin.

Value

Data.frame with columns 'bin', 'RT', 'peakWidth'

Examples

```
 data = data.frame(retention.time = seq(30,200, by=0.001)) ## one MS/MS per 0.1 sec
data$retention.length = seq(0.3, 0.6, length.out = nrow(data)) + rnorm(nrow(data), 0, 0.1)
 d = peakWidthOverTime(data)
 plot(d$RT, d$peakWidth)
```

plotTable *Plot a table with row names and title*

Description

Restriction: currently, the footer will be cropped at the table width.

Usage

```
plotTable(
  data,
  title = "",
  footer = "",
  col_names = colnames(data),
  fill = c("grey90", "grey70"),
  col = "black",
  just = "centre"
)
```

Arguments

data	A data.frame with columns as described above
title	Table title
footer	Footer text
col_names	Column names for Table
fill	Fill pattern (by row)
col	Text color (by column)
just	(ignored)

Value

gTree object with class 'PTXQC_table'

Examples

```
data = data.frame(raw.file = letters[1:4],
                  id.rate = 3:6)
plotTable(data,
          title = "Bad files",
          footer = "bottom",
          col_names = c("first col", "second col"),
          col=c("red", "green"))
```

plotTableRaw *Colored table plot.*

Description

Code taken from http://stackoverflow.com/questions/23819209/change-text-color-for-cells-using-tablegrob-in-r

Usage

```
plotTableRaw(data, colours = "black", fill = NA, just = "centre")
```

Arguments

data	Table as Data.frame
colours	Single or set of colours (col-wise)
fill	Cell fill (row-wise)
just	(ignored)

Value

gTable

plot_CalibratedMSErr *Plot bargraph of uncalibrated mass errors for each Raw file.*

Description

Boxes are optionally colored to indicate that a MQ bug was detected or if PTXQC detected a too narrow search window.

Usage

```
plot_CalibratedMSErr(
  data,
  MQBug_raw_files,
  stats,
  y_lim,
  extra_limit = NA,
  title_sub = ""
)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'mass.error..ppm.'
MQBug_raw_files	List of Raw files with invalid calibration values
stats	A data.frame with columns 'fc.raw.file', 'outOfCal'
y_lim	Range of y-axis
extra_limit	Position where a v-line is plotted (for visual guidance)
title_sub	Subtitle

Value

GGplot object

Examples

```
n = c(150, 1000, 1000, 1000)
data = data.frame(fc.raw.file = repEach(letters[4:1], n),
                  mass.error..ppm. = c(rnorm(n[1], 1, 2.4),
                                       rnorm(n[2], 0.5, 0.5),
                                       rnorm(n[3], 0.1, 0.7),
                                       rnorm(n[4], 0.3, 0.8)))
stats = data.frame(fc.raw.file = letters[4:1],
                   sd = c(2.4, 0.5, 0.7, 0.8),
                   outOfCal = c(TRUE, FALSE, FALSE, FALSE))
plot_CalibratedMSErr(data, MQBug_raw_files = letters[1], stats, y_lim = c(-20,20), 15, "subtitle")
```

---

plot_Charge — *The plots shows the charge distribution per Raw file. The output of 'mosaicize()' can be used directly.*

Description

The input is a data.frame with columns 'Var1' - name of the Raw file 'Var2' - charge (used as fill color) 'Var1_center' - contains X-position of the Raw file 'Var2_height' - relative frequency of the charge 'Margin_var1' - where each row represents one peptide sequence.

Usage

```
plot_Charge(d_charge)
```

Arguments

d_charge	A data.frame with columns as described above

Value

GGplot object

Examples

```
data = data.frame(raw.file = c(rep('file A', 100), rep('file B', 40)),
                      data = c(rep(2, 60), rep(3, 30), rep(4, 10),
                               rep(2, 30), rep(3, 7), rep(4, 3)))
plot_Charge(mosaicize(data))
```

`plot_ContEVD` *Plot contaminants from evidence.txt, broken down into top5-proteins.*

Description

Plot contaminants from evidence.txt, broken down into top5-proteins.

Usage

```
plot_ContEVD(data, top5)
```

Arguments

`data`	A data.frame with columns 'fc.raw.file', 'contaminant', 'pname', 'intensity'
`top5`	Name of the Top-5 Proteins (by relative intensity or whatever seems relevant)

Value

GGplot object

Examples

```
data = data.frame(intensity = 1:12,
                  pname = rep(letters[1:3], 4),
                  fc.raw.file = rep(paste("f", 1:4), each=3),
                  contaminant = TRUE)
## providing more proteins than present... d,e will be ignored
plot_ContEVD(data, top5 = letters[1:5])
## classify 'c' as 'other'
plot_ContEVD(data, top5 = letters[1:2])
```

plot_ContsPG *Plot contaminants from proteinGroups.txt*

Description

Plot contaminants from proteinGroups.txt

Usage

```
plot_ContsPG(data)
```

Arguments

data	A data.frame with columns 'group', 'cont_pc', 'logAbdClass'

Value

GGplot object

Examples

```
data = data.frame( 'group' = letters[1:10], 'cont_pc' = 2:11, 'logAbdClass' = c("low","high"))
plot_ContsPG(data)
```

plot_ContUser *Plot user-defined contaminants from evidence.txt*

Description

Kolmogorov-Smirnoff p-values are plotted on top of each group. High p-values indicate that Andromeda scores for contaminant peptides are equal or higher compared to sample peptide scores, i.e. the probability that sample peptides scores are NOT greater than contaminant peptide scores.

Usage

```
plot_ContUser(data, name_contaminant, extra_limit, subtitle = NULL)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'variable', 'value'
name_contaminant	Name of the contaminant shown in title
extra_limit	Position where a h-line is plotted (for visual guidance)
subtitle	Optional subtitle for plot

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = letters[1:3],
                  variable = c(rep("spectralCount", 3),
                               rep("intensity", 3),
                               rep("above.thresh", 3),
                               rep("score_KS", 3)),
                  value = c(10, 20, 15, 9, 21, 14, 0, 1, 1, 0.3, 0.01, 0.04))
plot_ContUser(data, "myco", 5, "subtitle")
```

---

plot_ContUserScore *Plot Andromeda score distribution of contaminant peptide vs. matrix peptides.*

Description

The data is expected to be an ECDF already, x being the Andromeda score, y being the culmulative probability. The Score is the probability of a Kolm.-Smirnoff test that the contaminant scores are larger (i.e. large p-values indicate true contamination). You will only see this plot if the but high-scoring contaminant peptides, which would erroneously give you a large p-value and make you believe your sample is contaminated although that's not the case.

Usage

```
plot_ContUserScore(data, raw.file, score)
```

Arguments

data	A data.frame with columns 'x', 'y', 'condition'
raw.file	Name of Raw file for which the data is displayed (will become part of the plot title)
score	Score of how distinct the distributions are (will become part of the title)

Value

GGplot object

Examples

```
data = data.frame(x = 10:60,
                  y = c(seq(0,1,length=51), seq(0.1, 1, length=51)),
                  condition = rep(c("sample","contaminant"), each=51))
plot_ContUserScore(data, 'test file', 0.96)
```

plot_CountData *Plot Protein groups per Raw file*

Description

The input is a data.frame with protein/peptide counts, where 'category' designates the origin of information (genuine ID, transferred ID, or both).

Usage

```
plot_CountData(data, y_max, thresh_line, title)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'counts', 'category'
y_max	Plot limit of y-axis
thresh_line	Position of a threshold line, indicating the usual target value
title	Main title, and optional subtitle (if vector of length 2 is provided)

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B"), each=3),
                  counts = c(3674, 593, 1120, 2300, 400, 600),
                  category = c("genuine","genuine+transferred","transferred"))
plot_CountData(data, 6000, 4000, c("EVD: Protein Groups count", "gain: 23%"))
```

plot_DataOverRT *Plot some count data over time for each Raw file.*

Description

The input is a data.frame with columns 'RT' - RT in seconds, representing one bin 'counts' - number of counts at this bin 'fc.raw.file' - name of the Raw file where each row represents one bin in RT.

Usage

```
plot_DataOverRT(
  data,
  title,
  y_lab,
  x_lim = range(data$RT),
  y_max = max(data$counts)
)
```

Arguments

data	A data.frame with columns as described above
title	The plot title
y_lab	Label of y-axis
x_lim	Limits of the x-axis (2-tuple)
y_max	Maximum of the y-axis (single value)

Details

At most nine(!) Raw files can be plotted. If more are given, an error is thrown.

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(paste('file', letters[1:3]), each=30),
                          RT = seq(20, 120, length.out = 30),
                  counts = c(rnorm(30, 400, 20), rnorm(30, 250, 15), rnorm(30, 50, 15)))
plot_DataOverRT(data, "some title", "count data")
```

---

plot_IDRate *Plot percent of identified MS/MS for each Raw file.*

Description

Useful for a first overall impression of the data.

Usage

```
plot_IDRate(data, id_rate_bad, id_rate_great, label_ID)
```

Arguments

data	A data.frame with columns as described above
id_rate_bad	Number below which the ID rate is considered bad
id_rate_great	Number above which the ID rate is considered great
label_ID	Named vector with colors for the categories given in data$cat

Details

The input is a data.frame with columns 'fc.raw.file' - name of the Raw file 'ms.ms.identified....' - fraction of identified MS/MS spectra in percent 'cat' - identification category as arbitrary string where each row represents one Raw file.

Value

GGplot object

Examples

```
id_rate_bad = 20; id_rate_great = 35;
label_ID = c("bad (<20%)" = "red", "ok (...)" = "blue", "great (>35%)" = "green")
data = data.frame(fc.raw.file = paste('file', letters[1:3]),
                  ms.ms.identified.... = rnorm(3, 25, 15))
data$cat = factor(cut(data$ms.ms.identified....,
                      breaks=c(-1, id_rate_bad, id_rate_great, 100),
                      labels=names(label_ID)))
plot_IDRate(data, id_rate_bad, id_rate_great, label_ID)
```

---

plot_IDsOverRT *Plot IDs over time for each Raw file.*

Description

Uses plot_DataOverRT() internally.

Usage

```
plot_IDsOverRT(data, x_lim = range(data$RT), y_max = max(data$counts))
```

Arguments

data	A data.frame with columns as described above
x_lim	Limits of the x-axis (2-tuple)
y_max	Maximum of the y-axis (single value)

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(paste('file', letters[1:3]), each=30),
                           RT = seq(20, 120, length.out = 30),
                  counts = c(rnorm(30, 400, 20), rnorm(30, 250, 15), rnorm(30, 50, 15)))
plot_IDsOverRT(data)
```

plot_IonInjectionTimeOverRT *Plot line graph of TopN over Retention time.*

Description

Number of Raw files must be 6 at most. Function will stop otherwise.

Usage

```
plot_IonInjectionTimeOverRT(data, stats, extra_limit)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'rRT', 'medIIT'
stats	A data.frame with columns 'fc.raw.file', 'mean'
extra_limit	Visual guidance line (maximum acceptable IIT)

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("d","a","x"), each=100),
                  rRT = seq(20, 120, length.out = 100),
                  medIIT = c(round(runif(100, min=3, max=5)),
                             round(runif(100, min=5, max=8)),
                             round(runif(100, min=1, max=3)))
                  )
stats = data.frame(fc.raw.file = c("d","a","x"),
                   mean = c(4, 6.5, 2))
plot_IonInjectionTimeOverRT(data, stats, 10)
```

plot_MBRAlign *Plot MaxQuant Match-between-runs alignment performance.*

Description

The plots shows the correction function applied by MaxQuant, and the residual RT (ideally 0) of each peptide to its reference. Uncalibrated peptides are shown in red, calibrated ones in green. The MaxQuant RT correction which was applied prior is shown in blue. The range of this function can give hints if the allowed RT search window (20min by default) is sufficient or if MaxQuant should be re-run with more tolerant settings.

Usage

```
plot_MBRAlign(data, y_lim, title_sub, match_tol)
```

Arguments

data	A data.frame with columns as described above
y_lim	Plot range of y-axis
title_sub	Subtitle
match_tol	Maximal residual RT delta to reference (usually ~1 min)

Details

The input is a data.frame with columns 'calibrated.retention.time' - resulting (hopefully) calibratated RT after MQ-recal (the X-axis of the plot) 'retention.time.calibration' - delta applied by MaxQuant 'rtdiff' - remaining RT diff to reference peptide of the same sequence 'RTdiff_in' - is the feature aligned (within 'match_tol')? 'fc.raw.file_ext' - raw file where each row represents one peptide whose RT was corrected by MaxQuant.

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file_ext = "file A", ## more than one would be possible
                  calibrated.retention.time = c(20:100),
                  retention.time.calibration = 6 + sin((20:100)/10))
data$rtdiff = rnorm(nrow(data))
data$RTdiff_in = c("green", "red")[1 + (abs(data$rtdiff) > 0.7)]

plot_MBRAlign(data, c(-10, 10), "fancy subtitle", 0.7)
```

---

plot_MBRgain *Plot MaxQuant Match-between-runs id transfer performance.*

Description

The plots shows the different categories of peak classes

Usage

```
plot_MBRgain(data, title_sub = "")
```

Arguments

data	A data.frame with columns as described above
title_sub	Subtitle text

Details

The input is a data.frame with columns 'fc.raw.file' - raw file name 'single' - fraction of peptides with are represent only once 'multi.inRT' - fraction of peptides with are represent multiple times, but within a certain RT peak width 'multi.outRT' - fraction of peptides with are represent multiple times, with large RT distance 'sample' - raw file where each row represents one peptide sequence.

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = paste("file", letters[1:4]),
                  abs = c(5461, 5312, 3618, 502),
                  pc = c(34, 32, 22, 2))
plot_MBRgain(data, "MBR gain: 18%")
```

`plot_MBRIDtransfer` *Plot MaxQuant Match-between-runs id transfer performance.*

Description

The plots shows the different categories of peak classes

Usage

```
plot_MBRIDtransfer(data)
```

Arguments

`data`	A data.frame with columns as described above

Details

The input is a data.frame with columns 'fc.raw.file' - raw file name 'single' - fraction of peptides with are represent only once 'multi.inRT' - fraction of peptides with are represent multiple times, but within a certain RT peak width 'multi.outRT' - fraction of peptides with are represent multiple times, with large RT distance 'sample' - raw file where each row represents one peptide sequence.

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B"), each = 3),
                  single = c(0.9853628, 0.8323160, 0.9438375,
                             0.9825538, 0.8003763, 0.9329961),
                  multi.inRT = c(0.002927445, 0.055101018, 0.017593087,
                                 0.005636457, 0.099640044, 0.031870056),
                  multi.outRT = c(0.01170978, 0.11258294, 0.03856946,
                                  0.01180972, 0.09998363, 0.03513386),
                  sample = rep(c("genuine", "transferred", "all"), 2))
plot_MBRIDtransfer(data)
```

`plot_MissedCleavages` *Plot bargraph of missed cleavages.*

Description

Per Raw file, an arbitrary number of missed cleavage classes (one per column) can be given. The total fraction of 3D-peaks must sum to 1 (=100 Columns are ordered by name.

Usage

```
plot_MissedCleavages(data, title_sub = "")
```

Arguments

`data`	A data.frame with columns 'fc.raw.file', '...' (missed cleavage classes)
`title_sub`	Plot's subtitle

Details

A visual threshold line is drawn at 75

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = letters[1:5],
                  MC0 = c(0.8, 0.5, 0.85, 0.2, 0.9),
                  MC1 = c(0.1, 0.4, 0.05, 0.7, 0.0),
                  "MS2+" =  c(0.1, 0.1, 0.1, 0.1, 0.1),
                  check.names = FALSE)
plot_MissedCleavages(data, "contaminant inclusion unknown")
```

plot_MS2Decal *Plot bargraph of oversampled 3D-peaks.*

Description

Per Raw file, at most three n's must be given, i.e. the fraction of 3D-peaks for n=1, n=2 and n=3(or more). The fractions must sum to 1 (=100

Usage

```
plot_MS2Decal(data)
```

Arguments

data	A data.frame with columns 'file', 'msErr', 'type'

Value

GGplot object

Examples

```
n = c(100, 130, 50)
data = data.frame(file = repEach(paste(letters[1:3],"\nLTQ [Da]"), n),
                  msErr = c(rnorm(n[1], 0.5), rnorm(n[2], 0.0), rnorm(n[3], -0.5)),
                  type = c("forward", "decoy")[1+(runif(sum(n))>0.95)])
plot_MS2Decal(data)
```

plot_MS2Oversampling *Plot bargraph of oversampled 3D-peaks.*

Description

Per Raw file, at most three n's must be given, i.e. the fraction of 3D-peaks for n=1, n=2 and n=3(or more). The fractions must sum to 1 (=100

Usage

```
plot_MS2Oversampling(data)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'n', 'fraction'

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(letters[1:3], each=3),
                  n = 1:3,
                  fraction = c(0.8, 0.1, 0.1, 0.6, 0.3, 0.1, 0.7, 0.25, 0.05))
plot_MS2Oversampling(data)
```

---

plot_peptideMods *Plot peptide modification frequencies*

Description

The input is a data.frame, as obtained from modsToTableByRaw().

Usage

```
plot_peptideMods(tbl, y_max = NA, show_missing_modification_levels = TRUE)
```

Arguments

tbl	A data.frame with 'fc.raw.file', 'modification_names' (can be a factor), and 'Freq' (0-100)
y_max	The upper limit of the y-axis's (==Freq); useful for multiple plots with identical limits; if 'NA' the limit is computed from the given 'tbl'
show_missing_modification_levels	If 'tbl$modification_names' is a factor and has more (but missing) levels than actually used, should missing values be dropped or assumed as '0' frequency?

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B"), each=3),
                  modifications = c("Oxidation (M)",
                  "Unmodified",
                  "Oxidation (M), Acetyl (Protein N-term)",
                  "2 Oxidation (M)",
                  "Unmodified",
                  "Unmodified"))
tbl = modsToTableByRaw(data)
plot_peptideMods(tbl,show_missing_modification_levels = TRUE)
```

plot_RatiosPG *Plot ratios of labeled data (e.g. SILAC) from proteinGroups.txt*

Description

The 'x' values are expected to be log2() transformed already.

Usage

```
plot_RatiosPG(df_ratios, d_range, main_title, main_col, legend_title)
```

Arguments

df_ratios	A data.frame with columns 'x', 'y', 'col', 'ltype'
d_range	X-axis range of plot
main_title	Plot title
main_col	Color of title
legend_title	Legend text

Value

GGplot object

Examples

```
x1 = seq(-3, 3, by = 0.1)
y1 = dnorm(x1)
x2 = seq(-5, 1, by = 0.1)
y2 = dnorm(x2, mean = -1)
data = data.frame( x = c(x1,x2),
                   y = c(y1,y2),
                   col = c(rep("ok", length(x1)), rep("shifted", length(x2))),
                   ltype = c(rep("solid", length(x1)), rep("dotted", length(x2))))
plot_RatiosPG(data, range(data$x), "Ratio plot", "red", "group")
```

plot_RTPeakWidth *Plot RT peak width over time*

Description

The input is a data.frame with already averaged counts over binned RT-slices.

Usage

```
plot_RTPeakWidth(data, x_lim, y_lim)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'RT', 'peakWidth'
x_lim	Plot range of x-axis
y_lim	Plot range of y-axis

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B", "file C"), each=81),
                  RT = c(20:100),
                peakWidth = c(rnorm(81, mean=20), rnorm(81, mean=10), rnorm(81, mean=30)))
plot_RTPeakWidth(data, c(10, 100), c(0, 40))
```

---

plot_ScanIDRate *Plot line graph of TopN over Retention time.*

Description

Number of Raw files must be 6 at most. Function will stop otherwise.

Usage

```
plot_ScanIDRate(data)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'scan.event.number', 'ratio', 'count'

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = factor(rep(c("d","a","x"), each=10), levels = c("d","a","x")),
                  scan.event.number = 1:10,
                  ratio = seq(40, 20, length.out=10),
                  count = seq(400, 200, length.out=10))
plot_ScanIDRate(data)
```

plot_TIC *Plot Total Ion Count over time*

Description

The input is a data.frame with already averaged counts over binned RT-slices.

Usage

```
plot_TIC(data, x_lim, y_lim)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'RT', 'intensity'
x_lim	Plot range of x-axis
y_lim	Plot range of y-axis

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("file A", "file B", "file C"), each=81),
                  RT = c(20:100),
                intensity = c(rnorm(81, mean=20), rnorm(81, mean=10), rnorm(81, mean=30)))
plot_TIC(data, c(10, 100), c(0, 40))
```

plot_TopN *Plot line graph of TopN over Retention time.*

Description

Number of Raw files must be 6 at most. Function will stop otherwise.

Usage

```
plot_TopN(data)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'scan.event.number', 'n'

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(c("d","a","x"), each=10),
                  scan.event.number = 1:10,
                  n = 11:20)
plot_TopN(data)
```

`plot_TopNoverRT` *Plot line graph of TopN over Retention time.*

Description

Number of Raw files must be 6 at most. Function will stop otherwise.

Usage

```
plot_TopNoverRT(data)
```

Arguments

`data`	A data.frame with columns 'fc.raw.file', 'rRT', 'topN'

Value

GGplot object

Examples

```
data = data.frame(fc.raw.file = rep(letters[1:3], each=100),
                  rRT = seq(20, 120, length.out = 100),
                  topN = c(round(runif(100, min=3, max=5)),
                           round(runif(100, min=5, max=8)),
                           round(runif(100, min=1, max=3)))
                  )
plot_TopNoverRT(data)
```

plot_UncalibratedMSErr *A boxplot of uncalibrated mass errors for each Raw file.*

Description

Boxes are optionally colored to indicate that a MQ bug was detected or if PTXQC detected a too narrow search window.

Usage

```
plot_UncalibratedMSErr(
  data,
  MQBug_raw_files,
  stats,
  y_lim,
  extra_limit,
  title_sub
)
```

Arguments

data	A data.frame with columns 'fc.raw.file', 'uncalibrated.mass.error..ppm.'
MQBug_raw_files	List of Raw files with invalid calibration values
stats	A data.frame with columns 'fc.raw.file', 'sd', 'outOfCal'
y_lim	Range of y-axis
extra_limit	Position where a v-line is plotted (for visual guidance)
title_sub	Subtitle

Value

GGplot object

Examples

```
n = c(150, 1000, 1000, 1000)
data = data.frame(fc.raw.file = repEach(letters[4:1], n),
                  uncalibrated.mass.error..ppm. = c(rnorm(n[1], 13, 2.4),
                                                    rnorm(n[2], 1, 0.5),
                                                    rnorm(n[3], 3, 0.7),
                                                    rnorm(n[4], 4.5, 0.8)))
stats = data.frame(fc.raw.file = letters[4:1],
                   sd_uncal = c(2.4, 0.5, 0.7, 0.8),
                   outOfCal = c(TRUE, FALSE, FALSE, FALSE))
plot_UncalibratedMSErr(data, MQBug_raw_files = letters[1],
                       stats, y_lim = c(-20,20), 15, "subtitle")
```

pointsPutX *Distribute a set of points with fixed y-values on a stretch of the x-axis.*

Description

#' Usage: ggplot(...) + geom_X(...) + pointsPutX(...)

Usage

```
pointsPutX(x_range, x_section, y, col = NA)
```

Arguments

x_range	[min,max] valid range of x-values
x_section	[min,max] fraction in which to distribute the values (in [0,1] for min,max, e.g. c(0.03,0.08) for 3-8%)
y	Y-values
col	Colour of the points (used as argument to aes(colour=))

Value

ggplot object with new geom_point

print.PTXQC_table *helper S3 class, enabling print(some-plot_Table-object)*

Description

helper S3 class, enabling print(some-plot_Table-object)

Usage

```
## S3 method for class 'PTXQC_table'
print(x, ...)
```

Arguments

x	Some Grid object to plot
...	Further arguments (not used, but required for consistency with other print methods)

printWithFooter *Augment a ggplot with footer text*

Description

Augment a ggplot with footer text

Usage

```
printWithFooter(gg_obj, bottom_left = NULL, bottom_right = NULL)
```

Arguments

gg_obj	ggplot2 object to be printed
bottom_left	Footer text for bottom left side
bottom_right	Footer text for bottom right side

Value

-

QCMetaFilenames *Define a Singleton class which holds the full raw filenames (+path) and their PSI-MS CV terms for usage in the mzQC metadata*

Description

Define a Singleton class which holds the full raw filenames (+path) and their PSI-MS CV terms for usage in the mzQC metadata

Define a Singleton class which holds the full raw filenames (+path) and their PSI-MS CV terms for usage in the mzQC metadata

Super class

R6P::Singleton -> QCMetaFilenames

Public fields

data Stores the data of the singleton. Set the data once before using the singleton all over the place

Methods

Public methods:

- QCMetaFilenames$clone()

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QCMetaFilenames$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`qcMetric-class` *Class which can compute plots and generate mzQC output (usually for a single metric).*

Description

Reference class which is instanciated with a metric description and a worker function (at initialization time, i.e. in the package) and can produce plots and mzQC values (at runtime, when data is provided) using setData().

Fields

`helpText` Description (lengthy) of the metric and plot elements

`workerFcn` Function which generates a result (usually plots). Data is provided using setData().

`plots` List of plots (after setData() was called)

`qcScores` Data.frame of scores from a qcMetric (computed within workerFcn())

`mzQC` An named list of mzQC MzQCqualityMetric's (named by their fc.raw.file for runQuality or concatenated fc.raw.files for setQualities (e.g. "file 1;file4")) (valid after setData() was called)

`qcCat` QC category (LC, MS, or prep)

`qcName` Name of the qcScore in the heatmap

`orderNr` Column index during heatmap generation and for the general order of plots

Examples

```
require(ggplot2)
dd = data.frame(x=1:10, y=11:20)
a = qcMetric$new(helpText="small help text",
                 ## arbitrary arguments, matched during setData()
                 workerFcn=function(.self, data, gtit)
                 {
                   ## usually some code here to produce ggplots
                   pl = lapply(1:2, function(xx) {
                       ggplot(data) +
                         geom_point(aes(x=x*xx,y=y)) +
```

```
                        ggtitle(gtit)
                    })
                  return(list(plots = pl))
                },
                qcCat="LC",
                qcName="MS/MS Peak shape",
                orderNr = 30)
## test some output
a$setData(dd, "my title")
a$plots  ## the raw plots
a$getPlots(TRUE) ## same as above
a$getPlots(FALSE) ## plots without title
a$getTitles()  ## get the titles of the all plots
a$helpText
a$qcName
```

---

qcMetric_MSMSScans_TopNoverRT-class *Metric for msmsscans.txt, showing TopN over RT.*

Description

Metric for msmsscans.txt, showing TopN over RT.

---

qualBestKS *From a list of vectors, compute all vs. all Kolmogorov-Smirnoff distance statistics (D)*

Description

... and report the row of the matrix which has maximum sum (i.e the best "reference" distribution). The returned data.frame has as many rows as distributions given and two columns. The first column 'name' gives the name of the list element, the second column 'ks_best' gives '1-statistic' of the Kolmogorov-Smirnoff test to the "reference" distribution (which was picked by maximising the sum of 'ks_best'). Thus, the row with a 'ks_best' of 1 is the reference distribution.

Usage

```
qualBestKS(x)
```

Arguments

x	List of vectors, where each vector holds a distribution

Value

A data.frame with ks-test values of the "reference" to all other distributions (see Details)

`qualCentered` *Quality metric for 'centeredness' of a distribution around zero.*

Description

Ranges between 0 (worst score) and 1 (best score). A median of zero gives the best score of 1. The closer the median is to the most extreme value of the distribution, the smaller the score (until reaching 0). Can be used for calibrated mass errors, as a measure of how well they are centered around 0. E.g. if the median is 0.1, while the range is [-0.5,0.5], the score will be 0.8 (punishing the 20 If the range of data is asymmetric, e.g. [-1.5,-0.5] and does not include zero, the score cannot reach 1, since the median can never be zero.

Usage

```
qualCentered(x)
```

Arguments

`x`	Numeric values (e.g. ppm errors)

Value

Value between [0, 1]

`qualCenteredRef` *Quality metric for 'centeredness' of a distribution around zero with a user-supplied range threshold.*

Description

Ranges between 0 (worst score) and 1 (best score). The best score is achieved when the median of 'x' is close to the center of the interval [-tol, tol]. If median of 'x' is close to the border (on either side), the score decreases linearly to zero. Can be used for uncalibrated mass errors, as a measure of how well they are centered around 0.

Usage

```
qualCenteredRef(x, tol)
```

Arguments

`x`	Vector of values (hopefully in interval [-tol, tol])
`tol`	Border of interval (must be positive)

Details

NA's are removed for all computations.

Value

Value between [0, 1]

`qualGaussDev` *Compute probability of Gaussian (mu=m, sd=s) at a position 0, with reference to the max obtainable probability of that Gaussian at its center.*

Description

Measure for centeredness around 0. Highest score is 1, worst score is 0.

Usage

```
qualGaussDev(mu, sd)
```

Arguments

`mu`	Center of Gaussian
`sd`	SD of Gaussian

Value

quality, ranging from 0 (bad agreement) to 1 (perfect, i.e. centered at 0)

`qualHighest` *Score an empirical density distribution of values, where the best possible distribution is right-skewed.*

Description

The score is computed according to

Usage

```
qualHighest(x, N)
```

Arguments

`x`	Vector of numeric values (e.g. height of histogram bins)
`N`	Length of x (just a precaution currently)

Details

q = ((N-1) - sum_i(((N-i-1)*x_i)) / (N-1)

Scores range from 0 (worst), to 1 (best). E.g. c(0,0,0,16) would yield a score of 1. c(16,0,0,0,0) gives a score of 0.

Value

Quality score in the range of [0,1]

Examples

```
qualHighest(c(0,0,0,16), 4)   ## 1
qualHighest(c(16,0,0,0), 4)   ## 0
qualHighest(c(1,1,1,1), 4)    ## 0.5
qualHighest(c(0,16,0,0), 4)   ## 1/3
```

---

qualLinThresh *Quality metric with linear response to input, reaching the maximum score at the given threshold.*

Description

Ranges between 0 (worst score) and 1 (best score). Useful for performance measures where reaching a certain reference threshold 't' will be enough to reach 100%. The input range from [0, t] is scored from 0-100%.

Usage

```
qualLinThresh(x, t = 1)
```

Arguments

x	Numeric value(s) between [0, inf]
t	Threshold value, which indicates 100%

Value

Value between [0, 1]

qualMedianDist *Quality metric which measures the absolute distance from median.*

Description

Ranges between 0 (worst score) and 1 (best score). Input must be between [0,1]. Deviations from the median of the sample represent the score for each sample point.

Usage

```
qualMedianDist(x)
```

Arguments

x	A vector numeric values between [0,1]

Value

A vector of the same size as x, with quality values between [0, 1]

qualUniform *Compute deviation from uniform distribution*

Description

The score ranges between 0 (worst score) and 1 (best score). Input 'x' is a vector of counts (or probabilities) for equally spaced bins in a histogram. A uniform distribution (e.g. c(3,3,3) will get a score of 1. The worst possible case (e.g. c(4,0,0)), will get a score of 0, and a linear increasing function (e.g. c(1,2,3)) will get something in between (0.585 here)

Usage

```
qualUniform(x, weight = vector())
```

Arguments

x	Vector of numeric intensity/count values (e.g. ID's per RT bin); bins are assumed to have equal widths
weight	Vector of weights for values in 'x' (same length as 'x').

Details

In addition, bin values can be weighted (e.g. by their confidence). The total sum of weights is normalized to 1 internally.

The distance function used is the square root of the absolute difference between a uniform distribution and the input 'x' (summed for each element of 'x'). This distance is normalized to the worst possible input (e.g. one bin with 100

Value

Value between [0, 1]

Examples

```
stopifnot(qualUniform(c(3,3,3))==1)
stopifnot(qualUniform(c(4,0,0))==0)

## how 'uniform' is a vector where only a single index has weight?-- answer: very
stopifnot(qualUniform(c(4,0,0), c(1,0,0))==1)
stopifnot(qualUniform(c(4,0,0), c(0,1,0))==1)
stopifnot(qualUniform(c(0,4,0))==0)
stopifnot(abs(qualUniform(c(3,2,1))-0.58578) < 0.0001)
stopifnot(abs(qualUniform(c(1,2,3))-0.58578) < 0.0001)
stopifnot(qualUniform(c(1,2,3), c(0,1,0))==1)
stopifnot(abs(qualUniform(c(1,2,3))-0.58578) < 0.0001)
stopifnot(abs(qualUniform(c(1,2,3), c(0,1,1))- 0.590316) < 0.0001)
stopifnot(abs(qualUniform(c(2,3), c(1,1))-0.552786) < 0.0001)
stopifnot(abs(qualUniform(1:120)-0.38661) < 0.0001)
```

---

read.MQ *Convenience wrapper for MQDataReader when only a single MQ file should be read and file mapping need not be stored.*

Description

For params, see `MQDataReader::readMQ()`.

Usage

```
read.MQ(
  file,
  filter = "",
  type = "pg",
  col_subset = NA,
  add_fs_col = 10,
  LFQ_action = FALSE,
  ...
)
```

Arguments

file	see `MQDataReader::readMQ()`
filter	see `MQDataReader::readMQ()`
type	see `MQDataReader::readMQ()`
col_subset	see `MQDataReader::readMQ()`

add_fs_col	see MQDataReader::readMQ()
LFQ_action	see MQDataReader::readMQ()
...	see MQDataReader::readMQ()

Value

see MQDataReader::readMQ()

`renameFile` *Given a vector of (short/long) filenames, translate to the (long/short) version*

Description

Given a vector of (short/long) filenames, translate to the (long/short) version

Usage

```
renameFile(f_names, mapping)
```

Arguments

f_names	Vector of filenames
mapping	A data.frame with from,to columns

Value

A vector of translated file names as factor (ordered by mapping!)

`repEach` *Repeat each element x_i in X, n_i times.*

Description

Repeat each element x_i in X, n_i times.

Usage

```
repEach(x, n)
```

Arguments

x	Values to be repeated
n	Number of repeat for each x_i (same length as x)

Value

Vector with values from x, n times

Examples

```
repEach(1:3, 1:3)  ## 1, 2, 2, 3, 3, 3
```

RSD *Relative standard deviation (RSD)*

Description

Simply CV*100

Usage

```
RSD(x)
```

Arguments

x	Vector of numeric values

Value

RSD

RTalignmentTree *Return a tree plot with a possible alignment tree.*

Description

This allows the user to judge which Raw files have similar corrected RT's (i.e. where aligned successfully). If there are clear sub-clusters, it might be worth introducing artifical fractions into MaxQuant, to avoid ID-transfer between these clusters (use the MBR-Align and MBR-ID-Transfer metrics to support the decision).

Usage

```
RTalignmentTree(df_evd, col_fraction = c())
```

Arguments

df_evd	Evidence table containing calibrated retention times and sequence information.
col_fraction	Empty vector or 1-values vector giving the name of the fraction column (if existing)

Details

If the input contains fractions, leaf nodes will be colored accordingly. Distinct sub-clusters should have their own color. If not, MaxQuant's fraction settings should be optimized. Note that introducing fractions in MaxQuant will naturally lead to a clustering here (it's somewhat circular).

Value

ggplot object containing the correlation tree

---

scale01linear *Scales a vector of values linearly to [0, 1] If all input values are equal, returned values are all 0*

Description

Scales a vector of values linearly to [0, 1] If all input values are equal, returned values are all 0

Usage

```
scale01linear(X)
```

Arguments

X	Vector of values

Value

Scaled vector

---

scale_x_discrete_reverse *Inverse the order of items on the x-axis (for discrete scales)*

Description

Inverse the order of items on the x-axis (for discrete scales)

Usage

```
scale_x_discrete_reverse(values, ...)
```

Arguments

values	The vector of values as given to the x aestetic
...	Other arguments forwarded to 'scale_y_discrete()'

Value

ggplot object, concatenatable with '+'

---

scale_y_discrete_reverse *Inverse the order of items on the y-axis (for discrete scales)*

Description

Inverse the order of items on the y-axis (for discrete scales)

Usage

```
scale_y_discrete_reverse(values, ...)
```

Arguments

values	The vector of values as given to the y aestetic
...	Other arguments forwarded to 'scale_y_discrete()'

Value

ggplot object, concatenatable with '+'

ScoreInAlignWindow *Compute the fraction of features per Raw file which have an acceptable RT difference after alignment*

Description

Using the result from 'alignmentCheck()', score the features of every Raw file and see if they have been properly aligned. Returned value is between 0 (bad) and 1 (all aligned).

Usage

```
ScoreInAlignWindow(data, allowed.deltaRT = 1)
```

Arguments

data	A data.frame with columns 'rtdiff' and 'raw.file'
allowed.deltaRT	The allowed matching difference (1 minute by default)

Value

A data.frame with one row for each raw.file and columns 'raw.file' and 'withinRT' (0-1)

shortenStrings *Shorten a string to a maximum length and indicate shorting by appending '..'*

Description

Some axis labels are sometimes just too long and printing them will either squeeze the actual plot (ggplot) or make the labels disappear beyond the margins (graphics::plot) One ad-hoc way of avoiding this is to shorten the names, hoping they are still meaningful to the viewer.

Usage

```
shortenStrings(x, max_len = 20, verbose = TRUE, allow_duplicates = FALSE)
```

Arguments

x	Vector of input strings
max_len	Maximum length allowed
verbose	Print which strings were shortened
allow_duplicates	If shortened strings are not discernible any longer, consider the short version valid (not the default), otherwise (default) return the full string (–> no-op)

Details

This function should be applied AFTER you tried more gentle methods, such as delLCP or simplifyNames.

Value

A vector of shortened strings

See Also

delLCP, simplifyNames

Examples

```
r = shortenStrings(c("gamg_101", "gamg_101230100451", "jurkat_06_100731121305", "jurkat_06_1"))
all(r == c("gamg_101", "gamg_101230100..", "jurkat_06_1007..", "jurkat_06_1"))
```

simplifyNames *Removes common substrings (infixes) in a set of strings.*

Description

Usually handy for plots, where condition names should be as concise as possible. E.g. you do not want names like 'TK20130501_H2M1_010_IMU008_CISPLA_E3_R1.raw' and 'TK20130501_H2M1_026_IMU008_CISP but rather 'TK.._010_I.._E3_R1.raw' and 'TK.._026_I.._E7_R2.raw'

If multiple such substrings exist, the algorithm will remove the longest first and iterate a number of times (two by default) to find the second/third etc longest common substring. Each substring must fulfill a minimum length requirement - if its shorter, its not considered worth removing and the iteration is aborted.

Usage

```
simplifyNames(
  strings,
  infix_iterations = 2,
  min_LCS_length = 7,
  min_out_length = 7
)
```

Arguments

strings	A vector of strings which are to be shortened
infix_iterations	Number of successive rounds of substring removal
min_LCS_length	Minimum length of the longest common substring (default:7, minimum: 6)
min_out_length	Minimum length of shortest element of output (no shortening will be done which causes output to be shorter than this threshold)

Value

A list of shortened strings, with the same length as the input

Examples

```
#library(PTXQC)
simplifyNames(c('TK20130501_H2M1_010_IMU008_CISPLA_E3_R1.raw',
                'TK20130501_H2M1_026_IMU008_CISPLA_E7_R2.raw'), infix_iterations = 2)
# --> "TK.._010_I.._E3_R1.raw","TK.._026_I.._E7_R2.raw"

try(simplifyNames(c("bla", "foo"), min_LCS_length=5))
# --> error, since min_LCS_length must be >=6
```

supCount — *Compute shortest prefix length which makes all strings in a vector uniquely identifyable.*

Description

If there is no unique prefix (e.g. if a string is contained twice), then the length of the longest string is returned, i.e. if the return value is used in a call to substr, nothing happens e.g. substr(x, 1, supCount(x)) == x

Usage

```
supCount(x, prefix_l = 1)
```

Arguments

x	Vector of strings
prefix_l	Starting prefix length, which is incremented in steps of 1 until all prefixes are unique (or maximum string length is reached)

Value

Integer with minimal prefix length required

Examples

```
supCount(c("abcde...", "abcd...", "abc..."))  ## 5

x = c("doubled", "doubled", "aLongDummyString")
all( substr(x, 1, supCount(x)) == x )
## TRUE (no unique prefix due to duplicated entries)
```

theme_blank — *A blank theme (similar to the deprecated theme_blank())*

Description

A blank theme (similar to the deprecated theme_blank())

Usage

```
theme_blank()
```

Value

A ggplot2 object, representing an empty theme

`thinOut` *Thin out a data.frame by removing rows with similar numerical values in a certain column.*

Description

All values in the numerical column 'filterColname' are assigned to bins of width 'binsize'. Only one value per bin is retained. All other rows are removed and the reduced data frame will all its columns is returned.

Usage

```
thinOut(data, filterColname, binsize)
```

Arguments

`data`	The data.frame to be filtered
`filterColname`	Name of the filter column as string
`binsize`	Width of a bin

Value

Data.frame with reduced rows, but identical input columns

`thinOutBatch` *Apply 'thinOut' on all subsets of a data.frame, split by a batch column*

Description

The binsize is computed from the global data range of the filter column by dividing the range into binCount bins.

Usage

```
thinOutBatch(data, filterColname, batchColname, binCount = 1000)
```

Arguments

`data`	The data.frame to be split and filtered(thinned)
`filterColname`	Name of the filter column as string
`batchColname`	Name of the split column as string
`binCount`	Number of bins in the 'filterColname' dimension.

Value

Data.frame with reduced rows, but identical input columns

wait_for_writable *Check if a file is writable and blocks an interactive session, waiting for user input.*

Description

This functions gives the user a chance to make the output file writeable before a write attempt is actually made by R to avoid having run the whole program again upon write failure.

Usage

```
wait_for_writable(
  filename,
  prompt_text = paste0("The file '", filename,
    "' is not writable. Please close all applications using this file. Press '",
    abort_answer, "' to abort!"),
  abort_answer = "n"
)
```

Arguments

filename	The file to test for writable
prompt_text	If not writable, show this prompt text to the user
abort_answer	If the user enters this string into the prompt, this function will stop()

Details

Note: The file will not be overwritten or changed by this function.

Value

TRUE if writable, FALSE if aborted by user or (not-writeable and non-interactive)

YAMLClass-class *Query a YAML object for a certain parameter.*

Description

If the object has the param, then return it. If the param is unknown, create it with the given default value and return the default.

Fields

yamlObj A Yaml object as created by yaml.load

Methods

`getYAML(param_name, default, min = NA, max = NA)` Query this YAML object for a certain parameter and return its value. If it does not exist it is created with a default value. An optional min/max range can be specified and will be enforced if the value is known (default will be used upon violation).

`setYAML(param_name, value)` Set a YAML parameter to a certain value. Overwrites the old value or creates a new entry if hithero unknown.

`writeYAML(filename)` Write YAML config (including some documentation) to a YAML file. Returns TRUE on success (always), unless writing the file generates an error.

Examples

```
yc = YAMLClass$new(list())
val = yc$getYAML("cat$subCat", "someDefault")
val  ## someDefault
val = yc$setYAML("cat$subCat", "someValue")
val  ## someValue
yc$getYAML("cat$subCat", "someDefault") ## still 'someValue' (since its set already)
```

%+% *A string concatenation function, more readable than 'paste()'.*

Description

A string concatenation function, more readable than 'paste()'.

Usage

```
a %+% b
```

Arguments

a	Char vector
b	Char vector

Value

Concatenated string (no separator)

Index