Package 'metabCombiner'

October 12, 2021

Version 1.2.2

Date 2021-08-23

Title Method for Combining LC-MS Metabolomics Feature Measurements

License GPL-3

Description This package aligns LC-HRMS metabolomics datasets acquired from biologically similar specimens analyzed under similar, but not necessarily identical, conditions. Peak-picked and simply aligned metabolomics feature tables (consisting of m/z, rt, and per-sample abundance measurements, plus optional identifiers & adduct annotations) are accepted as input. The package outputs a combined table of feature pair alignments, organized into groups of similar m/z, and ranked by a similarity score. Input tables are assumed to be acquired using similar (but not necessarily identical) analytical methods.

Depends R (>= 4.0), dplyr (>= 1.0)

Imports methods, mgcv, caret, S4Vectors, stats, utils, rlang, graphics, matrixStats, tidyr

Suggests knitr, rmarkdown, testthat, BiocStyle

BugReports https://www.github.com/hhabra/metabCombiner/issues

NeedsCompilation yes

RoxygenNote 7.1.1

Encoding UTF-8

Collate 'adjustData.R' 'batchCombine.R' 'calcScores.R' 'check_pars.R' 'classes.R' 'combinerCheck.R' 'compare_strings.R' 'data.R' 'detectFields.R' 'evaluateParams.R' 'fit_model.R' 'form.R' 'generics.R' 'labelRows.R' 'metabCombine.R' 'metabCombiner.R' 'metabCombiner_package_doc.R' 'metabData.R' 'methods-featdata.R' 'methods-metabCombiner.R' 'methods-metabData.R' 'mzGroup.R' 'params.R' 'plot_fit.R' 'selectAnchors.R' 'write2file.R' 'zzz.R'

VignetteBuilder knitr

biocViews Software, MassSpectrometry, Metabolomics

LazyData false

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

git_branch RELEASE_3_13

git_last_commit ef53e69

git_last_commit_date 2021-08-23

Date/Publication 2021-10-12

Author Hani Habra [aut, cre],
Alla Karnovsky [ths]

Maintainer Hani Habra <hhabra1@gmail.com>

R topics documented:

`adductdata` *Retrieve Adduct Annotations*

Description

This retrieves user-assigned adduct annotations from one or all constituent datasets of a `metabCombiner` object

Usage

```
adductdata(object, data = NULL)

## S4 method for signature 'metabCombiner'
adductdata(object, data = NULL)
```

Arguments

`object`	`metabCombiner` object
`data`	dataset identifier to extract information from; if NULL, extracts information from all datasets

Value

data frame of adduct annotations

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")

p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##retrieve all adduct data
adducts <- adductdata(p.comb, data = NULL)

##retrieve adduct data from p30
adducts <- adductdata(p.comb, data = "p30")
```

`adjustData` *Process and Filter Metabolomics Feature Lists*

Description

`adjustData` contains a set of pre-analysis steps for processing LC-MS metabolomics feature tables individually

Usage

```
adjustData(Data, misspc, measure, rtmin, rtmax, zero, duplicate)
```

Arguments

`Data`	a metabData object.
`misspc`	Numeric. Threshold missingness percentage for analysis.
`measure`	Character. Choice of central abundance measure; either "median" or "mean".
`rtmin`	Numeric. Minimum retention time for analysis.
`rtmax`	Numeric. Maximum retention time for analysis.
`zero`	Logical. Whether to consider zero values as missing.
`duplicate`	Ordered numeric pair (m/z, rt) tolerance parameters for duplicate feature search.

Details

The pre-analysis adjustment steps include: 1) Restriction to a feature retention time range `rtmin` $\leq$ rt $\leq$ `rtmax` 2) Removal of features with a percent missingness exceeding `misspc` 3) Removal of duplicate metabolomics features.

After processing, abundance quantile (Q) values are calculated between 0 & 1 for the remaining features, as ranked by the `measure` argument, unless provided by the user.

Value

Updated metabData object. The data field is processed by the listed steps and stats list updated to contain feature statistics.

See Also

metabData: the constructor for metabData objects, filterRT: function for filtering by retention times, findDuplicates: function for finding duplicates

---

batchCombine *Stepwise Multi-batch LC-MS Alignment*

Description

This is a method for aligning multiple batches of a single metabolomics experiment in a stepwise manner using the metabCombiner workflow. The input is a list of metabData objects corresponding to the batch data frames arranged in sequential order (i.e. batch 1,2,...,N), and parameter lists for each step; the output is an aligned feature table and a metabCombiner object composed from the input batches.

Usage

```
batchCombine(
  batches,
  binGap = 0.005,
  fitMethod = "gam",
  means = list(mz = TRUE, rt = TRUE, Q = TRUE),
  anchorParam = selectAnchorsParam(),
  fitParam = fitgamParam(),
  scoreParam = calcScoresParam(B = 30),
  reduceParam = reduceTableParam()
)
```

Arguments

batches	list of metabData objects corresponding to each LC-MS batch
binGap	numeric parameter used for grouping features by m/z. See ?mzGroup for more details.
fitMethod	RT spline-fitting method, either "gam" or "loess"
means	logical. Option to take average m/z, rt, and/or Q from metabCombiner. May be a 3-length vector, single value (TRUE/FALSE), or a list with names "mz", "rt", "Q" as names.
anchorParam	list of parameter values for selectAnchors() function
fitParam	list of parameter values for fit_gam() or fit_loess()
scoreParam	list of parameter values for calcScores()
reduceParam	list of parameter values for reduceTable()

Details

Retention time drifting is commonly observed in large-scale LC-MS experiments in which samples are analyzed in multiple batches. Conventional LC-MS pre-processing approaches may effectively align features detected in samples from within a single batch, but fail in many cases to account for inter-batch drifting, leading to misaligned features.

`batchCombine` assumes that each batch has been previously processed separately using conventional LC-MS preprocessing approaches (e.g. XCMS), and can be represented as a data frame. Each batch data feature table must be filtered and formatted as a `metabData` object and the batches must be arranged as a list in sequential order of acquisition.

`batchCombine` applies the `metabCombine` wrapper function to successive pairs of metabolomics batches in a stepwise manner. Each iteration consists of the key steps in the package workflow (feature m/z grouping, anchor selection, retention time spline fitting, pairwise scoring, & table reduction). The first two batches are aligned together, then the combined results are aligned with the third batch, and so forth. Parameters for each sub-method are arranged in list format, with their respective defaults (e.g. fitgamParam() lists the default values for the fit_gam function).

Following each iteration, m/z, rt, and Q values from the combined dataset may be averaged to use for comparison with the next batch's feature quantitative descriptors, if the `means` argument is set to TRUE; if set to FALSE, feature information is drawn from the latter of the previously combined batches, identical to the manner in which id & adduct descriptors are drawn.

Value

`object`	metabCombiner object of the final alignment; x is set to the penultimate batch and y is set to the final batch
`table`	combined feature table consisting of feature descriptor values followed by per-sample abundances and extra columns

Note

`batchCombine` is designed for aligning multi-batch datasets, i.e. where each batch is acquired in a roughly identical manner. It is not for disparately acquired LC-MS datasets (e.g. from different instruments, chromatographic systems, laboratories, etc.).

See Also

`metabCombine`

Examples

```
#identically formatted batches in list form
data(metabBatches)

#obtain list of metabData objects
batchdata <- lapply(metabBatches, metabData, samples = "POOL",
                    extra = "SAMP", zero = TRUE)

#optional: give each batch dataset a name
names(batchdata) <- paste("B", seq_along(batchdata), sep = "")
```

```
#customize main workflow parameter lists
saparam <- selectAnchorsParam(tolmz = 0.002, tolQ = 0.2, tolrtq = 0.1)
fgparam <- fitgamParam(k = 20, iterFilter = 1)
csparam <- calcScoresParam(A = 70, B = 35, C = 0.3)
rtparam <- reduceTableParam(minScore = 0.5, maxRTerr = 0.33)

#run batchCombine program
combinedRes <- batchCombine(batches = batchdata, binGap = 0.0075,
               means = list('mz' = TRUE, 'rt' = FALSE, 'Q' = FALSE),
               anchorParam = saparam, fitParam = fgparam,
               scoreParam = csparam, reduceParam = rtparam)

#aligned table results & metabCombiner object results
cTable <- combinedRes$table
object <- combinedRes$object

#if names were set earlier, the names should be returned by this
datasets(object)
```

`calcScores` *Compute Feature Similarity Scores*

Description

Calculates a pairwise similarity (between 0 & 1) between all grouped features in `metabCombiner` object. The similarity score calculation is described in `scorePairs`.

Usage

```
calcScores(
  object,
  A = 75,
  B = 10,
  C = 0.25,
  fit = c("gam", "loess"),
  groups = NULL,
  useAdduct = FALSE,
  adduct = 1.25,
  usePPM = FALSE,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

`object`	metabCombiner object.
`A`	Numeric weight for penalizing m/z differences.

B	Numeric weight for penalizing differences between fitted & observed retention times
C	Numeric weight for differences in Q (abundance quantiles).
fit	Character. Choice of fitted rt model, "gam" or "loess."
groups	integer. Vector of feature groups to score. If set to NULL (default), will compute scores for all feature groups.
useAdduct	logical. Option to penalize mismatches in (non-empty, non-bracketed) adduct column annotations.
adduct	numeric. If useAdduct is TRUE, divides score of mismatched, non-empty and non-bracked adduct column labels by this value.
usePPM	logical. Option to use relative (as opposed to absolute) m/z differences in score computations.
brackets_ignore	If useAdduct = TRUE, bracketed adduct character strings of these types will be ignored according to this argument

Details

This function updates the rtProj, score, rankX, and rankY columns in the combinedTable report. First, using the RT mapping model computed in the previous step(s), rtx values are projected onto rty. Then similarity scores are calculated based on m/z, rt (fitted vs observed), and Q differences, with multiplicative weight penalties A, B, and C.

If the datasets contain representative set of shared identities (idx = idy), evaluateParams provides some guidance on appropriate A, B, and C values to use. In testing, the best values for A should lie between 50 and 120, according to mass accuracy; B should lie between 5 and 15 depending on fitting accuracy (higher if datasets processed under roughly identical conditions) ; C should vary between 0 and 1, depending on sample similarity. See examples below.

If using ppm (usePPM = TRUE), do not use the above guidelines for A values. The suggested range is between 0.01 and 0.05, though this hasn't been thoroughly tested yet. Also, if using adduct information (useAdduct = TRUE), the score is divided by the numeric adduct argument if non-empty and non-bracketed adduct values do not match. Be sure that adduct annotations are accurate before using this functionality.

Value

metabCombiner object with updated combinedTable. rtProj column will contain fitted retention times determined from previously computed model; score will contain computed pairwise similarity scores of feature pairs; rankX & rankY are the integer ranks of scores for x & y features in descending order.

See Also

evaluateParams, scorePairs

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)

p.comb <- selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)
p.comb <- fit_gam(p.comb, k = 20, iterFilter = 1, family = "gaussian")

#example: moderate m/z deviation, accurate rt fit, high sample similarity
p.comb <- calcScores(p.comb, A = 90, B = 14, C = 0.8, useAdduct = FALSE,
         groups = NULL, fit = "gam", usePPM = FALSE)
cTable = combinedTable(p.comb)  #to view results


#example 2: high m/z deviation, moderate rt fit, low sample similarity
p.comb <- calcScores(p.comb, A = 50, B = 8, C = 0.2)

#example 3: low m/z deviation, poor rt fit, moderate sample similarity
p.comb <- calcScores(p.comb, A = 120, B = 5, C = 0.5)

#example 4: using ppm for mass deviation; note different A value
p.comb <- calcScores(p.comb, A = 0.05, B = 14, C = 0.5, usePPM = TRUE)

#example 5: limiting to specific m/z groups 1-1000
p.comb <- calcScores(p.comb, A = 90, B = 14, C = 0.5, groups = seq(1,1000))

#example 6: using adduct information
p.comb <- calcScores(p.comb, A = 90, B = 14, C = 0.5, useAdduct = TRUE,
                     adduct = 1.25)
```

`calcScoresParam` *List calcScores Defaults*

Description

List of default parameters for score calculation step of main package workflow. See help(calcScores) or ?calcScores for details.

Usage

```
calcScoresParam(
  A = 75,
  B = 10,
  C = 0.25,
  fit = "gam",
```

```
  groups = NULL,
  usePPM = FALSE,
  useAdduct = FALSE,
  adduct = 1.25,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

A	m/z difference specific weight; default: 75
B	RT prediction error specific weight; default: 10
C	Q difference specific weight; default: 0.25
fit	choice of fitted model ("gam" or "loess"); default: "gam"
groups	choice of m/z groups to score
usePPM	choice to use PPM for m/z differences; default: FALSE
useAdduct	choice to use adduct strings in scoring; default: FALSE
adduct	value divisor for mismatched adduct strings; default: 1.25
brackets_ignore	bracket types for ignoring string comparisons

Value

list of calcScores parameters

See Also

calcScores, metabCombine

Examples

```
cs_param <- calcScoresParam(A = 60, B = 15, C = 0.3)

cs_param <- calcScoresParam(A = 0.1, B = 20, C = 0.2, usePPM = TRUE)
```

---

combinedTable *Obtain Feature Alignment Report*

Description

Obtain constructed table reporting every possible metabolomics feature pair alignment.

Usage

```
combinedTable(object)

## S4 method for signature 'metabCombiner'
combinedTable(object)
```

Arguments

object	metabCombiner object.

Value

Feature Pair Alignment report data frame. The columns of the report are as follows:

idx	Identities of features from dataset X
idy	Identities of features from dataset Y
mzx	m/z values of features from dataset X
mzy	m/z values of features from dataset Y
rtx	retention time values of features from dataset X
rty	retention time values of features from dataset Y
rtProj	model-projected (X->Y) retention times values
Qx	abundance quantile values of features from dataset X
Qy	abundance quantile values of features from dataset Y
group	m/z feature group of feature pairing
score	computed similarity scores of feature pairing
rankX	ranking of pairing score for X dataset features
rankY	ranking of pairing score for Y dataset features
adductX	adduct label of features from dataset X
adductY	adduct label of features from dataset Y
...	Sample and extra columns from both datasets X & Y

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")

p.comb <- metabCombiner(p30, p20)
p.comb.table <- combinedTable(p.comb)
```

combinerCheck *Obtain Errors for metabCombiner Object Checks*

Description

This function stores and returns a customized error message when checking the validity of certain objects.

Usage

```
combinerCheck(errNo, type, error = "stop")
```

Arguments

errNo	integer error code.
type	character object type (either "combinedTable", "metabCombiner" or "metabData")
error	character. If "stop", gives an error message; if "warning", provides a warning message; if "silent", returns silently

Details

In certain functions, an object must be checked for correctness. A `metabData` must have a properly formatted dataset with the correct column names & types.A `metabCombiner` must have properly formatted `combinedTable`, with expected names and columns. If one of these conditions is not met, a non-zero numeric code is returned and this function is used to print a specific error message corresponding to the appropriate object and error code.

Value

A customized error message for specific object check.

crossValFit *Cross Validation for Model Fits*

Description

Helper function for `fit_gam()` & `fit_loess()`. Determines optimal value of `k` basis functions for Generalized Additive Model fits or `span` for loess fits from among user-defined choices, using a 10-fold cross validation minimizing mean squared error.

Usage

```
crossValFit(
  rts,
  fit,
  vals,
  bs,
  family,
  m,
  method,
  optimizer,
  control,
  message,
  ...
)
```

Arguments

rts	data.frame of ordered pair retention times
fit	Either "gam" for GAM fits, or "loess" for loess fits
vals	numeric vector: k values for GAM fits, spans for loess fits. Best value chosen by 10-fold cross validation.
bs	character. Choice of spline method, either "bs" or "ps"
family	character. Choice of mgcv family; see: ?mgcv::family.mgcv
m	integer. Basis and penalty order for GAM; see ?mgcv::s
method	character. Smoothing parameter estimation method; see: ?mgcv::gam
optimizer	character. Method to optimize smoothing parameter; see: ?mgcv::gam
control	control parameters for loess fits; see: ?loess.control
message	Option to print message indicating function progress
...	Other arguments passed to `mgcv::gam`.

Value

Optimal parameter value as determined by 10-fold cross validation

---

datasets *Obtain Dataset IDs*

Description

Each dataset in a `metabCombiner` object is represented by a character identifier. The datasets slot contains all these ids in a single vector, which can be obtained in sequential order with this accessor method

Usage

```
datasets(object, list = FALSE)

## S4 method for signature 'metabCombiner'
datasets(object, list = FALSE)
```

Arguments

object	metabCombiner object
list	logical, option to return in list format (TRUE) vs character vector format (FALSE)

Value

character vector of dataset identifiers

Examples

```
#' @examples
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")

p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##datasets extraction: expect "p30", "p20"
sets <- datasets(p.comb, list = FALSE)
```

---

detectFields *Detect metabData Input Columns*

Description

This function ensures that metabolomics datasets used as inputs for the program possess all of the required fields, plus any optional columns that may appear in the final report table.

Usage

```
detectFields(Data, table, mz, rt, id, adduct, samples, extra, Q)
```

Arguments

Data	a metabData object.
table	data frame containing metabolomics features or path to metabolomics data file.
mz	Character name(s) / regular expressions associated with data column containing m/z values. The first column whose name contains this expression will be selected for analysis.
rt	Character name(s) / regular expression associated with data column containing retention time values. The first column whose name contains this expression will be selected for analysis.
id	Character name(s) or regular expression associated with data column containing metabolomics feature identifiers. The first column whose name contains this expression will be selected for analysis.
adduct	Character name(s) or regular expression associated with data column containing adduct, formula, or additional annotations. The first column whose name contains this expression will be selected for analysis.
samples	Character names of columns containing sample values. All numeric columns containing these keywords are selected for analysis. If no keywords given, searches for longest stretch of numeric columns remaining.
extra	Character names of columns containing additional feature information, e.g. non-analyzed sample values. All columns containing these keywords are selected for analysis.
Q	Character name(s) or regular expression associated with numeric feature abundance quantiles.

Value

an initialized and formatted metabData object.

---

evaluateParams *Evaluate Similarity Score Parameters*

Description

This function provides a method for guiding selection of suitable values for A, B, & C weight arguments in the calcScores method, based on the similarity scores of shared identified compounds. Datasets must have at least one identity in common (i.e. idx = idy, case-insensitive), and preferably more than 10.

Usage

```
evaluateParams(
  object,
  A = seq(60, 150, by = 10),
  B = seq(6, 15),
```

```
  C = seq(0.1, 0.5, by = 0.1),
  fit = c("gam", "loess"),
  usePPM = FALSE,
  minScore = 0.5,
  penalty = 5,
  groups = NULL,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

object	metabCombiner object
A	Numeric weights for penalizing m/z differences.
B	Numeric weights for penalizing differences between fitted & observed retention times
C	Numeric weight for differences in Q (abundance quantiles).
fit	Character. Choice of fitted rt model, "gam" or "loess."
usePPM	logical. Option to use relative parts per million (ppm) as opposed to absolute) m/z differences in score computations.
minScore	numeric minimum score to count towards objective function calculation for known matching features (idx = idy) and mismatches.
penalty	numeric. Subtractive mismatch penalty.
groups	integer. Vector of feature groups to score. If set to NULL (default), will compute scores for all feature groups.
brackets_ignore	bracketed identity and adduct character strings of these types will be ignored according to this argument

Details

This uses an objective function, based on the accurate and inaccurate alignments of shared pre-identified compounds. For more details, see: objective.

Value

A data frame with the following columns:

A	m/z weight values
B	rt weight values
C	Q weight values
score	objective function evaluation of (A,B,C) weights

Note

In contrast to calcScores function, A, B, & C take numeric vectors as input, as opposed to constants. The total number of rows in the output will be equal to the products of the lengths of these input vectors

See Also

calcScores, objective

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)

p.comb = selectAnchors(p.comb, windx = 0.03, windy = 0.02)
p.comb = fit_gam(p.comb, k = 20, iterFilter = 2)

#example 1
scores = evaluateParams(p.comb, A = seq(60,100,10), B = seq(10,15), C = 0.5,
    minScore = 0.7, penalty = 10)

##example 2: using PPM mass deviation (note change to A argument)
scores = evaluateParams(p.comb, usePPM = TRUE, A = seq(0.01,0.05,0.01))

##example 3: limiting to groups 1-2000
scores = evaluateParams(p.comb, minScore = 0.5, groups = 1:2000)
```

---

featdata *Obtain Feature Metadata*

Description

metabCombiner objects organize metabolomics feature information in the "featdata" slot. This method retrieves all metadata or that of one dataset. The rows should identically correspond to the same rows from the combinedTable data frame.

Usage

```
featdata(object, data = NULL)

## S4 method for signature 'metabCombiner'
featdata(object, data = NULL)
```

Arguments

object	a metabCombiner object
data	character dataset identifier

Value

data frame of feature metadata from one or all datasets

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")

p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

#full metadata extraction
fdata <- featdata(p.comb, data = NULL)

#single dataset feature information extraction
fdata <- featdata(p.comb, data = "p20")
```

`filterAnchors` *Filter Outlier Ordered Pairs*

Description

Helper function for `fit_gam` & `fit_loess`. It filters the set of ordered pairs using the residuals calculated from multiple GAM / loess fits.

Usage

```
filterAnchors(
  rts,
  fit,
  vals,
  outlier,
  coef,
  iterFilter,
  prop,
  bs,
  m,
  family,
  method,
  optimizer,
  control,
  message,
  ...
)
```

Arguments

rts	Data frame of ordered retention time pairs.
fit	Either "gam" for GAM fits, or "loess" for loess fits
vals	numeric values: k values for GAM fits, spans for loess fits
outlier	Thresholding method for outlier dection. If "MAD", the threshold is the mean absolute deviation (MAD) times coef; if "boxplot", the threshold is coef times IQR plus 3rd quartile of a model's absolute residual values.
coef	numeric (> 1) multiplier for determining thresholds for outliers (see outlier argument)
iterFilter	integer number of outlier filtering iterations
prop	numeric. A point is excluded if deemed a residual in more than this proportion of fits. Must be between 0 & 1.
bs	character. Choice of spline method from mgcv; either "bs" or "ps"
m	integer. Basis and penalty order for GAM; see ?mgcv::s
family	character. Choice of mgcv family; see: ?mgcv::family.mgcv
method	character. Smoothing parameter estimation method; see: ?mgcv::gam
optimizer	character. Method to optimize smoothing parameter; see: ?mgcv::gam
control	control parameters for loess fits; see: ?loess.control
message	Option to print message indicating function progress
...	other arguments passed to mgcv::gam.

Value

anchor rts data frame with updated weights.

---

filterRT *Filter Features by Retention Time*

Description

Restricts input metabolomics feature table in metabData object to a range of retention times defined by rtmin & rtmax.

Usage

```
filterRT(data, rtmin, rtmax)
```

Arguments

data	formatted metabolomics data frame.
rtmin	lower range of retention times for analysis. If "min", defaults to minimum observed retention time. .
rtmax	upper range of retention times for analysis. If "max", defaults to maximum observed retention time.

Details

Retention time restriction is often recommended to aid the analysis of comparable metabolomics datasets. The beginning and end of a chromatogram typically contain features that do not correspond with true biological compounds derived from the sample. `rtmin` and `rtmax` should be set slightly before and slightly after the first and last commonly observed metabolites, respectively.

Value

A data frame of metabolomics features, limited to time window `rtmin` $\leq$ rt $\leq$ `rtmax`)

`findDuplicates` *Find and Remove Duplicate Features*

Description

Pairs of features with nearly identical m/z and retention time values are removed in this step.

Usage

```
findDuplicates(data, missing, counts, duplicate)
```

Arguments

`data`	Constructed metabolomics data frame.
`missing`	Numeric vector. Percent missingness for each feature.
`counts`	Numeric vector. Central measure for each feature.
`duplicate`	Ordered numeric pair (m/z, rt) tolerance parameters for duplicate feature search.

Details

Pairs of features are deemed duplicates if pairwise differences in m/z & rt fall within tolerances defined by the `duplicate` argument. If a pair of duplicate features is found, one member is removed. The determination of which feature to remove is first by percent missingness, followed by central abundance measure (median or mean). If the features have equal missingness and abundance, then row order determines the feature to be removed.

Value

integer indices of removable duplicate features

`fitgamParam` *List fit_gam Defaults*

Description

List of default parameters for GAM fitting step of main package workflow, which can be used as input for the wrapper functions. See help(fit_gam) or ?fit_gam for more details.

Usage

```
fitgamParam(
  useID = FALSE,
  k = seq(10, 20, 2),
  iterFilter = 2,
  outlier = "MAD",
  coef = 2,
  prop = 0.5,
  weights = 1,
  bs = "bs",
  family = "scat",
  m = c(3, 2),
  method = "REML",
  optimizer = "newton",
  message = TRUE
)
```

Arguments

`useID`	choice of preserving identity-based anchors; default: FALSE
`k`	values for GAM basis dimension k
`iterFilter`	number of outlier filtering iterations; default: 2
`outlier`	outlier filtering method (either "MAD" (mean absolute deviation) or "boxplot"); default: "MAD"
`coef`	outlier filtering coefficient; default: 2
`prop`	minimum proportion of fits in which a point can be a flagged outlier; default: 0.5
`weights`	optional supplied weights to individual points; default: 1
`bs`	choice of spline type ("bs" or "ps"); default: "bs"
`family`	choice of family ("scat" or "gaussian"); default: "scat"
`m`	basis and penalty order; default: c(3,2)
`method`	smoothing parameter estimation method; default: "REML"
`optimizer`	numerical optimization for GAM; default: "newton"
`message`	option to print progress message; default: TRUE

Value

list of fit_gam parameters

See Also

fit_gam, metabCombine

Examples

```
fitParam <- fitgamParam(k = c(12,14,18,20), iterFilter = 1, bs = "ps",
                        family = "gaussian", method = "GCV.Cp")
```

---

fitloessParam *List fitLoess Defaults*

Description

List of default parameters for loess fitting step of main package workflow, See help(fit_loess) or ?fit_loess for more details.

Usage

```
fitloessParam(
  useID = FALSE,
  spans = seq(0.2, 0.3, by = 0.02),
  outlier = "MAD",
  coef = 2,
  iterFilter = 2,
  prop = 0.5,
  weights = 1,
  message = TRUE,
  control = loess.control(surface = "direct", iterations = 10)
)
```

Arguments

useID	choice of preserving identity-based anchors; default: FALSE
spans	values for span parameter which controls degree of smoothing
outlier	outlier filtering method (either "MAD" or "boxplot"); default: "MAD"
coef	outlier filtering coefficient; default: 2
iterFilter	number of outlier filtering iterations; default: 2
prop	minimum proportion of fits where a point can be a flagged outlier; default: 0.5
weights	optional supplied weights to individual points; default: 1
message	option to print progress message; default: TRUE
control	loess-specific control parameters; see: ?loess.control

Value

list of fit_loess parameters:

See Also

fit_loess, metabCombine

Examples

```
fitParam <- fitloessParam(spans = c(0.2,0.25,0.3), outlier = "boxplot",
                          iterFilter = 3, coef = 1.5, message = FALSE,
                          control = loess.control(iterations = 4))
```

---

fit_gam *Fit RT Projection Model With GAMs*

Description

Fits a (penalized) basis splines curve through a set of ordered pair retention times, modeling one set of retention times (rty) as a function on the other set (rtx). Outlier filtering iterations are performed first, then with the remaining points, the best value of parameter k is selected through 10-fold cross validation.

Usage

```
fit_gam(
  object,
  useID = FALSE,
  k = seq(10, 20, 2),
  iterFilter = 2,
  outlier = c("MAD", "boxplot"),
  coef = 2,
  prop = 0.5,
  weights = 1,
  bs = c("bs", "ps"),
  m = c(3, 2),
  family = c("scat", "gaussian"),
  method = "REML",
  optimizer = "newton",
  message = TRUE,
  ...
)
```

Arguments

object	a metabCombiner object.
useID	logical. If set to TRUE, matched ID anchors detected from previous step will never be flagged as outliers.
k	integer k values controlling the dimension of the basis of the GAM fit (see: ?mgcv::s). Best value chosen by 10-fold cross validation.
iterFilter	integer number of outlier filtering iterations to perform
outlier	Thresholding method for outlier dection. If "MAD", the threshold is the mean absolute deviation (MAD) times coef; if "boxplot", the threshold is coef times IQR plus 3rd quartile of a model's absolute residual values.
coef	numeric (> 1) multiplier for determining thresholds for outliers (see outlier argument)
prop	numeric. A point is excluded if deemed a residual in more than this proportion of fits. Must be between 0 & 1.
weights	Optional user supplied weights for each ordered pair. Must be of length equal to number of anchors (n) or a divisor of (n + 2).
bs	character. Choice of spline method from mgcv, either "bs" (basis splines) or "ps" (penalized basis splines)
m	integer. Basis and penalty order for GAM; see ?mgcv::s
family	character. Choice of mgcv family; see: ?mgcv::family.mgcv
method	character smoothing parameter estimation method; see: ?mgcv::gam
optimizer	character. Method to optimize smoothing parameter; see: ?mgcv::gam
message	Option to print message indicating function progress
...	Other arguments passed to mgcv::gam.

Details

A set of ordered pair retention times must be previously computed using selectAnchors(). The minimum and maximum retention times from both input datasets are included in the set as ordered pairs (min_rtx, min_rty) & (max_rtx, max_rty). The weights argument initially determines the contribution of each point to the model fits; they are equally weighed by default, but can be changed using an n+2 length vector, where n is the number of ordered pairs and the first and last of the weights determines the contribution of the min and max ordered pairs; by default, all weights are initially set to 1 for equal contribution of each point.

The model complexity is determined by k. Multiple values of k are allowed, with the best value chosen by 10 fold cross validation. Before this happens, certain ordered pairs are removed based on the model errors. In each iteration, a GAM is fit using each selected value of k. Depending on the outlier argument, a point is "removed" from the model (i.e. its corresponding weight set to 0) if its residual is above the threshold for a proportion of fitted models, as determined by prop. If an anchor is an "identity" (idx = idy, detected in the selectAnchors by setting useID to TRUE), then setting useID here prevents its removal.

Other arguments, e.g. family, m, optimizer, bs, and method are GAM specific parameters from the mgcv R package. The family option is currently limited to the "scat" (scaled t) and "gaussian" families; scat family model fits are more robust to outliers than gaussian fits, but compute much slower. Type of splines are currently limited to basis splines ("bs" or "ps").

Value

metabCombiner with a fitted GAM model object

See Also

selectAnchors,fit_loess,

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)

p.comb = selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)
anchors = getAnchors(p.comb)

#version 1: using faster, but less robust, gaussian family
p.comb = fit_gam(p.comb, k = c(10,12,15,17,20), prop = 0.5,
    family = "gaussian", outlier = "MAD", coef = 2)


#version 2: using slower, but more robust, scat family
p.comb = fit_gam(p.comb, k = seq(12,20,2), family = "scat",
                     iterFilter = 1, coef = 3, method = "GCV.Cp")

#version 3 (with identities)
p.comb = selectAnchors(p.comb, useID = TRUE)
anchors = getAnchors(p.comb)
p.comb = fit_gam(p.comb, useID = TRUE, k = seq(12,20,2), iterFilter = 1)

#version 4 (using identities and weights)
weights = ifelse(anchors$labels == "I", 2, 1)
p.comb = fit_gam(p.comb, useID = TRUE, k = seq(12,20,2),
                     iterFilter = 1, weights = weights)

#version 5 (using boxplot-based outlier detection
p.comb = fit_gam(p.comb, k = seq(12,20,2), outlier = "boxplot", coef = 1.5)

#to preview result of fit_gam
plot(p.comb, pch = 19, outlier = "h", xlab = "CHEAR Plasma (30 min)",
     ylab = "Red-Cross Plasma (20 min)", main = "Example GAM Fit")
```

`fit_loess` *Fit RT Projection Model With LOESS*

Description

Fits a local regression smoothing spline through a set of ordered pair retention times. modeling one set of retention times (rty) as a function on the other set (rtx). Filtering iterations of high residual points are performed first. Multiple acceptable values of `span` can be used, with one value selected through 10-fold cross validation.

Usage

```
fit_loess(
  object,
  useID = FALSE,
  spans = seq(0.2, 0.3, by = 0.02),
  outlier = c("MAD", "boxplot"),
  coef = 2,
  iterFilter = 2,
  prop = 0.5,
  weights = 1,
  message = TRUE,
  control = loess.control(surface = "direct", iterations = 10)
)
```

Arguments

`object`	a `metabCombiner` object.
`useID`	logical. If set to TRUE, matched ID anchors detected from previous step will never be flagged outliers.
`spans`	numeric span values (between 0 & 1) used for loess fits
`outlier`	Thresholding method for outlier dection. If "MAD", the threshold is the mean absolute deviation (MAD) times `coef`; if "boxplot", the threshold is `coef` times IQR plus 3rd quartile of a model's absolute residual values.
`coef`	numeric (> 1) multiplier for determining thresholds for outliers (see `outlier` argument)
`iterFilter`	integer number of outlier filtering iterations to perform
`prop`	numeric. A point is excluded if deemed a residual in more than this proportion of fits. Must be between 0 & 1.
`weights`	Optional user supplied weights for each ordered pair. Must be of length equal to number of anchors (n) or a divisor of (n + 2)
`message`	Option to print message indicating function progress
`control`	control parameters for loess fits; see: ?loess.control

Value

metabCombiner object with model slot updated to contain a fitted loess model

See Also

selectAnchors,fit_gam

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)
p.comb = selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)

#version 1
p.comb = fit_loess(p.comb, spans = seq(0.2,0.3,0.02), iterFilter = 1)

#version 2 (using weights)
anchors = getAnchors(p.comb)
weights = c(2, rep(1, nrow(anchors)), 2)  #weight = 2 to boundary points
p.comb = fit_loess(p.comb, spans = seq(0.2,0.3,0.02), weights = weights)

#version 3 (using identities)
p.comb = selectAnchors(p.comb, useID = TRUE, tolmz = 0.003)
p.comb = fit_loess(p.comb, spans = seq(0.2,0.3,0.02), useID = TRUE)

#to preview result of fit_loess
plot(p.comb, fit = "loess", xlab = "CHEAR Plasma (30 min)",
     ylab = "Red-Cross Plasma (20 min)", pch = 19,
     main = "Example fit_loess Result Fit")
```

---

getAnchors *Get Ordered Retention Time Pairs*

Description

This returns the data frame of feature alignments used to anchor a retention time projection model, constructed by selectAnchors.

Usage

```
getAnchors(object)

## S4 method for signature 'metabCombiner'
getAnchors(object)
```

Arguments

object	metabCombiner object

Value

Data frame of anchor features

See Also

selectAnchors

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red")

p.comb <- metabCombiner(p30, p20)
p.comb <- selectAnchors(p.comb, windx = 0.05, windy = 0.03)

anchors <- getAnchors(p.comb)
```

---

getCoefficients *Obtain Last-Used Score Coefficients*

Description

Provides the last used weight arguments from `calcScores()` function. Returns empty list if `calcScores()` has not yet been called.

Usage

```
getCoefficients(object)

## S4 method for signature 'metabCombiner'
getCoefficients(object)
```

Arguments

object	metabCombiner object

Value

A list of the last used weight parameters:

A	Specific weight penalizing feature m/z differences
B	Specific weight penalizing retention time projection error
C	Specific weight penalizing differences in abundance quantiles

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red")

p.comb <- metabCombiner(p30, p20)
p.comb <- selectAnchors(p.comb, windx = 0.05, windy = 0.04, tolrtq = 0.15)
p.comb <- fit_gam(p.comb, k = 20, iterFilter = 1, family = "gaussian")
p.comb <- calcScores(p.comb, A = 90, B = 14, C = 0.5)

getCoefficients(p.comb)
```

---

getData *Get Processed Dataset*

Description

The metabData constructor creates a formatted dataset from the input, which may be accessed using this method.

Usage

```
getData(object)

## S4 method for signature 'metabData'
getData(object)
```

Arguments

object	metabData object

Value

Single Metabolomics Data Frame

Examples

```
data(plasma30)

p30 <- metabData(plasma30, samples = "CHEAR")
data <- getData(p30)
```

---

getExtra *Get Extra Data Column Names*

Description

Get Extra Data Column Names

Usage

```
getExtra(object, data = NULL)

## S4 method for signature 'metabCombiner'
getExtra(object, data = NULL)

## S4 method for signature 'metabData'
getExtra(object)
```

Arguments

object	metabCombiner or metabData object
data	dataset identifier for metabCombiner objects

Value

character vector of extra column names

Examples

```
data(plasma30)
p30 <- metabData(plasma30, samples = "CHEAR", extra = "Red")
getExtra(p30)
```

getModel *Get Fitted RT Model*

Description

Returns the last fitted RT projection model from a metabCombiner object of type "gam" or "loess".

Usage

```
getModel(object, fit = c("gam", "loess"))

## S4 method for signature 'metabCombiner'
getModel(object, fit = c("gam", "loess"))
```

Arguments

object	metabCombiner object
fit	Choice of model, "gam" or "loess"

Value

nonlinear retention time fit object

See Also

fit_gam, fit_loess

Examples

```
data(plasma30)
data(plasma20)
p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.005)
p.comb <- selectAnchors(p.comb, tolrtq = 0.15, tolQ = 0.2, windy = 0.02)
p.comb <- fit_gam(p.comb, iterFilter = 1, k = 20, family = "gaussian")
p.comb <- fit_loess(p.comb, iterFilter = 1, spans = 0.2)
model.gam <- getModel(p.comb, fit = "gam")
model.loess <- getModel(p.comb, fit = "loess")
```

getSamples *Get Sample Names From metabCombiner or metabData Object*

Description

Returns the sample names from one of the two datasets used in metabCombiner analysis, denoted as 'x' or 'y.'

Usage

```
getSamples(object, data = NULL)

## S4 method for signature 'metabCombiner'
getSamples(object, data = NULL)

## S4 method for signature 'metabData'
getSamples(object)
```

Arguments

object	metabCombiner or metabData object
data	dataset identifier for metabCombiner objects

Value

character vector of sample names. For metabCombiner objects these may come from the 'x' dataset (if data = "x") or the 'y' dataset (if data = "y").

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)

p.comb <- metabCombiner(xdata = p30, ydata = p20)

getSamples(p30)
getSamples(p.comb, data = "x")  #equivalent to previous
getSamples(p20)
getSamples(p.comb, data = "y")  #equivalent to previous
```

getStats *Get Object Statistics*

Description

Prints out a list of object-specific statistics for both `metabCombiner` and `metabData` objects

Usage

```
getStats(object)

## S4 method for signature 'metabCombiner'
getStats(object)

## S4 method for signature 'metabData'
getStats(object)
```

Arguments

`object`	metabCombiner or metabData object

Value

list of object-specific statistics

Methods (by class)

- `metabCombiner`: Method for 'metabCombiner' object

Examples

```
data(plasma30)
data(plasma20)
p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)

getStats(p30) #metabData stats

p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.005)
p.comb <- selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)
p.comb <- fit_gam(p.comb, iterFilter = 1, k = 20)

getStats(p.comb) #metabCombiner stats
```

iddata *Retrieve Feature Identities*

Description

This retrieves user-assigned feature identities from one or all constituent datasets of a `metabCombiner` object

Usage

```
iddata(object, data = NULL)

## S4 method for signature 'metabCombiner'
iddata(object, data = NULL)
```

Arguments

`object`	`metabCombiner` object
`data`	dataset identifier to extract information from; if NULL, extracts information from all datasets

Value

data frame of feature identities

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")
p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##retrieve all ids
ids <- iddata(p.comb, data = NULL)

##retrieve ids from p30
ids <- iddata(p.comb, data = "p30")
```

identityAnchorSelection *Select Matching Ids as Anchors*

Description

This is an optional helper function for `selectAnchors`. Uses identities to guide selection of ordered retention time pairs. If useID option is set to TRUE, it will select pairs of features with matching ID character strings before proceeding with iterative anchor selection.

Usage

```
identityAnchorSelection(cTable, windx, windy, useID, brackets)
```

Arguments

`cTable`	data frame, contains only feature ids, mzs, rts, Qs, & labels
`windx`	numeric positive retention time exclusion window in X dataset
`windy`	numeric positive retention time exclusion window in Y dataset
`useID`	logical. Operation proceeds if TRUE, terminates otherwise.
`brackets`	If useID = TRUE, bracketed identity strings of the types included in this argument will be ignored

Details

Identity anchors are allowed to violate constraints of m/z, Q, and rtq difference tolerances, and will not be removed if they fall within a rt exclusion window of other features. If a name appears more than once, only the pair with the highest relative abundance is selected.

Value

combinedTable with updated anchor labels

See Also

`selectAnchors`

isCombinedTable *Determine* combinedTable *Validity*

Description

Checks whether input object is a valid metabData.Returns an integer code if invalid. Function is used alongside combinerCheck.

Usage

```
isCombinedTable(object)
```

Arguments

object	Any R object.

Details

a proper combinedTable must have the following characteristics to be deemed valid for metabCombiner operations:

1) It must be a data.frame with at least 16 columns and at least 1 row 2) The first 16 columns must be named "rowID", "idx","idy","mzx","mzy","rtx", "rty", "rtProj","Qx","Qy","group","score","rankX","rankY","adductx", & "adducty" in this exact order 3) The first 16 columns must be of class: "numeric" "character","character", "numeric","numeric","numeric", "numeric", "numeric","numeric","numeric", "integer", "numeric", "integer", "integer","character", "character"

4) The group column must have no missing or negative values

Failing any one of these criteria causes an error

Value

0 if object is a valid Combiner Table; an integer code otherwise

isMetabCombiner *Determine if object is a valid metabCombiner object*

Description

Checks whether input object is a valid metabCombiner.Returns an integer code if invalid. Function is used alongside combinerCheck.

Usage

```
isMetabCombiner(object)
```

Arguments

object	Any R object.

Value

0 if object is a valid metabData object; an integer code otherwise

---

isMetabData *Determine validity of input metabData object*

Description

Checks whether input object is a valid metabData.Returns an integer code if invalid. Function is used alongside combinerCheck.

Usage

```
isMetabData(object)
```

Arguments

object	Any R object

Value

0 if object is a valid metabData object; an integer code otherwise.

---

iterativeAnchorSelection *Iterative Selection of Ordered Pairs*

Description

This is a helper function for selectAnchors. Anchors are iteratively selected from highly abundant feature pairs, subject to feature m/z, rt, & Q constraints set by the user.

Usage

```
iterativeAnchorSelection(cTable, windx, windy, swap = FALSE)
```

Arguments

cTable	data frame, contains only feature ids, mzs, rts, Qs, & labels
windx	numeric positive retention time exclusion window in X dataset.
windy	numeric positive retention time exclusion windown in Y dataset.
swap	logical. When FALSE, searches for abundant features in dataset X, complemented by dataset Y features; when TRUE, searches for abundant features in dataset Y, complemented by dataset X features.

Value

data frame of anchor feature alignments.

See Also

selectAnchors

---

labelRows *Annotate and Remove Report Rows*

Description

This is a method for annotating identity-matched, removable, & conflicting feature pair alignment (FPA) rows in the combinedTable report. Simple thresholds for score, rank, retention time error and delta score can computationally reduce the set of possible FPAs to the most likely compound matches. FPAs falling within some small measure (in score or mz/rt) of the top-ranked row are organized into subgroups to facilitate inspection; setting delta to 0 automatically reduces to 1-1 matches.

reduceTable behaves identically to labelRows, but with delta set to 0 & remove set to TRUE, automatically limiting to 1 - 1 feature matches constrained by rank and score threshold parameters. Rank threshold defaults are also stricter with reduceTable.

Usage

```
labelRows(
  object,
  minScore = 0.5,
  maxRankX = 3,
  maxRankY = 3,
  method = c("score", "mzrt"),
  delta = 0.1,
  maxRTerr = 10,
  remove = FALSE,
  balanced = TRUE,
  brackets_ignore = c("(", "[", "{")
)
```

```
reduceTable(
  object,
  maxRankX = 2,
  maxRankY = 2,
  minScore = 0.5,
  maxRTerr = 10,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

object	Either a metabCombiner object or combinedTable.
minScore	numeric minimum allowable score (between 0 & 1) for metabolomics feature pair alignments
maxRankX	integer maximum allowable rank for X dataset features.
maxRankY	integer maximum allowable rank for Y dataset features.
method	Conflict detection method. If equal to "score" (default), assigns a conflict subgroup if score of lower-ranking FPA is within some tolerance of higher-ranking FPA. If set to "mzrt", assigns a conflicting subgroup if within a small m/z & rt distance of the top-ranked FPA.
delta	numeric score or mz/rt distances used to define subgroups. If method = "score", a value (between 0 & 1) score difference between a pair of conflicting FPAs. If method = "mzrt", a length 4 numeric: (m/z, rt, m/z, rt) tolerances, the first pair for X dataset features and the second pair for Y dataset features.
maxRTerr	numeric maximum allowable error between model-projected retention time (rtProj) and observed retention time (rty)
remove	Logical. Option to keep or discard rows deemed removable.
balanced	Logical. Optional processing of "balanced" groups, defined as groups with an equal number of features from input datasets where all features have a 1-1 match.
brackets_ignore	character. Bracketed identity strings of the types in this argument will be ignored

Details

metabCombiner initially reports all possible FPAs in the rows of the combinedTable report. Most of these are misalignments that require removal. This function is used to automate most of the reduction process by labeling rows as removable or conflicting, based on certain conditions, and is performed after computing similarity scores.

A label may take on one of four values:

a) "": No determination made b) "IDENTITY": an alignment with matching identity "idx & idy" strings c) "REMOVE": a row determined to be a misalignment d) "CONFLICT": competing alignments for one or multiple shared features

The labeling rules are as follows: 1) Rows with matching idx & idy strings are labeled "IDENTITY". These rows are not labeled "REMOVE", irrespective of subsequent criteria. 2) Groups determined to be 'balanced': label rows with rankX > 1 & rankY > 1 "REMOVE" irrespective

of delta criteria 3) Rows with a score < minScore: label "REMOVE" 4) Rows with rankX > maxRankX and/or rankY > maxRankY: label "REMOVE" 5) Conflicting subgroup assignment as determined by method & delta arguments. Conflicting alignments following outside delta thresholds: labeled "REMOVE". Otherwise, they are assigned a "CONFLICT" label and subgroup number.

Value

updated combinedTable or metabCombiner object. The table will have three new columns:

labels	characterization of feature alignments as described
subgroup	conflicting subgroup number of feature alignments
alt	alternate subgroup for rows in multiple feature pair conflicts

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)
p.comb = selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)
p.comb = fit_gam(p.comb, k = 20, iterFilter = 1)
p.comb = calcScores(p.comb, A = 90, B = 14, C = 0.5)

###merge combinedTable and featdata
cTable = cbind.data.frame(combinedTable(p.comb), featdata(p.comb))

##example using score-based conflict detection method
lTable = labelRows(cTable, maxRankX = 3, maxRankY = 2, minScore = 0.5,
         method = "score", maxRTerr = 0.5, delta = 0.2)

##example using mzrt-based conflict detection method
lTable = labelRows(cTable, method = "mzrt", maxRankX = 3, maxRankY = 2,
                     delta = c(0.005, 0.5, 0.005,0.3), maxRTerr = 0.5)
```

---

labelRowsParam *List labelRows & reduceTable Defaults*

Description

List of default parameters for combinedTable row annotation and removal. See help(labelRows) or ?labelRows for more details. reduceTableParam loads parameters for the more automated reduceTable function

Usage

```
labelRowsParam(
  maxRankX = 3,
  maxRankY = 3,
  minScore = 0.5,
  delta = 0.1,
  method = "score",
  maxRTerr = 10,
  balanced = TRUE,
  remove = FALSE,
  brackets_ignore = c("(", "[", "{")
)

reduceTableParam(
  maxRankX = 2,
  maxRankY = 2,
  minScore = 0.5,
  maxRTerr = 10,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

maxRankX	maximum rank allowable for X features
maxRankY	maximum rank allowable for Y features
minScore	minimum score threshold; default: 0.5
delta	score distance or mz/rt difference tolerances for subgrouping; default: 0.1
method	thresholding method for subgroup detection ("score" or "mzrt"); default: "score"
maxRTerr	maximum allowable difference between predicted RT (rtProj) & observed RT (rty); default: 10 minutes
balanced	option to reduce balanced groups; default: TRUE
remove	option to eliminate rows determined as removable; default: FALSE
brackets_ignore	bracket types for ignoring string comparisons

Value

list of labelRows parameters

See Also

labelRows, metabCombine

Examples

```
lrParams <- labelRowsParam(maxRankX = 2, maxRankY = 2, delta = 0.1,
                           maxRTerr = 0.5)
```

metabBatches *Three LC-MS Metabolomics Batch Datasets*

Description

An example multi-batch LC-MS metabolomics analysis of human plasma, used to demonstrate batchCombine. Due to the large size of the full experimental data, only three of the batches are loaded here with a subset of the samples and features from each batch.

Usage

```
data(metabBatches)
```

Format

A list containing three identically formatted data frames

metabCombine *metabCombiner Wrapper Function*

Description

metabCombine wraps the five main metabCombiner workflow steps into a single wrapper function. Parameter list arguments organize program parameters by constituent package functions.

Usage

```
metabCombine(
  xdata,
  ydata,
  binGap = 0.005,
  xid = NULL,
  yid = NULL,
  means = list(mz = FALSE, rt = FALSE, Q = FALSE),
  fitMethod = "gam",
  anchorParam = selectAnchorsParam(),
  fitParam = fitgamParam(),
  scoreParam = calcScoresParam(),
  labelParam = labelRowsParam()
)
```

Arguments

xdata	metabData object. One of two datasets to be combined.
ydata	metabData object. One of two datasets to be combined.
binGap	numeric parameter used for grouping features by m/z. See ?mzGroup for more details.
xid	character identifier of xdata. If xdata is a metabData, assigns a new ID for this dataset; if xdata is a metabCombiner, must be assigned to one of the existing dataset IDs. See details for more information.
yid	character identifier of ydata. If ydata is a metabData, assigns a new ID for this dataset; if ydata is a metabCombiner, must be assigned to one of the existing dataset IDs. See details for more information.
means	logical. Option to take average m/z, rt, and/or Q from metabComber. May be a vector (length = 3), single value (TRUE/FALSE), or a list with names "mz", "rt", "Q" as names.
fitMethod	RT spline-fitting method, either "gam" or "loess"
anchorParam	list of parameter values for selectAnchors() function
fitParam	list of parameter values for fit_gam() or fit_loess()
scoreParam	list of parameter values for calcScores()
labelParam	list of parameter values for labelRows()

Details

The five main steps in metabCombine are 1) m/z grouping & combined table construction, 2) selection of ordered pair RT anchors, 3) nonlinear spline (Basis Spline GAM or LOESS) fitting to predict RTs, 4) score calculation and feature pair alignment ranking, 5) combined table row annotation and reduction. metabData arguments xdata & ydata and m/z grouping binGap are required for step 1.

Steps 2-5 are handled by anchors, fit, scores, & labels, respectively, with lists containing the argument values for each step expected for these arguments. selectAnchorsParam, fitgamParam, fitloessParam, calcScoresParam, & labelRowsParam load the default program values of selectAnchors, fit_gam, fit_loess, calcScores & labelRows, respectively. These program arguments should be modified as necessary for the datasets used for analysis.

By default, the RT fitting method (fitMethod) is set to "gam", which means the argument fit is a list of parameters for fit_gam; if the (fitMethod) argument is set to "loess", then the fit argument expects a list of fit_loess parameters.

Value

a metabCombiner object following complete analysis

See Also

selectAnchorsParam, fitgamParam, calcScoresParam, labelRowsParam, fitloessParam

Examples

```
data("plasma20")
data("plasma30")

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)

#parameter lists:
saParam <- selectAnchorsParam(tolrtq = 0.2, windy = 0.02, tolmz = 0.002)
fitParam <- fitgamParam(k = seq(12,15), iterFilter = 1, outlier = "boxplot",
                        family = "gaussian", prop = 0.6, coef = 1.5)
scoreParam <- calcScoresParam(A = 75, B = 15, C = 0.3)
labelParam <- labelRowsParam(maxRankX = 2, maxRankY = 2, delta = 0.1)

#metabCombine wrapper
p.combined <- metabCombine(xdata = p30, ydata = p20, binGap = 0.0075,
                           anchorParam = saParam, fitParam = fitParam,
                           scoreParam = scoreParam, labelParam = labelParam)

##to view results
p.combined.table <- combinedTable(p.combined)
```

`metabCombiner` *Form a metabCombiner object.*

Description

This constructs an object of type `metabCombiner` from a pair of metabolomics datasets, formatted as either `metabData` (single-dataset class) or `metabCombiner` (combined-dataset class). An initial table of possible feature pair alignments is constructed by grouping features into m/z groups controlled by the binGap argument

Usage

```
metabCombiner(
  xdata,
  ydata,
  binGap = 0.005,
  xid = NULL,
  yid = NULL,
  means = list(mz = FALSE, rt = FALSE, Q = FALSE)
)
```

Arguments

xdata	metabData or metabCombiner object
ydata	metabData or metabCombiner object
binGap	numeric parameter used for grouping features by m/z. See ?mzGroup for more details.
xid	character identifier of xdata. If xdata is a metabData, assigns a new ID for this dataset; if xdata is a metabCombiner, must be assigned to one of the existing dataset IDs. See details for more information.
yid	character identifier of ydata. If ydata is a metabData, assigns a new ID for this dataset; if ydata is a metabCombiner, must be assigned to one of the existing dataset IDs. See details for more information.
means	logical. Option to take average m/z, rt, and/or Q from metabComber. May be a vector (length = 3), a single value (TRUE/FALSE), or a list with names "mz", "rt", "Q" as names.

Details

This function serves as a constructor of the metabCombiner combined dataset class and the entry point in the main workflow for pairwise dataset alignment. Two arguments must be specified, xdata and ydata, which must be both metabData objects, both metabCombiner objects, or one metabData and one metabCombiner. Each scenario is listed here:

1) If xdata & ydata are metabData objects, a new metabCombiner object is constructed with an alignment of this pair. New character identifiers are assigned to each dataset (xid & yid, respectively); if these are unassigned, then "1" and "2" will be their respective ids. xdata & ydata will be the active "dataset x" and "dataset y" used for the paired alignment.

2) If xdata is a metabCombiner and ydata is a metabData, then the result is the existing metabCombiner xdata augmented by an additional dataset, ydata. One set of meta-data (id, m/z, rt, Q, adduct labels) from xdata is used for alignment with the respective information from ydata, which is controlled by the xid argument; see the datasets method for extracting existing dataset ids. A new identifier yid is assigned to ydata, which must be distinct from the current dataset identifier.

3) If xdata is a metabData and ydata is a metabCombiner, then a similar process to #2 occurs, with xdata augmented to the existing ydata object and one of the constitutent dataset's meta-data is accessed, as controlled by the yid argument. One major difference is that rts of ydata serve as the "reference" or dependent variable in the spline-fitting step.

4) If xdata and ydata are both metabCombiner objects, the resulting metabCombiner object aligns information from both combined datasets. As before, one set of values contained in xdata (specified by xid argument) is used to align to the values from ydata (controlled by yid argument). The samples and extra columns are concatenated from all datasets.

For metabCombiner object inputs, the mean of the numeric fields (m/z, rt, Q) from all constituent datasets can be used in alignment in place of values from a single dataset. These are controlled by the means argument. By default this is a list value with "mz", "rt" and "Q" as names, but may also accept a sinle logical or a length-3 logical vector. If set to a single logical, then all three fields are averaged (TRUE) or not averaged (FALSE). If a three-length argument is supplied (e.g. c(TRUE, FALSE, FALSE)), then the values correspond to m/z, rt, and Q respectively.

Value

a `metabCombiner` object constructed from xdata and ydata, with features grouped by m/z according to the binGap argument.

Note

If using a `metabCombiner` object as input, only one row is allowed per feature corresponding to its first appearance. It is strongly recommended to reduce the table to 1-1 paired matches prior to aligning it with a new dataset.

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)

p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075,
                       xid = "p30", yid = "p20")
```

---

metabCombiner-class *'metabCombiner' Combined Metabolomics Dataset Class*

Description

This is the main object for the `metabCombiner` package workflow. This object holds a combined feature table, along with a retention time warping model, the ordered pair anchors used to generate this model, important information organized by dataset, and key object statistics.

Slots

`combinedTable` data frame displaying all feature pair alignments, combining measurements of all possible shared compounds

`featdata` data frame of feature metadata (id, m/z, rt, Q, adduct)

`anchors` data frame of feature pairs used for RT warping model

`model` list containing the last fitted nonlinear model(s)

`datasets` list of constituent datasets from xdata & ydata inputs

`xy` current X & Y datasets

`nonmatched` list of data frames consisting of nonmatched features

`coefficients` list of last used A,B,C similarity weight values

`samples` list of sample name vectors from input datasets

`extra` list of extra column name vectors from input datasets

`stats` set of useful metabCombiner statistics

`metabData` *Constructor for the metabData object.*

Description

This is a constructor for objects of type `metabData`.

Usage

```
metabData(
  table,
  mz = "mz",
  rt = "rt",
  id = "id",
  adduct = "adduct",
  samples = NULL,
  Q = NULL,
  extra = NULL,
  rtmin = "min",
  rtmax = "max",
  misspc = 50,
  measure = c("median", "mean"),
  zero = FALSE,
  duplicate = c(0.0025, 0.05)
)
```

Arguments

`table`	Path to file containing feature table or data.frame object containing features
`mz`	Character name(s) or regular expression associated with data column containing m/z values. The first column whose name contains this expression will be selected for analysis.
`rt`	Character name(s) or regular expression associated with data column containing retention time values. The first column whose name contains this expression will be selected for analysis.
`id`	Character name(s) or regular expression associated with data column containing metabolomics feature identifiers. The first column whose name contains this expression will be selected for analysis.
`adduct`	Character name(s) or regular expression associated with data column containing adduct or chemical formula annotations. The first column whose name contains this expression will be selected for analysis.
`samples`	Character name(s) or regular expression associated with data columns. All numeric columns whose names contain these keywords are selected for analysis. If no keywords given, program searches longest stretch of remaining numeric columns.

Q	Character name(s) or regular expression associated with numeric feature abundance quantiles. If NULL, abundance quantiles are calculated from sample intensities.
extra	Character names of columns containing additional feature information, e.g. non-analyzed sample values. All columns containing these keywords selected and will be displayed in the final output.
rtmin	Numeric. Minimum retention time for analysis.
rtmax	Numeric. Maximum retention time for analysis.
misspc	Numeric. Threshold missingness percentage for analysis.
measure	Central quantitation measure, either "median" or "mean".
zero	Logical. Whether to consider zero values as missing.
duplicate	Numeric ordered pair (m/z, rt) duplicate feature tolerances. Pairs of features within these tolerances are deemed duplicates and one of the pair is removed (see: findDuplicates)

Details

Processed metabolomics feature table must contain columns for m/z, rt, and numeric sample intensities. Some optional fields such as identity `id` and `adduct` label columns may also be supplied. Non-analyzed columns can be included into the final output by specifying the names of these columns in the `extra` argument. All required arguments are checked for validity (e.g. no negative m/z or rt values, each column is used at most once, column types are valid, etc...).

Following this is a pre-analysis filtering of rows that are either: 1) Outside of a specified retention time range (`rtmin`,`rtmax`), 2) Missing in excess of `misspc` percent of analyzed samples, or 3) deemed duplicates by small pairwise <m/z, rt> differences as specified by the `duplicate` argument.

Remaining features are ranked by abundance quantiles, Q, using a central `measure`, either "median" or "mean." Alternatively, the abundance quantiles column can be specified in the argument `Q`.

Value

An object of class metabData containing the specific information specified by `mz`,`rt`,`samples`,`id`,`adduct`,`Q`,and `extra` arguments, and adjusted by pre-processing steps.

Examples

```
data(plasma30)

#samples: CHEAR; RedCross samples non-analyzed "extra" columns
p30 <- metabData(plasma30, mz = "mz", rt = "rt", id = "identity",
                 adduct = "adduct", samples = "CHEAR", extra = "RedCross")

getSamples(p30)  #should print names of 5 CHEAR Sample column names
getExtra(p30)    #should print names of 5 Red Cross Sample column names

#equivalent to above
p30 <- metabData(plasma30, id = "id", samples = "CHEAR", extra = "Red")
```

```
#analyzing Red Cross samples with retention time limitations (0.5-17.5min)
p30 <- metabData(plasma30, samples = "Red", rtmin = 0.5, rtmax = 17.5)
data = getData(p30)
range(data$rt)

#using regular expressions for field searches
p30.2 <- metabData(plasma30, id = "identity|id|ID", samples = ".[3-5]$")
getSamples(p30.2)    #should print all column names ending in .3, .4, .5
```

`metabData-class` *'metabData' Single Metabolomics Dataset Class*

Description

This class is designed to process and format input metabolomics feature tables. It stores the information from individual metabolomics datasets, including the formatted feature table, sample names, and feature statistics.

Slots

`data` formatted metabolomics data frame.

`samples` character vector of analyzed sample names

`extra` character vector of non-analyzed columns names

`stats` A list of dataset statistics

`filtered` A list of filtered dataset features

`mzdata` *Retrieve m/z Values*

Description

This retrieves feature m/z values from one or all constituent datasets of a `metabCombiner` object. Alternatively, the average m/z value can be retrieved.

Usage

```
mzdata(object, data = NULL, value = c("obs", "mean"))

## S4 method for signature 'metabCombiner'
mzdata(object, data = NULL, value = c("obs", "mean"))
```

Arguments

object	metabCombiner object
data	dataset identifier to extract information from; if NULL, extracts data frame information from all datasets
value	Either "obs" (observed - default option) or "mean" value

Value

data frame of m/z values (if NULL) or single vector of m/z values

data(plasma30) data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR") p20 <- metabData(head(plasma20,500), samples = "Red") p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##retrieve all m/z mz <- mzdata(p.comb, data = NULL)

##retrieve m/z from p30 mz <- mzdata(p.comb, data = "p30")

##retrieve mean m/z mz <- mzdata(p.comb, value = "mean")

---

mzGroup *Binning of mass spectral features in m/z dimension*

Description

Features in two input feature lists are grouped by their m/z values.

Usage

```
mzGroup(xset, yset, binGap)
```

Arguments

xset	data frame containing metabolomics features
yset	data frame containing metabolomics features
binGap	numeric gap value between consecutive sorted & pooled feature m/z values.

Details

The m/z values from both datasets are pooled, sorted, and binned by the binGap argument. Feature groups form when there is at least one pair of features from both datasets whose consecutive difference is less than binGap. Grouped features are joined together in combinedTable data report.

Value

list object containing updated xset & yset with group information

nonmatched *Get Nonmatched Features*

Description

Features that lack a any counterparts in the complementary dataset may be obtained from this method. If data is set to "x" or "y", will retrieve data from the current X or Y dataset, respectively. If data is set to NULL, will retrieve the list of nonmatched features.

Usage

```
nonmatched(object, data = "x")

## S4 method for signature 'metabCombiner'
nonmatched(object, data = "x")
```

Arguments

object	metabCombiner object
data	dataset identifier for metabCombiner objects; if NULL, returns full list of non-matched features

Value

Data frame of non-matched features corresponding to data argument

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red", rtmax = 17.25)
p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.005)

nnmx <- nonmatched(p.comb, data = "x")
nnmy <- nonmatched(p.comb, data = "y")
```

`objective` *Weight Parameter Objective Function*

Description

This function evaluates the A, B, C weight parameters in terms of score separability of matching versus mismatching compound alignments. Higher objective function value imply a superior weight parameter selection.

Usage

```
objective(
  cTable,
  idtable,
  A,
  B,
  C,
  minScore,
  mzdiff,
  rtdiff,
  qdiff,
  rtrange,
  adductdiff,
  penalty,
  matches,
  mismatches
)
```

Arguments

`cTable`	data frame. Abridged `metabCombiner` report table.
`idtable`	data frame containing all evaluated identities
`A`	Numeric weight for penalizing m/z differences.
`B`	Numeric weight for penalizing differences between fitted & observed retention times
`C`	Numeric weight for differences in Q (abundance quantiles).
`minScore`	numeric. Minimum score to count towards objective value.
`mzdiff`	numeric differences between feature m/z values
`rtdiff`	Differences between model-projected retention time value & observed retention time
`qdiff`	Difference between feature quantile Q values.
`rtrange`	range of dataset Y retention times
`adductdiff`	Numeric divisors of computed score when non-empty adduct labels do not match
`penalty`	positive numeric penalty wherever S(i,j) > S(i,i), i =/= j

matches	integer row indices of identity matches
mismatches	list of integer identity row mismatches for each identity

Details

First, the similarity scores between all grouped features are calculated as described in scorePairs

Then, the objective value for a similarity S is evaluated as:

$$OBJ(S) = \sum h(S(i,i)) - h(S(i,j)) - p(S(i,i) > S(i,j))$$

-S(i,i) represents the similarity between correct identity alignments
-S(i,j), represents the maximum similarity of i to grouped feature j, i =/= j (the highest-scoring misalignment)
-h(x) = x if x > minScore, 0 otherwise
-p(COND) = 0 if the condition is true, and a penalty value otherwise

This is summed over all labeled compound identities (e.g. idx = idy) shared between input datasets.

Value

A numeric value quantifying total separability of compound match similarity scores from mismatch scores, given A,B,C values

---

plasma20 *20 minute LC-MS Analysis of Human Plasma*

Description

An example metabolomics analysis of human plasma from Red Cross and CHEAR cohorts, plus pooled aliquots and blanks, acquired with a 20 minute total Reversed-Phase Liquid Chromatography & QTOF-MS instrument in the positive ionization mode.

Usage

```
data(plasma20)
```

Format

A data frame with 8910 rows and 22 columns.

plasma30 *30 minute LC-MS Analysis of Human Plasma*

Description

An example metabolomics analysis of human plasma from Red Cross and CHEAR cohorts, plus pooled aliquots and blanks, acquired with a 30 minute total Reversed-Phase Liquid Chromatography and a QTOF-MS instrument in the positive ionization mode.

Usage

```
data(plasma30)
```

Format

A data frame with 8286 rows and 22 columns

plot,metabCombiner,ANY-method *Plot metabCombiner Fits*

Description

This is a plotting method for metabCombiner objects. It displays ordered pairs and a curve fit computed using `fit_gam` or `fit_loess`, using base R graphics.

Usage

```
## S4 method for signature 'metabCombiner,ANY'
plot(x, y, ...)

plot_fit(
  object,
  fit = c("gam", "loess"),
  pcol = "black",
  lcol = "red",
  lwd = 3,
  pch = 19,
  outlier = "show",
  ocol = "springgreen4",
  legend = c("anchor", "outlier"),
  ...
)
```

Arguments

x	metabCombiner object
y	...
...	Other variables passed into graphics::plot
object	metabCombiner object
fit	choice of model (either "gam" or "loess").
pcol	color of the normal points (ordered RT pair) in the plot
lcol	color of the fitted line in the plot
lwd	line width of the curve fit between anchor points
pch	plot character type; see ?graphics::par for details
outlier	display option for outliers. If "show" or "s", treats outlier points like normal anchors; if "remove" or "r", removes outlier points from the plot; if "highlight" or "h", displays outliers with a different color and associated legend.
ocol	color of the outlier points; outlier argument must be set to "highlight" or "h"
legend	length-2 character vector indicating point labels in the legend if outlier argument set to "highlight" or "h"

Value

no values returned

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb = metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)
p.comb = selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windy = 0.02)
p.comb = fit_gam(p.comb, k = 20, iterFilter = 1, family = "gaussian")

##plot of GAM fit
plot(p.comb, main = "Example GAM Fit Plot", xlab = "X Dataset RTs",
     ylab = "Y Dataset RTs", pcol = "red", lcol = "blue", lwd = 5,
     fit = "gam", outliers = "remove")

grid(lwd =  2, lty = 3 ) #adding gridlines
```

Qdata — *Retrieve Relative Abundance Values*

Description

This retrieves feature Q values from one or all constituent dataset features of a `metabCombiner` object. Alternatively, the average Q value can be retrieved.

Usage

```
Qdata(object, data = NULL, value = c("obs", "mean"))

## S4 method for signature 'metabCombiner'
Qdata(object, data = NULL, value = c("obs", "mean"))
```

Arguments

`object`	`metabCombiner` object
`data`	dataset identifier to extract information from; if NULL, extracts information from all datasets
`value`	Either "obs" (observed - default option) or "mean" average value

Value

data frame or vector of relative ranked abundance (Q) values

data(plasma30) data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR") p20 <- metabData(head(plasma20,500), samples = "Red") p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##retrieve all Q Q <- Qdata(p.comb, data = NULL)

##retrieve Q from p30 Q <- Qdata(p.comb, data = "p30")

##retrieve mean Q Q <- Qdata(p.comb, value = "mean")

rtdata — *Retrieve Retention Time Values*

Description

This retrieves feature RT values from one or all constituent dataset features of a `metabCombiner` object. Alternatively, the average RT value can be retrieved.

Usage

```
rtdata(object, data = NULL, value = c("obs", "mean"))

## S4 method for signature 'metabCombiner'
rtdata(object, data = NULL, value = c("obs", "mean"))
```

Arguments

object	metabCombiner object
data	dataset identifier to extract information from; if NULL, extracts information from all datasets
value	Either"obs" (observed - default option) or "mean"

Value

data frame or vector of retention time values

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR")
p20 <- metabData(head(plasma20,500), samples = "Red")
p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

##retrieve all RTs
rt <- rtdata(p.comb, data = NULL)

##retrieve RTs from p30
rt <- rtdata(p.comb, data = "p30")

##retrieve mean RT
rt <- rtdata(p.comb, value = "mean")
```

---

scorePairs *Calculate Pairwise Alignment Scores*

Description

Helper function for calcScores & evaluateParams. Calculates a pairwise similarity score between grouped features using differences in m/z, rt, and Q.

Usage

```
scorePairs(A, B, C, mzdiff, rtdiff, qdiff, rtrange, adductdiff)
```

Arguments

A	Numeric weight for penalizing m/z differences.
B	Numeric weight for penalizing differences between fitted & observed retention times.
C	Numeric weight for differences in Q (abundance quantiles).
mzdiff	Numeric differences between feature m/z values
rtdiff	Differences between model-projected retention time value & observed retention time
qdiff	Difference between feature quantile Q values
rtrange	Range of dataset Y retention times
adductdiff	Numeric divisors of computed score when non-empty adduct labels do not match

Details

The score between two grouped features x & y is calculated as:

$$S = -exp(-A|mzx - mzy| - B|rty - rtproj|/rtrange - C|Qx - Qy|)$$

where mzx & Qx correspond to the m/z and abundance quantile values of feature x; mzy, rty, and Qy correspond to the m/z, retention time, and quantile values of feature y; rtproj is the model-projected retention time of feature x onto the Y dataset chromatogram and rtrange is the retention time range of the Y dataset chromatogram. A, B, C are non-negative constant weight parameters for penalizing m/z, rt, and Q differences. Values between 0 (no confidence alignment) and 1 (high confidence alignment).

Value

Numeric similarity score between 0 & 1

---

selectAnchors *Select Anchors for Nonlinear RT Model*

Description

A subset of possible alignments in the combinedTable are used as ordered pairs to anchor a retention time projection model. Alignments of abundant features are prominent targets for anchor selection, but shared identified features (i.e. feature pairs where idx = idy) may be used.

Usage

```
selectAnchors(
  object,
  useID = FALSE,
  tolmz = 0.003,
  tolQ = 0.3,
  tolrtq = 0.3,
  windx = 0.03,
  windy = 0.03,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

`object`	metabCombiner object.
`useID`	logical. Option to first search for IDs as anchors.
`tolmz`	numeric. m/z tolerance for prospective anchors
`tolQ`	numeric. Quantile Q tolerance for prospective anchors
`tolrtq`	numeric. Linear RT quantile tolerance for prosepctive anchors.
`windx`	numeric. Retention time exclusion window around each anchor in X dataset. Optimal values are between 0.01 and 0.05 min (1-3s)
`windy`	numeric. Retention time exclusion window around each anchor in dataset Y. Optimal values are between 0.01 and 0.05 min (1-3s)
`brackets_ignore`	If useID = TRUE, bracketed identity strings of the types included in this argument will be ignored.

Details

In order to map between two sets of retention times, a set of ordered pairs need to be selected for the spline fit. This function relies on mutually abundant features to select these ordered pairs. In iterative steps, the most abundant (as indicated by Q value) in one dataset is selected along with its counterpart, and all features within some retention time window specified by `windx` & `windy` arguments are excluded. This process is repeated until all features have been considered.

`tolQ` & `tolmz` arguments restrict to feature pairs that have differences in Q & m/z within these tolerances. `tolrtq` further limits to feature pairs those with relative differences in linear retention time quantiles, calculated as $rtqx = (rtx - min(rtx))/(max(rtx) - min(rtx))$ & $rtqy = (rty - min(rty))/(max(rty) - min(rty))$

Shared identities (in which idx & idy columns have matching, non-empty & non-bracketed strings) may be used if `useID` is set to TRUE. In this case, shared identities will be searched first and will not be subject to any of the restrictions in m/z, Q, or rt. The iterative process proceeds after processing of shared identities.

Value

`metabCombiner` object with updated `anchors` slot. This is a data.frame of feature pairs that shall be used to map between retention times using a GAM or LOESS model.

`idx`	identities of features from dataset X
`idy`	identities of features from dataset Y
`mzx`	m/z values of features from dataset X
`mzy`	m/z values of features from dataset Y
`rtx`	retention time values of features from dataset X
`rty`	retention time values of features from dataset Y
`rtProj`	model-projected retention time values from X to Y
`Qx`	abundance quantile values of features from dataset X
`Qy`	abundance quantile values of features from dataset Y
`adductX`	adduct label of features from dataset X
`adductY`	adduct label of features from dataset Y
`group`	m/z feature group of feature pairing
`labels`	anchor labels; "I" for identity, "A" for normal anchors

See Also

`getAnchors`, `fit_gam`, `fit_loess`

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.005)

##example 1 (no known IDs used)
p.comb <- selectAnchors(p.comb, tolmz = 0.003, tolQ = 0.3, windx = 0.03,
    windy = 0.02, tolrtq = 0.3)

##example 2 (known IDs used)
p.comb <- selectAnchors(p.comb, useID = TRUE, tolmz = 0.003, tolQ = 0.3)

##To View Plot of Ordered Pairs
anchors = getAnchors(p.comb)
plot(anchors$rtx, anchors$rty, main = "Selected Anchor Ordered Pairs",
    xlab = "rtx", ylab = "rty")
```

`selectAnchorsParam` *List selectAnchors Defaults*

Description

List of default parameters for anchor selection step of main package workflow, which can be used as input for the wrapper functions. See help(selectAnchors) or ?selectAnchors for more details.

Usage

```
selectAnchorsParam(
  useID = FALSE,
  tolmz = 0.003,
  tolQ = 0.3,
  tolrtq = 0.3,
  windx = 0.03,
  windy = 0.03,
  brackets_ignore = c("(", "[", "{")
)
```

Arguments

`useID`	Choice of using IDs for anchor selection; default: FALSE
`tolmz`	m/z tolerance for ordered pair features; default: 0.003
`tolQ`	Q tolerance for ordered pair features; default: 0.3
`tolrtq`	RT quantile tolerance for ordered pair features; default: 0.5
`windx`	X feature RT window parameter. Default: 0.03
`windy`	Y feature RT window parameter. Default: 0.03
`brackets_ignore`	bracket types for ignoring string comparisons

Value

list of selectAnchors parameters

See Also

`selectAnchors`, `metabCombine`

Examples

```
sa_param <- selectAnchorsParam(tolmz = 0.002, tolQ = 0.2, windy = 0.02)
```

write2file *Print metabCombiner Report to File.*

Description

Prints a combinedTable report to a file, specified by file argument. Output file has an empty line between each separate m/z group for ease of viewing.

Usage

```
write2file(object, file, sep = ",")
```

Arguments

object	metabCombiner object or combinedTable
file	character string naming the output file path
sep	Character field separator. Values within each row are separated by this character.

Value

no values returned

Examples

```
data(plasma30)
data(plasma20)

p30 <- metabData(plasma30, samples = "CHEAR")
p20 <- metabData(plasma20, samples = "Red", rtmax = 17.25)
p.comb <- metabCombiner(xdata = p30, ydata = p20, binGap = 0.0075)

p.comb <- selectAnchors(p.comb, tolmz = 0.003, tolrtq = 0.3, windy = 0.02)
p.comb <- fit_gam(p.comb, k = 20, iterFilter = 1)
p.comb <- calcScores(p.comb, A = 90, B = 14, C = 0.5)
p.comb <- labelRows(p.comb, maxRankX = 2, maxRankY = 2, remove = TRUE)


###using metabCombiner object as input
write2file(p.comb, file = "plasma-combined.csv", sep = ",")

###using combinedTable report and feature data as input
cTable <- cbind.data.frame(combinedTable(p.comb), featdata(p.comb))
write2file(cTable, file = "plasma-combined.txt", sep = "\t")
```

x *Obtain XY Dataset Identifier*

Description

metabCombiner alignment is performed in a pairwise manner between two datasets generically termed "X" & "Y". These methods prints the identifier associated with dataset X and Y, contained within the xy slot of a constructed metabCombiner object.

Usage

```
x(object)

y(object)

## S4 method for signature 'metabCombiner'
x(object)

## S4 method for signature 'metabCombiner'
y(object)
```

Arguments

object	metabCombiner object

Value

character X or Y dataset identifiers

data(plasma30) data(plasma20)

p30 <- metabData(head(plasma30,500), samples = "CHEAR") p20 <- metabData(head(plasma20,500), samples = "Red") p.comb <- metabCombiner(p30, p20, xid = "p30", yid = "p20")

#expected: "p30" x(p.comb)

#expected: "p20" y(p.comb)

Index