Package 'metaMS'

November 12, 2026

Type Package

Title MS-based metabolomics annotation pipeline

Version 1.48.0

Depends R (>= 4.0), methods, CAMERA, xcms (>= 1.35)

Imports Matrix, tools, robustbase, BiocGenerics, graphics, stats, utils

Suggests metaMSdata, RUnit

Description MS-based metabolomics data processing and compound annotation pipeline.

URL https://github.com/yguitton/metaMS

License GPL (>= 2)

biocViews ImmunoOncology, MassSpectrometry, Metabolomics

Encoding UTF-8

RoxygenNote 7.1.0

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

git_branch RELEASE_3_23

git_last_commit 82c42f6

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Ron Wehrens [aut] (author of GC-MS part, Initial Maintainer),
Pietro Franceschi [aut] (author of LC-MS part),
Nir Shahaf [ctb],
Matthias Scholz [ctb],
Georg Weingart [ctb] (development of GC-MS approach),
Elisabete Carvalho [ctb] (testing and feedback of GC-MS pipeline),
Yann Guitton [ctb, cre] (ORCID:
<https://orcid.org/0000-0002-4479-0636>),
Julien Saint-Vanne [ctb]

Maintainer Yann Guitton <yann.guitton@gmail.com>

Contents

metaMS-package . . . 3
addRI . . . 4
alignmentLC . . . 5
AnnotateTable . . . 6
annotations2tab . . . 8
constructExpPseudoSpectra . . . 9
createSTDdbGC . . . 10
createSTDdbLC . . . 12
errf . . . 14
exptable . . . 15
FEMsettings . . . 17
GCresults . . . 20
getAnnotationLC . . . 20
getAnnotationMat . . . 21
getFeatureInfo . . . 22
getPeakTable . . . 23
LCDBtest . . . 24
LCresults . . . 25
match2ExtDB . . . 25
matchExpSpec . . . 26
matchSamples2DB . . . 27
matchSamples2Samples . . . 28
metaMSsettings-class . . . 29
metaSetting-methods . . . 32
msp . . . 33
peakDetection . . . 35
plotPseudoSpectrum . . . 36
printString . . . 37
processStandards . . . 37
readStdInfo . . . 38
removeDoubleMasses . . . 39
runCAMERA . . . 40
runGC . . . 41
runLC . . . 43
threeStdsDB . . . 45
treat.DB . . . 46

Index **48**

metaMS-package *Analysis pipeline for MS-based metabolomics data*

Description

Analysis pipeline for MS-based metabolomics data: basic peak picking and grouping is done using functions from packages xcms and CAMERA. The main output is a table of feature intensities in all samples, which can immediately be analysed with multivariate methdos. The package supports the creation of in-house databases of mass spectra (including retention information) of pure chemical compounds. Such databases can then be used for annotation purposes.

Details

Index:

AnnotateFeature	Feature Wise Annotation
AnnotateTable	AnnotateTable
FEMsettings	FEM Settings for 'metaMS'
LCDBtest	Sample DB for LC-MS annotation
alignmentLC	LC alignment
construct.msp	Functions to handle msp-type objects (GC-MS)
constructExpPseudoSpectra	Create a list of all pseudospectra found in a GC-MS experiment of several samples.
createSTDdbGC	Create an in-house database for GC-MS annotation
createSTDdbLC	Create an in-house database for LC-MS annotation
exptable	Sample table for DB generation (LC)
generateStdDBGC	Convert an msp object into a GC database object
getAnnotationLC	get LC annotation
getAnnotationMat	Subfunction GC-MS processing
getFeatureInfo	Construct an object containing all meta-information of the annotated pseudospectra (GC-MS).
getPeakTable	get peak table
matchExpSpec	Match a GC-MS pseudospectrum to a database with a weighted crossproduct criterion.
matchSamples2DB	Match pseudospectra from several samples to an in-house DB (GC-MS)
matchSamples2Samples	Compare pseudospectra across samples (GC-MS)
peakDetection	Wrapper for XCMS peak detection, to be used for both GC-MS and LC-MS data.
plotPseudoSpectrum	Plot a pseudospectrum.
processStandards	Process input files containing raw data for pure standards.
readStdInfo	Read information of injections of standards

	from a csv file.
runCAMERA	Run CAMERA
runGC	Wrapper for processing of GC-MS data files
runLC	Wrapper for processing of LC-MS data files
treat.DB	Scaling of pseudospectra in an msp object.

The most important functions for running the pipeline are runGC and runLC - in-house databases are created by functions createSTDdbGC and createSTDdbLC.

Author(s)

Ron Wehrens [aut, cre] (author of GC-MS part), Pietro Franceschi [aut] (author of LC-MS part), Nir Shahaf [ctb], Matthias Scholz [ctb], Georg Weingart [ctb] (development of GC-MS approach), Elisabete Carvalho [ctb] (testing and feedback of GC-MS pipeline)

Maintainer: Ron Wehrens <ron.wehrens@fmach.it>

---

addRI *Add retention index information to an msp object*

Description

Given an msp object and the retention times and indices of a series of reference compounds, the function adds an RI field to every entry of the msp object. This will only be done if there is not already an RI field: existing information will not be overwritten.

Usage

```
addRI(mspobj, RIstandards, isMSP = TRUE)
```

Arguments

mspobj	An msp object.
RIstandards	A two-column matrix containing for the standards defining the RI scale both retention times and retention indices.
isMSP	Logical: if TRUE, then the spectra are stored in slot pspectrum; otherwise the spectra are simply the list elements of DB - matrices with three columns.

Value

An msp object, now also containing an RI slot.

Note

If the retention time of a compound is outside the range of the RI standards, NA will be used as RI value.

Author(s)

Ron Wehrens

Examples

```
if (require(metaMSdata)) {
  manual.fname <- list.files(system.file("extdata", package = "metaMSdata"),
                             pattern = "msp", full.names = TRUE)
  manual <- read.msp(manual.fname)
  RIstandards <- cbind("rt" = c(1.54, 1.68, 1.99, 2.7, 4.36, 6.81, 9.43,
                                11.88, 14.17, 16.34, 18.39, 20.33, 22.18,
                                23.93, 25.5, 27.18, 28.72, 30.26, 31.75,
                                33.19, 34.58, 35.95),
                       "RI" = (6:27)*100)

  manualRI <- addRI(manual, RIstandards)
}
```

---

alignmentLC *LC alignment*

Description

Performs grouping and retention time correction via xcms. The settings are specified as a list with a form similar to the one discussed in the help of FEMsettings. The sequence of actions depends on the type of retention time correction, which is specified inside the Retcor list inside the Alignment slot (see details). Integration of areas of missing peaks is performed depending on the fillPeaks setting.

Usage

```
alignmentLC(xset, settings)
```

Arguments

xset	The xcmsSet object resulting from peak detection
settings	The subset of settings contained into the "Alignment" element of the XCMSsettings list.

Details

The sequence of actions depends on the algorithm used by xcms for retention time correction. For the density-based approach the sequence is: 1) across sample grouping, 2) retention time correction, 3) second across sample grouping, 4) optional fill-peaks. For "obiwarp", instead, the sequence of actions is: 1) retention time correction, 2) across sample grouping, 3) optional fill-peaks. For across-sample grouping in xcms, the minsamp parameter can be specified in the settings either as a minimum number of samples (min.class.size) or as the fraction of samples per class

(min.class.fraction). If both are given the smallest number is used. If the retention time correction is performed by the density-based approach, the settings allow to express the xcms parameters missing and extra as fractions. When "obiwarp" is used for retention time correction, the sample with the bigger number of features is automatically selected as the "reference" sample. The xcms parameters for the retcor xcms function can be specified in the Retcor list included in the Alignment slot of FEMsettings.

Value

A grouped and rt-aligned xcmsSet object.

Author(s)

Ron Wehrens and Pietro Franceschi

See Also

FEMsettings

Examples

```
## Example of results
  data(LCresults) ## pre-compiled results
  LCresults$PeakTable
```

AnnotateTable *Annotate a Peaktable*

Description

Functions which annotate a peaktable on the bases af a database of standards. Not meant to be called directly by the user.

Usage

```
## Annotate one feature
AnnotateFeature(input, DB, settings, errf)

## Annotate a full table of features
AnnotateTable(peaktable, errf, DB, settings)
```

Arguments

input	A vector with three elements in the form (mz,rt,I).
peaktable	A peaktable (matrix) with three column corresponding to mz,rt and I values for a series of features.
errf	The file containing the error function used to predict the tolerance on the m/z value used for the matching against the DB.

DB	A dataframe used for the annotation. See the help of LCDBtest for a description of the DB.
settings	The subset of settings contained into the "match2DB" element of the XCMSsettings list.

Details

The annotation of each feature is performed by comparing its m/z value and its retention time to a database provided by the user. To account for shifts in retention time and mass occurring during data acquisition, the matching of a specific feature against the DB is done with a specific tolerance in mass and in retention time.

Retention time tolerance
The retention time tolerance is specified (in minutes) in the settings list (field rttol). This value is instrument- and chromatography-dependent.

m/z tolerance
The tolerance on the mass scale mainly depends on the characteristics of the spectrometer used for the acquisition. For Q-TOF instruments it has been recently shown (see references) that the optimal mass tolerance can be expressed as a function of the m/z value and of the logarithm of the ion intensity log10(I). As a trend, the mass drift will be bigger for smaller ions and for low intensity signals.
In the present implementation the tolerance in mass can be either fixed over the complete mass range or calculated as a function of the mz and I values of each feature. In the simplest case, the fixed mass tolerance is provided in the mzwindow (in Dalton!) element of the list of settings.
Alternatively, one can provide (supplying the errf argument) a function used to calculate the mz tolerance (in ppm!) as a function of the fields of the input vector ((mz,rt,I)).
As discussed in the publication, for a Waters Synapt Q-TOF the function is a linear model taking as inputs M = input["mz"], logI = log10(input["I"]). This error function can be calculated by analyzing the results of the injections of the chemical standards. To avoid unreasonable small errors where data for mz and I are not available, the minimum value for the mass tolerance is explicitly set in the settings (ppm). This value should match the technical characteristics of the spectrometer.

To reduce the number of false positives and make the annotation more reliable, a match is retained only if more than one feature associated to a specific compound is found in the list of features. How many "validation" features are required is defined in the list of settings in the minfeat element. At this validation level, another retention time tolerance is introduced: two or more features validate one specific annotation if their retention time are not very much different. This rt tolerance is also defined in the settings (the rtval field). As a general suggestion, rtval should be kept smaller than rttol. The latter, indeed, refers to the matching of a peaktable with a database which has been created from the injections of the chemical standards during different instrumental runs (maybe also with different columns). On the other hand, rtval accounts for smaller retention time shifts, occurring within the same LC run.

For the description of the structure of the DB, refer to the help of the LCDBtest dataset.

Value

A list with the following elements

annotation.table	A data.frame withe the results of the annotation and the reference to the DB

compounds	The names of the annotated compounds
IDs	The IDs of the annotated compounds
multiple.annotations	The features with multiple annotations
ann.features	The features with annotation

Author(s)

Pietro Franceschi

References

N. Shahaf, P. Franceschi, P. Arapitsas, I. Rogachev, U. Vrhovsek and R. Wehrens: "Constructing a mass measurement error surface to improve automatic annotations in liquid chromatography/mass spectrometry based metabolomics". Rapid Communications in Mass Spectrometry, 27(21), 2425 (2013).

Examples

```
## Example of results
  data(GCresults) ## pre-compiled results
  GCresults$PeakTable
```

---

annotations2tab *Conversion of a list of annotation results into a table*

Description

Function annotations2tab converts the output of matchSamples2DB into a table, in the case of multiple DB hits for one pseudospectrum distinguishing the "best" hit from the "alternative" hits. Function makeAnnotation prepares a data.frame object with the correct number of rows. Both functions are not meant to be called directly by the user.

Usage

```
annotations2tab(annlist, matches)
makeAnnotation(n)
```

Arguments

annlist	Annotation list, output of matchSamples2DB.
matches	Object containing all match factors - needed to distinguish the best match from the rest in the case of a double annotation.
n	Number of rows in the annotation data.frame (possibly zero).

Value

A three-column `data.frame`, containing the numbers of the pseudospectrum in the experimental data, the numbers of the best hits in the DB, and finally the alternative hits ain the DB

Author(s)

Ron Wehrens

`constructExpPseudoSpectra`

Create a list of all pseudospectra found in a GC-MS experiment of several samples.

Description

Function constructExpPseudoSpectra creates an msp object containing all the patterns referenced to in the annotation. The first argument is the output of function matchSamples2Samples and contains the full annotation matrix and the pseudospectra of the known unknowns; the second is the msp object containing the standards that are actually found. Not meant to be called directly by the user.

Usage

```
constructExpPseudoSpectra(allMatches, standardsDB)
```

Arguments

`allMatches`	Result of a call to `matchSamples2DB` or `matchSamples2Samples`: a list containing the element `annotation`.
`standardsDB`	Database of standard spectra, in msp format.

Value

A list of spectra. All spectra from the database that are referenced to in the annotation slot will be present, and will be followed by all unknown spectra. In order to be able to see which patterns in the experimental data are pointing to which spectra, an extra slot `DB.id` is added, containing the position of the reference spectrum in the original database (which is what the numbers of the annotation object are referring to). Renumbering this is impractical since some patterns may be seen as alternative annotations, but may have no hits of their own - including them would lead to rows containing only zeros.

Author(s)

Ron Wehrens

See Also

`matchSamples2DB`, `matchSamples2Samples`

Examples

```
## Example of results
  data(GCresults) ## pre-compiled results
  GCresults$PseudoSpectra
```

---

createSTDdbGC *Create an in-house database for GC-MS annotation*

Description

For creating an in-house instrument-specific annotation database, injections of pure standards need to be processed. All patterns in the vicinity of the retention time of the standard (to be provided by the user) will be compared to an external database - in case of a sufficient match, they will be retained in the database. The `generateStdDBGC` is not meant to be called directly by the user.

Usage

```
createSTDdbGC(stdInfo, settings, extDB = NULL, manualDB = NULL,
              RIstandards = NULL, nSlaves = 0)
generateStdDBGC(totalXset, settings, extDB = NULL, manualDB = NULL,
                RIstandards = NULL)
```

Arguments

`stdInfo`	Information of the standards, given in the form of a `data.frame`. Minimal information: `stdFile`, `Name`, `CAS`, monoisotopic mass (`monoMW`), and retention time (`rt`). The filenames in slot `stdFile` should include path information. If this argument is NULL, this function can be used to process a manually curated DB. Arguments `stdInfo` and `manualDB` cannot be both NULL.
`settings`	A list of settings, to be used in peak picking and pattern comparison.
`extDB`	The external database containing spectra, with which to compare the patterns found in the standards files.
`manualDB`	A database of manually curated spectra, that will be incorporated in the final DB without any further checks.
`totalXset`	A list of xset objects, as generated by `peakDetection`.
`RIstandards`	A two-column matrix containing for the standards defining the RI scale both retention times and retention indices. If not given, no RI values will be calculated and retention times will be used instead.
`nSlaves`	Number of cores to be used in peak picking.

Details

Function createSTDdbGC creates a database object containing validated pseudospectra for a number of compounds. The injections of the standards, described in the input object stdInfo, are processed using function processStandards; comparison with the external database, inclusion of manual compounds and final formatting are done in function generateStdDBGC. Several situations can be envisaged:

A: a series of injections of standards needs to be compared with a standard library, such as the NIST. In this case, both stdInfo and extDB need to be non-null, and the result will be a database in which the entries have a sufficient match with the external DB. If manualDB is also non-null, these entries will be added too (without checking).

B: for a series of injections no standard library information is available (extDB is NULL, and stdInfo is not), and the function simply returns all patterns eluting around the indicated retention time. This allows for subsequent manual validation and pruning. If manualDB is non-null, these entries will be added, but since this is a somewhat unusual thing to do, a warning will be given.

C: a manual database needs to be processed to be useable as a real database. This basically entails renaming the rt and rt.sd fields into std.rt and std.rt.sd, and a similar action for any RI field.

Value

The output of createSTDdbGC (and generateStdDBGC, which is the last function called in createSTDdbGC) is a list, where every entry describes one compound/spectrum combination. For use in annotation, the following fields are mandatory: Name, std.rt, pspectrum and monoMW.

Author(s)

Ron Wehrens

See Also

processStandards, generateStdDBGC

Examples

```
  data(threeStdsNIST)  ## provides object smallDB, excerpt from NIST DB
  ## Not run:
if (require(metaMSdata)) {
  ## Sitation A: create a DB of standards.
  ## first tell the system where to look
  data(threeStdsInfo)
  all.files <- list.files(system.file("extdata", package = "metaMSdata"),
                          pattern = "_GC_", full.names = TRUE)
  stdInfo[,"stdFile"] <- rep(all.files[3], 3)

  data(FEMsettings)    ## provides a.o. TSQXLS.GC, the GC settings file
  data(threeStdsNIST)  ## provides object smallDB, excerpt from NIST DB

  DB <- createSTDdbGC(stdInfo, TSQXLS.GC, extDB = smallDB)
  }
```

```
## saved in "threeStdsDB.RData" in the data directory of the metaMS
## package

## Situation B: do not check the data with an external database. Now
## the fields bestDBmatch and validation will be absent.
DB <- createSTDdbGC(stdInfo, TSQXLS.GC, extDB = NULL)

## Situation C: create a DB directly from an msp file (manual DB)
manual.fname <- list.files(system.file("extdata", package = "metaMSdata"),
                           pattern = "msp", full.names = TRUE)
manual <- read.msp(manual.fname)
DB <- createSTDdbGC(stdInfo = NULL, settings = TSQXLS.GC,
                    manualDB = manual)

## End(Not run)
```

`createSTDdbLC` *Create an in-house database for LC-MS annotation*

Description

For creating an in-house instrument-specific annotation database, injections of pure standards need to be processed. For each standard the analyst provides a validated reference value for retention time and m/z, generally corresponding to the major ionic signal for this compound. On the bases of this data, the database is constructed by automatically extracting the features identified in the vicinity of the retention time of the standard.
The function `generateSTDdbLC` is not meant to be called directly - use `createSTDdbLC` instead.

Usage

```
createSTDdbLC(stdInfo, settings, polarity, Ithr = 10, nSlaves = 0)
generateStdDBLC(stdxsets, settings, Ithr)
```

Arguments

`stdInfo`	Information of the standards, given in the form of a data.frame. Minimal information: `stdFile`, `compound`, `ChemSpiderID`, observed m/z (`mz.observed`), and retention time in miutes (`RTman`). The filenames in slot `stdFile` should include path information.
`settings`	A list of settings, to be used in peak picking and pattern comparison (see details).
`polarity`	The polarity of the injection: "positive" or "negative"
`Ithr`	The intensity threshold used to decide weather or not a feature should be included in the DB. Typically acting on the `maxo` value.
`stdxsets`	A list of CAMERA objects resulting from the analysis (performed by `processStandards`) of the standard injections listed in the `stdInfo` table.
`nSlaves`	Number of cores to be used in peak picking.

Details

The DB is created with the following workflow. Peak picking is performed on each standard file by using the settings specified in the settings list. CAMERA is used to group together the different features by considering their retention time and the correlation among the extracted ion traces. The list of features is searched looking for the values for mz and Rt included in the `stdInfo` table (see the help of `exptable` for more details), with the mass and retention time tolerances specified in the "DBconstruction" element of the settings list. In presence of positive match for the feature `f`, a spectral fingerprint is constructed by using all the features with an intensity bigger than `Ithr` which are in the same pcgroup of `f`. A match is retained only if the spectral fingerprint is composed of more than `minfeat` elements. This parameter is also included in the list of settings.

Value

A list with three elements.

`Reftable`	the original table used for the creation of the DB.
`Info`	a list with the settings used for the DB generation and the date.
`DB`	the DB which can be used in `runLC` for annotation.

Author(s)

Pietro Franceschi

Examples

```
if (require(metaMSdata)) {
  ##  load the manually curated table for the standards
  data(exptable)
  ## add location of cdf file from which the standards DB is going to be
  ## built - this depends on your platform and requires the metaMSdata package
  cdfpath <- system.file("extdata", package = "metaMSdata")

  ## files
  files <- list.files(cdfpath, "_RP_", full.names=TRUE)

  ## get the complete names for the files
  exptable$stdFile <-
    sapply(exptable$stdFile,
           function(x)
           files[grep(x,files)])

  ## Not run:
  ## load the settings for the analysis
  data(FEMsettings)

  ## set the minimum number of features to 2
  metaSetting(Synapt.RP, "DBconstruction")$minfeat  <- 2

  ## create the DB
  LCDBtest <- createSTDdbLC(stdInfo=exptable,
```

```
                            settings = Synapt.RP,
                            polarity = "positive",
                            Ithr = 20)

## End(Not run)
  ## saved in "LCDBtest.RData" in the data directory of the metaMS
  ## package
}
```

---

errf *Mass error surface for Waters Synapt Q-TOF spectrometers*

Description

This function can be used to calculate the optimal mz tolerance (in ppm) for annotation. The surface has been developed for a Waters Synapt QTOF spectrometer.

Usage

```
data(errf)
```

Format

`errf` is a linear model of the form `M + logI + M * logI`:

M The mz of the ion

logI Logarithm of the intensity of the ion.

Details

The function is a linear approaximation of the complete error function (see Reference). The latter has been calculated by comparing the measured and nominal mass of several hundreds of standards. The experiments were performed by using a Waters Synapt Q-TOF spectrometer so this specific surface is valid only for this model of spectrometers.

Author(s)

Pietro Franceschi

References

N. Shahaf, P. Franceschi, P. Arapitsas, I. Rogachev, U. Vrhovsek and R. Wehrens: "Constructing a mass measurement error surface to improve automatic annotations in liquid chromatography/mass spectrometry based metabolomics". Rapid Communications in Mass Spectrometry, 27(21), 2425 (2013).

Examples

```
## <--------------    direct use of the error function     -------------- >
## load the Synapt-QTOF error function
data(errf)

## predict the mass error in ppm
newdata <- data.frame(M = c(105, 131, 157), logI = c(1, .5, 1.4))
predict(errf, newdata) ## mass error in ppm


## <--------------    create a dummy error function     -------------- >
## dataset to evaluate it:
## "M" is the mz,
## "logI" is the log of the intensity
## "err" is the mass error in ppm. The error is the difference between the
## actual m/z of a known ion, and the one measured with the spectrometer

MErr.data <- data.frame("M" = seq(1,500,2),
                        "logI" = rnorm(250, mean = 5, sd = 1),
                        "err" = rnorm(250, mean = 40, sd = 5))

## create the linear model
dummy.model <- lm(err~M+logI, data = MErr.data)

## Not run:
## <--------------    Use this for the annotation     -------------- >
## load the example xcmsSet
data(LCresults)
data(FEMsettings)

## Run the analysis with an adaptive mass tolerance
result.adaptive.dummy <- runLC(xset = LCresults$xset,
                               settings = Synapt.RP,
                               DB = LCDBtest$DB,
                               errf = dummy.model)

## <-----------    Use the Synapt Q-TOF error function     ----------- >
## load the Synapt-QTOF error function
data(errf)

result.adaptive  <- runLC(xset = LCresults$xset,
                          settings = Synapt.RP,
                          DB = LCDBtest$DB,
                          errf = errf)

## End(Not run)
```

---

exptable *Sample table for the generation of a database of standards (LCMS)*

Description

An example of a table used to construct a database of standards. This particular table is used to obtain object `LCDBtest`.

Usage

```
data(exptable)
```

Format

`exptable` is a `data.frame` summarizing information on the chemical standards used for creating a database.

Column description:

ChemSpiderID The Chemspider ID of a specific compound.

compound A string containing the (human) readable name of the compound.

formula The formula of the compound.

M.ref The theoretical value for the observed m/z reference ion. This can be, for example, the protonated (de-protonated) molecular ion, a known adduct or a characteristic fragment.

mz.observed The manually validated m/z value of the reference ion.

RTman The manually validated retention time value for the standard.

stdFile The name (including the path) of the raw data file of the standard.

Details

Mandatory fields are: `ChemSpiderID`, `mz.observed`, `RTman`, and `stdFile`.

In the current implementation the `M.ref` value is not used in the creation of the DB. The difference between `M.ref` and `m.observed`, however, could be used to construct the mass error surface used during feature annotation.

The sample table is also an element (`Reftable`) of the LCDB list generated by `createSTDdbLC`.

Author(s)

Pietro Franceschi

See Also

`createSTDdbLC`

FEMsettings *Example settings for* metaMS

Description

Examples of settings needed for functions runLC and runGC: Synapt.RP, Synapt.NP, TSQXLS.GC and Orbitrap.RP. These four particular settings are fine-tuned for the analysis of LC-MS runs, both normal-phase and reverse-phase chromatography (Waters Synapt G1-Thermo Orbitrap)and GC-MS experiments (ThermoXLS TQQ).

Usage

```
data(FEMsettings)
```

Format

Four objects of class metaMSsettings.

Author(s)

Ron Wehrens and Pietro Franceschi

See Also

findPeaks, annotate

Examples

```
## Not run:
## The three sets of settings are created as follows:
Synapt.NP <- metaMSsettings(protocolName = "Synapt.QTOF.NP",
                            chrom = "LC",
                            PeakPicking = list(
                              method = "matchedFilter",
                              step = 0.05,
                              fwhm = 20,
                              snthresh = 4,
                              max = 50),
                            Alignment = list(
                              min.class.fraction = .3,
                              min.class.size = 3,
                              mzwid = 0.1,
                              bws = c(130, 10),
                              missingratio = 0.2,
                              extraratio = 0.1,
                              Retcor = list(
                                method = "linear",
                                family = "symmetric"),
                              fillPeaks = TRUE),
```

```
                            CAMERA = list(
                              perfwhm = 0.6,
                              cor_eic_th = 0.7,
                              ppm= 5))
metaSetting(Synapt.NP, "match2DB") <- list(
            rtdiff = 1.5,
            rtval = .1,
            mzdiff = 0.005,
            ppm = 5,
            minfeat = 2)
metaSetting(Synapt.NP, "DBconstruction") <- list(
            minfeat = 3,
            rttol = .3,
            mztol = .01)

## For reverse-phase LC, settings are very similar: the only difference
## is in the alignment settings
Synapt.RP <- Synapt.NP
metaSetting(Synapt.RP, "protocolName") <- "Synapt.QTOF.RP"
metaSetting(Synapt.RP, "Alignment") <- list(
            min.class.fraction = .3,
            min.class.size = 3,
            mzwid = 0.1,
            bws = c(30, 10),
            missingratio = 0.2,
            extraratio = 0.1,
            Retcor = list(
              method = "linear",
              family = "symmetric"),
            fillPeaks = TRUE)


## For the orbitrap.RP
Orbitrap.RP <- metaMSsettings(protocolName = "Orbitrap",
                              chrom = "LC",
                               PeakPicking = list(
                                 method = "centWave",
                                 ppm = 5,
                                 prefilter = c(3,10000),
                                 peakwidth = c(15,40)),
                              Alignment = list(
                                 bws = 30,
                                 min.class.fraction = 0.3,
                                 min.class.size = 3,
                                 mzwid = 0.01,
                                 Retcor = list(
                                  method = "obiwarp",
                                  profStep = 0.2),
                                 fillPeaks = TRUE),
                              CAMERA = list(
                                 perfwhm = 0.6,
                                 cor_eic_th = 0.7,
                                 ppm = 5))
```

```
metaSetting(Orbitrap.RP, "match2DB") <- list(
            rtdiff = 1.5,
            rtval = .1,
            mzdiff = 0.005,
            ppm = 5,
            minfeat = 2)
metaSetting(Orbitrap.RP, "DBconstruction") <- list(
            minfeat = 3,
            rttol = .3,
            mztol = .01)

## For the thermo TQ

TSQXLS.GC <- metaMSsettings("protocolName" = "TSQXLS.QQQ.GC",
                            "chrom" = "GC",
                            PeakPicking = list(
                              method = "matchedFilter",
                              step = 0.5,
                              steps = 2,
                              mzdiff = .5,
                              fwhm = 5,
                              snthresh = 2,
                              max = 500),
                           CAMERA = list(perfwhm = 1))
metaSetting(TSQXLS.GC, "DBconstruction") <- list(
            minintens = 0.0,
            rttol = .1,
            intensityMeasure = "maxo",
            DBthreshold = .80,
            minfeat = 5)
metaSetting(TSQXLS.GC, "match2DB") <- list(
            simthresh = 0.80,
            timeComparison = "rt",
            rtdiff = .5,
            RIdiff = 5,
            minfeat = 2)
metaSetting(TSQXLS.GC, "matchIrrelevants") <- list(
            irrelevantClasses = c("Bleeding", "Plasticizers"),
            timeComparison = "rt",
            RIdiff = 2,
            rtdiff = .05,
            simthresh = 0.70)
metaSetting(TSQXLS.GC, "betweenSamples") <- list(
            min.class.fraction = .5,
            min.class.size = 5,
            timeComparison = "rt",
            rtdiff = .05,
            RIdiff = 2,
            simthresh = .95)

## End(Not run)
```

GCresults *Results of metaMS for a small GC-MS data set*

Description

This data file contains the results of applying the metaMS pipeline to a small GC-MS data set consisting of repeated injections of a set of chemical standards. Its aim is to demonstrate the structure of the (intermediate) outcomes without having to do the slightly time-consuming calculations.

Usage

```
data(GCresults)
```

Details

Data object GCresults is the result of a complete execution of runGC, using the example database for annotation purposes.

See Also

runGC

getAnnotationLC *Get LC annotation*

Description

Main function for the annotation of an xcmsSet or CAMERA object. This function is not meant to be called directly. Use runLC instead.

Usage

```
getAnnotationLC(xs, settings, DB, errf)
```

Arguments

xs	The xcmsSet (or CAMERA) object to be annotated.
settings	The subset of settings contained into the match2DB elements of the settings list. See the help of FEMsettings for details.
DB	The database used within AnnotateTable for the annotation of the peaklist. See the help of FLCDBtest for details.
errf	The file containing the error function used to predict the m/z tolerance. See the help of AnnotateTable for details.

Details

The function extracts from the xs object a Peaktable with the intensities of the features across all the samples. Since this Peaktable is meant to be used only for annotation (and not for subsequent statistical analysis), the intensities are expressed asmaxo - the absolute maximum of the signal over the detected chromatographic peak (see the documentation of xcms for more details). Within the function the peaktable is converted into a matrix in the form (mz,rt,I) used by AnnotateTable. If xs contains more than one sample, the intensity is the maximum intensity of each feature across all the samples.

Value

A list with two elements. raw is the complete output of AnnotateTable. for_table is a data.frame which summarizes the outputs of the annotation (see AnnotateTable) and it is included in the output of the runLC main function.

Author(s)

Pietro Franceschi

References

N. Shahaf, P. Franceschi, P. Arapitsas, I. Rogachev, U. Vrhovsek and R. Wehrens: "Constructing a mass measurement error surface to improve automatic annotations in liquid chromatography/mass spectrometry based metabolomics". Rapid Communications in Mass Spectrometry, 27(21), 2425 (2013).

See Also

AnnotateTable, LCDBtest, FEMsettings

Examples

```
## Example of results
  data(LCresults) ## pre-compiled results
  LCresults$PeakTable
```

---

getAnnotationMat *Obtain relative quantitative annotation results for GC-MS*

Description

Function getAnnotationMat obtains relative intensities for features in individual samples. A robust linear regression is performed when the number of common features is five or larger; for 2-4 peaks a weighted linear regression is used, and if only one peak is in common the intensity ratio for this peak is used. Reference patterns are the patterns in the database of standards or the patterns in the "unknowns" element of the allMatches object. Not meant to be called directly by the user.

Usage

```
getAnnotationMat(exp.msp, pspectra, allMatches)
relInt(pat, refpat)
```

Arguments

exp.msp	List of experimental pseudospectra.
pspectra	Spectra from the in-house database.
allMatches	Match information in the form of a list.
pat, refpat	Both pseudospectra.

Value

Function getAnnotationMat returns a matrix containing all patterns (standards as well as unknowns) in the rows, and numeric values signifying relative intensities in all samples in the columns. These relative intensities are the quantities calculated by relInt, which simply returns one number.

Author(s)

Ron Wehrens

Examples

```
## Example of results
  data(GCresults) ## pre-compiled results
  GCresults$PeakTable
```

---

getFeatureInfo *Construct an object containing all meta-information of the annotated pseudospectra (GC-MS).*

Description

Function getFeatureInfo yields a data.frame with meta-information on all features detected in the samples. Features are rows; information is in the columns. Not meant to be called directly by the user.

Usage

```
getFeatureInfo(stdDB, allMatches, sampleList)
```

Arguments

stdDB	In-house database of standards.
allMatches	Object containing annotation information, generated by functions like matchSamples2DB and matchSamples2Samples.
sampleList	Object containing pseudospectra of samples.

Value

A `data.frame` summarizing all meta-information of the pseudospectra found in the samples. Exactly what columns are included depends on the information contained in the in-house database. The last column is always "rt".

Author(s)

Ron Wehrens

Examples

```
## Example of results
  data(GCresults) ## pre-compiled results
  GCresults$PeakTable
```

---

getPeakTable *Extract a peak table from an xcms or CAMERA object*

Description

Extracts the peak table (a `data.frame`) from an xcms (or CAMERA) object (without compound annotation). The peak table contains the mass and retention time for all the features and their intensities across the samples. This function is not meant to be called directly, but it is internally used by `runLC`, `getAnnotationLC`, `createSTDdbLC`.

Usage

```
getPeakTable(xs, intval = "into")
```

Arguments

`xs`	The xcmsSet (or CAMERA) object
`intval`	The intensity measure extracted form `xs`.

Details

The function process an `xs` object and returns for it a PeakTable which associates intensities to features and samples. The default measure for the intensity is `into` (the chromatographic peak area for a feature), but in the case of annotation, `maxo` (value for the intensity of the ion over the chromatographic peak) is used to measure the intensity. For a more detailed description of the possible intensity measures refer to the documentation of `xcms]`.

Value

A data frame with the intensity for each feature (rows) in all the samples (columns). Each feature is identified by its m/z value and retention time (in minutes). If the `xs` object is of class CAMERA, the results of the camera annotation (isotope, adduct, pcgroup) are included in the table.

Author(s)

Pietro Franceschi

Examples

```
## Example of results
  data(LCresults) ## pre-compiled results
  LCresults$PeakTable
```

`LCDBtest` *Sample DB for LC-MS annotation*

Description

This database has been generated using the `exptable` dataset to process the data included in the `metaMSdata` package.

Usage

```
data(LCDBtest)
```

Format

`LCDBtest` is a list with three elements:

The initial table used to generate the DB.

RefTable Some info on the DB: date of creation, and settings used.

DB A `data.frame` containing the actual information. See the details section.

Details

The `DB` `data.frame` contains the following fields:

ChemSpiderID The Chem spider identifier for the compound.

compound A human-readable name for the compound.

adduct The output of the CAMERA annotation of the standard file.

isotopes The output of the CAMERA annotation of the standard file.

mz, rt the mz and rt values.

maxo the `maxo` intensity value for a given feature in the injection.

validated a string which defines the origin of a specific feature.

Author(s)

Pietro Franceschi

See Also

`createSTDdbLC`, `exptable`, `FEMsettings`

LCresults *Result metaMS for a small LC-MS data set*

Description

This data file contains the result of applying the metaMS pipeline to a small LC-MS data set consisting of repeated injections of a set of chemical standards. Its aim is to demonstrate the structure of the outcomes without having to do the slightly time-consuming calculations.

Usage

```
data(LCresults)
```

Details

Data object `LCresult` contains the output list of `runLC`.

match2ExtDB *Match GC-MS spectra to an external reference DB*

Description

When building an in-house database, it is imperative that pseudospectra are validated. This function provides the possibility of automatic validation by comparing the spectra to an external reference database, such as the NIST. Not meant to be called directly by the user.

Usage

```
match2ExtDB(xsetList, extDB, settings)
```

Arguments

`xsetList`	A list of xcmsSet objects.
`extDB`	External database in msp format.
`settings`	Settings for the comparison, including as the most important the minimal number of features required in a valid pseudospectrum (`minfeat`), and the minimal agreement to speak of a match (`DBthreshold`).

Value

An msp object containing validated pseudospectra.

Author(s)

Ron Wehrens

matchExpSpec *Match a GC-MS pseudospectrum to a database with a weighted crossproduct criterion.*

Description

Function matchExpSpec calculates match factors for a pseudospectrum with all entries in the database. A plot of the best match can be provided.

Usage

```
matchExpSpec(pspec, DB, DB.treated, plotIt = FALSE,
             scale.p = c("sqrt", "none"), mass.weight = TRUE, ...)
```

Arguments

pspec	The pseudospectrum, a two- or three-column matrix where the third column (the retention time) will be ignored.
DB	Database of standards.
DB.treated	Logical, indicating whether the database has already been scaled (TRUE) or not.
plotIt	Logical: show best match?
scale.p	indicates whether "sqrt" scaling or no scaling ("none") is to be applied. This should usually be "sqrt".
mass.weight	Logical, indicating whether heavier masses receive higher weight. Should usually be TRUE.
...	Further arguments for the pseudospectrum plot (if shown).

Value

A vector of match factors.

Author(s)

Ron Wehrens

Examples

```
data(threeStdsNIST) ## gives smallDB, containing 78 patterns
data(threeStdsDB)   ## gives DB, containing 3 patterns :-D

matchExpSpec(DB[[1]]$pspectrum, smallDB, DB.treated = FALSE, plotIt = TRUE)
```

`matchSamples2DB` *Match pseudospectra from several samples to an in-house DB (GC-MS)*

Description

Compare experimental results to a database of pseudospectra. The result is a nested list, containing for every pseudospectrum in every sample either an index of a corresponding pattern in the DB (if a hit is found) or nothing (if no hit is found). Not meant to be called directly by the user.

Usage

```
matchSamples2DB(xset.msp, DB, settings, quick)
```

Arguments

`xset.msp`	An object containing a list of pseudospectra.
`DB`	The in-house database.
`settings`	The settings, in the form of a list.
`quick`	Logical. If TRUE, scaling of the pseudospectra (which is necessary for comparing to the database) will be done once using all masses in the pseudospectrum. This mode is routinely applied when comparing with a database of artefacts such as bleeding patterns or plasticizers. When comparing with a database of chemical standards, however, only peaks smaller than the molecular weight (+ 4, to allow for isotopes) should be taken into account in the scaling, and hence the scaling needs to be re-done for every comparison. This is _not_ so quick...

Value

A list object, with one element for each experimental sample. Every element again is a list with an entry for each pseudospectrum from that sample: if the element is empty, there is no match with the DB - a number points to the DB entry that gives a hit. Elements can contain more than one number; these will be split into one "best" annotation and "alternative" annotations in function `annotations2tab`.

Author(s)

Ron Wehrens

Examples

```
## Example of settings
  data(FEMsettings)
  metaSetting(object = TSQXLS.GC, field= "match2DB")
```

matchSamples2Samples *Compare pseudospectra across samples (GC-MS)*

Description

Function matchSamples2Samples matches pseudospectra across all samples - if a pseudospectrum is present at more or less the same retention time in several samples, it can get the status of "unknown". Exactly how much difference there may be between the pseudospectra and retention times, and how often it should be present, is determined by the settings. The auxiliary function match.unannot.patterns compares two msp objects, representing experimental samples. Both are not meant to be called directly by the user.

Usage

```
matchSamples2Samples(xset.msp.scaled, xset.msp, annotations, settings)
match.unannot.patterns(msp1, msp2, settings)
```

Arguments

xset.msp.scaled	Scaled version of all pseudospectra in the experimental patterns - a nested list, with one entry for each sample, and within every entry an element for each pseudospectrum.
xset.msp	Unscaled version of the first argument: both arguments are provided for efficiency reasons.
annotations	Annotations of all pseudospectra: only patterns without annotations will be considered.
settings	Settings determining what a valid "unknown" is. For an example, see the man page of FEMsettings, field matchIrrelevants.
msp1, msp2	lists of pseudospectra

Value

Function matchSamples2Samples returns an updated annotation object such as the one returned by matchSamples2DB, but now with an additional unknowns element, containing the pseudospectra that are recognized as "unknowns".

Function match.unannot.patterns returns a list of combinations of pseudospectra IDs, retention times (or retention indices) and match factors (only for those combinations that satisfy the criteria on retention time (index) and match factor).

Author(s)

Ron Wehrens

Examples

```
## Example of settings
  data(FEMsettings)
  metaSetting(object = TSQXLS.GC, field= "betweenSamples")
```

---

metaMSsettings-class *Class* "metaMSsettings"

Description

This class contains all settings needed to run the metaMS pipelines for LC-MS (runLC) and GC-MS (runGC). Slots PeakPicking, Alignment and CAMERA are simply handed over to the appropriate **xcms** and **CAMERA** functions; all other slots contain settings for **metaMS** functions.

Objects from the Class

Objects can be created by calls of the form metaMSsettings(...). See the example below.

Slots

Note: all slots describing retention times or retention time differences use minutes and not seconds. If a slot is only relevant for either GCMS or LCMS, this is indicated explicitly.

protocolName: Object of class "character": the name of the instrumental protocol, a unique identifier.

chrom: Object of class "character": chromatography. Either "LC" or "GC".

PeakPicking: Object of class "list": The parameters used for **xcms** peakpicking. See the arguments of findPeaks.

Alignment: Object of class "list": The parameters used for grouping and alignment. min.class.fraction and min.class.size are used to calculate the minsample xcms parameter. bws is a vector of the two bandwidths used for grouping before and after retention time alignent. missingratio and extraratio are used to set the values for missing and extra as a function of the number of samples. LC only.

CAMERA: Object of class "list": The parameters for CAMERA.

match2DB.rtdiff: Object of class "numeric": the maximal difference in retention time to match each feature with the entry in the DB.

match2DB.minfeat: Object of class "numeric": for LC, the minimal number of matching features within a retention time interval of width rtval before we speak of a hit. For GC, the minimal number of common masses for calculating a match factor.

match2DB.rtval: Object of class "numeric": the tolerance in retention time for features used in annotation. LC only.

match2DB.mzdiff: Object of class "numeric": the mass accuracy which is used if no error surface is provided. LC only.

match2DB.ppm: Object of class "numeric": the minimum mass tolerance allowed when the error surface is used. LC only.

`match2DB.simthresh`: Object of class `"numeric"`: the minimal match factor to speak of a hit. GC only.

`match2DB.timeComparison`: Object of class `"character"`: either "rt" or "RI". GC only.

`match2DB.RIdiff`: Object of class `"numeric"`: maximal retention index difference with DB entry (GC only).

`DBconstruction.minfeat`: Object of class `"numeric"`: the minimum number of features necessary to include a compound in the DB.

`DBconstruction.rttol`: Object of class `"numeric"`: the tolerance in retention time to match experimental features with the reference table.

`DBconstruction.mztol`: Object of class `"numeric"`: the tolerance in m/z (in dalton) to match experimental features with the reference table. LC only.

`DBconstruction.minintens`: Object of class `"numeric"`: the minimum intensity for a feature to be included in the list. GC only.

`DBconstruction.intensityMeasure`: Object of class `"character"`: either `"into"` or `"maxo"`. GC only.

`DBconstruction.DBthreshold`: Object of class `"numeric"`: minimal match factor with an external DB for a pseudospectrum to be included in the DB of standards. GC only.

`matchIrrelevants.irrelevantClasses`: Object of class `"character"`: classes of compounds are considered as irrelevant (a vector of string constants, which should exactly match the `Class` element in the DB entries). GC only.

`matchIrrelevants.simthresh`: Object of class `"numeric"`: the minimal match factor to speak of a hit. GC only.

`matchIrrelevants.timeComparison`: Object of class `"character"`: either "rt" or "RI". GC only.

`matchIrrelevants.rtdiff`: Object of class `"numeric"`: maximal retention time difference between two unknowns - this can be set to a very high value if a pattern is to be removed whatever the retention time. GC only.

`matchIrrelevants.RIdiff`: Object of class `"numeric"`: maximal retention index difference between two unknowns - this can be set to a very high value if a pattern is to be removed whatever the Retention Index. GC only.

`betweenSamples.min.class.fraction`: Object of class `"numeric"`: fraction of samples in which a pseudospectrum is present before it is regarded as an unknown. GC only.

`betweenSamples.min.class.size`: Object of class `"numeric"`: absolute number of samples in which a pseudospectrum is present before it is regarded as an unknown. GC only.

`betweenSamples.timeComparison`: Object of class `"character"`: either "rt" or "RI". GC only.

`betweenSamples.rtdiff`: Object of class `"numeric"`: max retention time difference between pseudospectra in different samples. GC only.

`betweenSamples.RIdiff`: Object of class `"numeric"`: max retention index difference between pseudospectra in different samples. GC only.

`betweenSamples.simthresh`: Object of class `"numeric"`: similarity threshold fo comparing pseudospectra in different samples. GC only.

Methods

metaSetting<- `signature(object = "metaMSsettings")`: change values in a metaMSsettings object.

metaSetting `signature(object = "metaMSsettings")`: get values from a metaMSsettings object.

show `signature(object = "metaMSsettings")`: show a metaMSsettings object.

Author(s)

Ron Wehrens

See Also

FEMsettings

Examples

```
showClass("metaMSsettings")

## Not run:
## The three sets of settings are created as follows:
Synapt.NP <- metaMSsettings(protocolName = "Synapt.QTOF.NP",
                            chrom = "LC",
                            PeakPicking = list(
                              method = "matchedFilter",
                              step = 0.05,
                              fwhm = 20,
                              snthresh = 4,
                              max = 50),
                            Alignment = list(
                              min.class.fraction = .3,
                              min.class.size = 3,
                              mzwid = 0.1,
                              bws = c(130, 10),
                              missingratio = 0.2,
                              extraratio = 0.1,
                              retcormethod = "linear",
                              retcorfamily = "symmetric",
                              fillPeaks = TRUE),
                            CAMERA = list(
                              perfwhm = 0.6,
                              cor_eic_th = 0.7,
                              ppm= 5))
metaSetting(Synapt.NP, "match2DB") <- list(
            rtdiff = 1.5,
            rtval = .1,
            mzdiff = 0.005,
            ppm = 5,
            minfeat = 2)
metaSetting(Synapt.NP, "DBconstruction") <- list(
            minfeat = 3,
```

```
            rttol = .3,
            mztol = .01)

## End(Not run)
```

`metaSetting-methods` *Get or set values in metaMSsettings objects*

Description

Accessor function for `metaMSsettings` objects, allowing to get or set values from individual slots.

Methods

`signature(object = "metaMSsettings")` Get or set values from individual slots in a `metaMSsettings` objects.

See Also

`FEMsettings`

Examples

```
## Not run:
## The three sets of settings are created as follows:
Synapt.NP <- metaMSsettings(protocolName = "Synapt.QTOF.NP",
                            chrom = "LC",
                            PeakPicking = list(
                              method = "matchedFilter",
                              step = 0.05,
                              fwhm = 20,
                              snthresh = 4,
                              max = 50),
                            Alignment = list(
                              min.class.fraction = .3,
                              min.class.size = 3,
                              mzwid = 0.1,
                              bws = c(130, 10),
                              missingratio = 0.2,
                              extraratio = 0.1,
                              retcormethod = "linear",
                              retcorfamily = "symmetric",
                              fillPeaks = TRUE),
                            CAMERA = list(
                              perfwhm = 0.6,
                              cor_eic_th = 0.7,
                              ppm= 5))
metaSetting(Synapt.NP, "match2DB") <- list(
            rtdiff = 1.5,
            rtval = .1,
```

```
            mzdiff = 0.005,
            ppm = 5,
            minfeat = 2)
metaSetting(Synapt.NP, "DBconstruction") <- list(
            minfeat = 3,
            rttol = .3,
            mztol = .01)

## End(Not run)
```

---

msp *Functions to handle msp-type objects (GC-MS)*

Description

Functions to construct, read, write and filter so-called msp objects, collections of spectra with additional information. This other information is of free format, but typically contains fields like "Name", "CAS", "ChemspiderID", and "rt".

Usage

```
construct.msp(spectra, extra.info)
read.msp(file, only.org = FALSE,
         org.set = c("C", "H", "D", "N", "O", "P", "S"), noNumbers = NULL)
filter.msp(msp, rtrange = NULL, mzrange = NULL, minintens = 0, minPeaks = 0)
write.msp(msp, file, newFile = TRUE)
xset2msp(xsetList, settings)
to.msp(object, file = NULL, settings = NULL, ndigit = 0,
       minfeat, minintens, intensity = c("maxo", "into"),
       secs2mins = TRUE)
SearchNIST(mspfile=NULL, savepath=NULL)
```

Arguments

spectra	a list of two-column or three-column numerical matrices. Two-column matrices only contain mz and intensity information, three-column matrices have a third column that gives retention times for the individual peaks.
extra.info	a nested list containing all other slots that should be included in the msp object. The length of this list should be equal to the number of spectra in the first argument.
file	name of input or output file containing data in msp format. If NULL in function to.msp, the input object is returned as an msp object. If non-null, the information will be written to the file with this name.
only.org	logical: if TRUE this will skip all entries that have non-organical elements in the "Formula" field.
org.set	List of elements to be considered as organic.

noNumbers	Optional argument, indicating which fields are not to be converted into numeric entries. If not given, this currently defaults to the following fields: Name, CAS, stdFile, date, validated, ChemspiderID, SMILES, InChI, Class, comment, csLinks.
msp	msp object.
rtrange, mzrange, minfeat, minintens, minPeaks	restrictions to what constitutes genuine pseudospectra in terms of minimal numbers of features, minimum feature intensity, and the intensity measure used (peak area or peak height).
intensity	the measure to use for the intensity of a peak: either the peak height ("maxo") or area ("into").
ndigit	number of digits for the mz values - use 0 for nominal mass data.
newFile	logical: if TRUE starts a new file, otherwise appends.
xsetList	a list of xcmsSet objects.
settings	list containing settings.
object	either an object of class "xsAnnotate" or a peaktable containing mz, rt and I information. Function to.msp will convert this object into a list that can be written as an msp file.
secs2mins	logical: if TRUE converts retention times to minutes (otherwise seconds).
mspfile	a msp file generated by to.msp function
savepath	Path to the directory were NIST MSsearch program results will be saved

Details

Even though the msp format handled by these functions is quite flexible, there are a couple of requirements that are not always satisfied by msp files generated by other software, the most important one being that one line may only contain one keyword. If more than one keyword is present, the second will likely not be read. Furthermore, the current implementation assumes that peaks in pseudospectra are represented in the form mz1, I1; mz2, I2;, etcetera - in other cases these are in brackets. To solve such issues, the most easy fix for the moment is to edit the msp file and change things globally.

Other issues may arise with the keywords. While read.msp will be able to read msp files with non-standard keywords, the metaMS package expects at least the following list to be present (case-sensitive): “Name”, “validated”, “CAS”, “rt”, “monoMW”, and “Class”. The final keyword is always “Num Peaks”, and should be followed by the list of mz-I combinations.

Value

Most of the functions here create msp files.

Author(s)

Ron Wehrens

Examples

```
data("threeStdsDB")
## Not run:
write.msp(DB, file = "huhn.msp")
DB2 <- read.msp("huhn.msp")

## End(Not run)
```

`peakDetection` *Wrapper for XCMS peak detection, to be used for both GC-MS and LC-MS data.*

Description

XCMS peak detection using settings, defined for individual laboratories and depending on the chromatographic and mass-spectrometric characteristics of the instruments at hand.

Usage

```
peakDetection(files, settings, rtrange = NULL, mzrange= NULL,
              convert2list = FALSE, nSlaves = 0)
```

Arguments

`files`	input files (including path names) that will be processed by xcms.
`settings`	a list of settings that will be passed on to the xcmsSet function. See the help of `FEMsettings` for a detailed description of the fields in this list.
`rtrange`	If non-NULL, a vector to subset the region of the chromatography retained for further analysis (given in minutes).
`mzrange`	If non-NULL, a vector indicating the subset of the mass spectrum retained for the analysis.
`convert2list`	logical. If `TRUE`, the xcmsSet object will be split into a list of one-sample objects which will be treated separately. This is useful not only in GC-MS data, where peak alignment is avoided, but also in setting up databases of standards, where no common peaks between injections are expected to occur.
`nSlaves`	Number of cores to be used in the peak picking.

Value

Either an xcmsSet object, or a list of one-sample xcmsSet objects.

Author(s)

Ron Wehrens and Pietro Franceschi

Examples

```
## Example of settings
  data(FEMsettings)
  metaSetting(object = TSQXLS.GC, field= "PeakPicking")
```

`plotPseudoSpectrum` *Plot a pseudospectrum.*

Description

Auxiliary function for plotting a particular pseudospectrum. M/z values are in the first column of the matrix, and an intensity measure (either maxo, into or something else) in the second. The third column is disregarded, usually contains retention time information

Usage

```
plotPseudoSpectrum(psspc, ...)
```

Arguments

`psspc`	Pseudospectrum, consisting of a two- or three-column matrix. The first column contains the m/z values, the second the intensities. A third column containing retention time information may be present, but is not used in this function.
`...`	Additional graphical parameters.

Details

A stick spectrum is shown on the graphical device.

Value

A three column matrix m/z and Intensity and Retention time.

Author(s)

Ron Wehrens

Examples

```
data("threeStdsDB")
plotPseudoSpectrum(DB[[1]]$pspectrum)
```

printString *Functions for metaMS-formatted text output*

Description

Functions to present progress output, warnings or information, in a consistent way in the console window. Not meant to be called by the user.

Usage

```
printString(..., screenwidth = 72)
printWarning(...)
printInfo(...)
```

Arguments

`...`	Text strings. These will be concatenated inside the function.
`screenwidth`	Width of the text field.

Value

A string to print.

Author(s)

Ron Wehrens and Pietro Franceschi

processStandards *Process input files containing raw data for pure standards.*

Description

Peak picking and further processing for raw data of pure standards, including CAMERA processing. This function is not meant to be called directly - use `createSTDdbLC` or `createSTDdbGC` instead.

Usage

```
processStandards(stdInfo, settings, polarity = NULL, nSlaves)
```

Arguments

stdInfo	Object describing the pure standards: a data.frame containing, e.g., the name of the file, the name of the standard, descriptors like CAS or Chemspider IDs, etcetera.
settings	Settings list, containing sublists for peak picking and CAMERA grouping (GC-MS) or annotation (LC-MS).
polarity	Polarity of the analysis (used for CAMERA). Possible values are "positive" or "negative". Ignored for GC-MS.
nSlaves	Number of cores to be used in peak picking.

Value

A list of CAMERA objects resulting from the analysis of the standard injections listed in the stdInfo table.

Author(s)

Ron Wehrens and Pietro Franceschi

See Also

xcmsSet, runCAMERA

Examples

```
## Example of results
  data(GCresults) ## pre-compiled results
  GCresults$PeakTable
```

---

readStdInfo *Read information of GC injections of standards from a csv file*

Description

The csv contains all information necessary to process a series of injections of standards (GC-MS). Required fields: Name, RTman, monoMW, stdFile, and an identifier such as CAS or ChemspiderID. At the moment, the system is completely based on CAS, although this may change in the future.

Usage

```
readStdInfo(stdInfo, InputDir, sep = "", dec = ".", ...)
```

Arguments

stdInfo	Input file in csv format, containing information on standards.
InputDir	Location of input files.
sep, dec, ...	optional arguments to read.table.

Details

In addition to reading all information on the chemical standards (whcih will eventually be transferred into an in-house database), the function checks whether some input files are unavailable, and whether some data files are not used. In the first case, an error is returned, in the second case a warning.

Value

A data.frame

Author(s)

Ron Wehrens

Examples

```
if (require(metaMSdata)) {
  ## this will lead to the completed version of the R object that is also
  ## available by typing "data(threeStdsInfo)", now containing the
  ## directory information that is not available in the RData object.

  input.file <- list.files(system.file("extdata", package = "metaMSdata"),
                           pattern = "csv", full.names = TRUE)
  threeStdsInfo <- readStdInfo(input.file,
                               system.file("extdata", package = "metaMSdata"),
                               sep = ";", dec = ",")
  ## only one of the files is used to set up the database, the others
  ## are for testing annotation
}
```

---

removeDoubleMasses *Remove double m/z entries in a pseudospectrum*

Description

Since in nominal-mass GC data m/z values are rounded to whole numbers, in some cases an m/z value may occur multiple times - in that case the mean of the intensities is kept (and the retention times are averaged). removeDoubleMasses takes a list of spectra in the form of a three-column matrix, (mz, I, rt), summing the intensities and averaging the retention times for multiple identical masses. Not meant to be called directly by the user.

Usage

```
removeDoubleMasses(spclist)
```

Arguments

spclist	A list of spectra, each one consisting of a three-column matrix (mz, I, rt).

Value

The function returns a list of spectra, where all "double" peaks have been averaged.

Author(s)

Ron Wehrens

`runCAMERA` *The CAMERA element in the metaMS pipeline*

Description

Run CAMERA package with settings from the settings list (see `FEMsettings`). Works both for LC and GC. Not meant to be called directly by the user.

Usage

```
runCAMERA(xset, chrom = c("LC", "GC"), settings, polarity, quick = TRUE)
```

Arguments

`xset`	For LC, an `xcmsSet` object; for GC, a list of `xcmsSet` objects.
`chrom`	The type of chromatography. Either "LC" or "GC".
`settings`	The subset of settigs contained into the "CAMERA" element of the settings list.
`polarity`	The polarity of the injection, used by CAMERA to look for common adducts.
`quick`	Only relevant for LC data. If `TRUE`, CAMERA runs only `groupFWHM` and `findIsotopes`. If `FALSE`, correlations between EICs are calculated and used for grouping (which can take some time).

Details

In the case of LC the function is used in data analysis and DB creation. In the first case, it increases the level of the annotation and it it, by default, run with `quick = TRUE`. For DB creation, the grouping of the features into "pcgroups" (features with similar retention time) is used to choose the features to be included into the database. In this case camera is run with `quick = FALSE`: `quick` determines whether or not the correlation among the extracted ion chromatograms should be used to "validate" the pcgroups.

For GC data, only the grouping done by `groupFWHM` is performed: basically this clusters peaks with a similar retention time.

Value

An annotated xcmsSet object (an object of class CAMERA).

Author(s)

Ron Wehrens and Pietro Franceschi

See Also

annotate, groupFWHM

Examples

```
## Example of results
  data(LCresults) ## pre-compiled results
  LCresults$PeakTable
```

---

runGC *Wrapper for processing of GC-MS data files*

Description

Main function of the pipeline for GC-MS data processing. It includes XCMS peak detection, definition of pseudospectra, and compound identification by comparison to a database of standards. The function also takes care of removal of artefacts like column bleeding and plasticizers, and definition of unknowns, consistently present across samples.

Usage

```
runGC(files, xset, settings, rtrange = NULL, DB = NULL,
      removeArtefacts = TRUE, findUnknowns = nexp >= mcs,
      returnXset = FALSE, RIstandards = NULL, nSlaves = 0)
```

Arguments

files	input files, given as a vector of strings containing the complete paths. All formats supported by XCMS can be used.
xset	alternatively, one can present a list of xcmsSet objects for whom CAMERA grouping has been done. In this case, only the annotation process will be done. If both files and xset are given, the former takes precedence.
settings	a nested list of settings, to be used at individual steps of the pipeline.
rtrange	part of the chromatograms that is to be analysed. If given, it should be a vector of two numbers indicating minimal and maximal retention time (in minutes).
DB	database containing the spectra of the pure standards. At least the following fields should be present: Name, std.rt, pspectrum and monoMW.
removeArtefacts	logical, whether or not to remove patterns identified as (e.g.) column bleeding. Only performed if a database containing such patterns is available.
findUnknowns	logical, whether to find patterns without identification that are present consistently in several samples. The default is to use TRUE if the number of samples is larger than the min.class.size setting in the 'betweenSamples' metaSetting.

returnXset	logical: should the XCMS output be returned? If yes, this is a a list of xcmsSet objects, one element for each input file.
RIstandards	A two-column matrix containing for the standards defining the RI scale both retention times and retention indices. If not given, no RI values will be calculated and retention times will be used instead.
nSlaves	Number of cores to be used in peak picking.

Value

A list with the following elements:

PeakTable	data.frame containing annotation information. Every line is a feature, i.e. a pseudospectrum. The first columns are used to give information about these features, a.o. compound name, CAS and Chemspider IDs, etcetera. The last of these meta-information columns is always the one giving the retention time: "rt". After that, columns correspond to input files, and give measures of intensities for every single one of the features. If a feature is not detected in a sample, this is indicated with "0" (zero).
PseudoSpecra	A list of pseudospectra in msp format, in the same order as the rows in the PeakTable.
xset	optionally, the xcmsSet object is returned, which can be useful for more detailed inspection of the results. It can also be used as an input for runGC, e.g., to test different annotation settings independently of the xcms/CAMERA part.
sessionInfo	The output of sessionInfo() to keep track of the sw version used for the processing

Author(s)

Ron Wehrens

References

R. Wehrens, G. Weingart and F. Mattivi: "An open-source pipeline for GC-MS-based untargeted metabolomics". Submitted.

See Also

msp, treat.DB, runCAMERA, peakDetection, matchSamples2DB, matchSamples2Samples, getAnnotationMat, addRI

Examples

```
## analysis of an xset object
data(threeStdsDB)
data(FEMsettings)

data(GCresults) ## pre-compiled results
names(GCresults)
```

```
## Not run:
if (require(metaMSdata)) {
  ## object GCresults is created by
  cdfdir <- system.file("extdata", package = "metaMSdata")
  cdffiles <- list.files(cdfdir, pattern = "_GC_",
                         full.names = TRUE, ignore.case = TRUE)
  GCresults <- runGC(files = cdffiles, settings = TSQXLS.GC, DB = DB,
                     returnXset = TRUE)

  ## to start directly from the XCMS/CAMERA results and not include
  ## peak picking in the pipeline, simply provide the "xset" argument
  ## rather than the "files" argument.

  ## no annotation database
  result.noannot <- runGC(xset = GCresults$xset, settings = TSQXLS.GC)
  }

## End(Not run)
```

---

runLC *Wrapper for processing of LC-MS data files*

Description

Main function of the pipeline for LC-MS data processing. It includes XCMS peak detection, grouping and alignment, CAMERA and feature annotation by comparison to a database of standards. The function also calculates the mass tolerance on the bases of the ion intensity and ion mass.

Usage

```
runLC(files, xset, settings, rtrange = NULL, mzrange = NULL, DB = NULL,
      polarity = "positive", errf = NULL, returnXset = FALSE,
      intensity = "into", nSlaves = 0)
```

Arguments

`files`	input files, given as a vector of strings containing the complete paths. All formats supported by XCMS can be used.
`xset`	alternatively, one can present an object of class `xsAnnotate`, instead of the input files. In that case all of the XCMS and CAMERA tools: peak picking, CAMERA grouping, retention time correction and filling of missing peaks will be considered already done, and only annotation will be performed. This can be useful if one wants to compare different settings for annotation. If both `files` and `xset` are given, the former takes precedence.
`settings`	nested list of settings, to be used at individual steps of the pipeline. See the help of `FEMsettings` for details.
`rtrange`	An optional vector to slice the retention time range (in minutes).

mzrange	An optional vector to slice the mass spectrum
DB	database containing the spectra of the pure standards. For the description refer to the LCDBtest help.
polarity	The polarity of the analysis used for CAMERA annotation. Either "positive" or "negative".
errf	A model used to calculate the mass tolerance in ppm for each features on the bases of its mass, retention time and intensity. For further details refer to the help of AnnotateTable.
returnXset	logical: should the XCMS output be returned? If yes, this is a a list of xcmsSet objects, one element for each input file.
intensity	The intensity measure used in the output peaktable. The available intensities are the ones provided by xcms. The default one is the total intensity (integral) of the feature on the detected chromatographic peak.
nSlaves	Number of cores to be used in peak picking.

Details

The mzrange and rtrange parameters are used to subset the mass and retention times considered in the analysis, reducing possible alignment problems at the extremes.
The error function calculates the expected m/z tolerance for feature annotation based on the mz and I values of each feature. To have a more complete description of the process refer to the help of AnnotateTable and the literature reference. An example is provided as well. Note that the use of "lm" is only one of the possible choices, but all kind of functional approximations working with the predict function could be used. If the error function is not provided the mass tolerance will be fixed to the value defined in the settings list.

Value

A list with three elements:

PeakTable	data.frame containing annotation information. Every line is a feature. The first columns are used to give information about these features, annotation, CAMERA, Chemspider IDs, etcetera. The last of these meta-information columns is always the one giving the retention time: "rt". After that, columns correspond to input files, and give measures of intensities for every single one of the features.
Annoation	The complete output of the AnnotateTable function.
Settings	The settings used in the pipeline.
xset	optionally, the xcmsSet/CAMERA object is returned, which can be useful for more detailed inspection of the results.
sessionInfo	The output of sessionInfo() to keep track of the sw version used for the processing

Author(s)

Pietro Franceschi

References

N. Shahaf, P. Franceschi, P. Arapitsas, I. Rogachev, U. Vrhovsek and R. Wehrens: "Constructing a mass measurement error surface to improve automatic annotations in liquid chromatography/mass spectrometry based metabolomics". Rapid Communications in Mass Spectrometry, 27(21), 2425 (2013).

Examples

```
data(LCresults)
names(LCresults)

## Not run:
## load the settings for the analysis
data(FEMsettings)

## load the annotation DB
data(LCDBtest)

## load the Synapt Q-TOF error function
data(errf)

results.xset  <- runLC(xset = LCresults$xset, settings = Synapt.RP,
                       DB = LCDBtest$DB)

## to start directly from the CDF files and include peak picking in the
## pipeline, simply provide the "files" argument rather than the "xset" argument

if (require(metaMSdata)) {
  ## get the path
  cdfpath <- system.file("extdata", package = "metaMSdata")

  ## files
  files <- list.files(cdfpath, "_RP_", full.names=TRUE)

  ## <-------------    Use the Synapt Q-TOF error function     -------------- >
  result.adaptive <- runLC(files, settings = Synapt.RP,
                           DB = LCDBtest$DB, errf = errf)

  ## <--------    Run the analysis with a fixed mass tolerance      --------- >
  result.fixed <- runLC(files, settings = Synapt.RP,
                        DB = LCDBtest$DB)
}

## End(Not run)
```

---

threeStdsDB *Information on three chemical standards measured in GC-MS (liquid injection)*

Description

User information needed to build up an in-house database, the external database for cross-checking mass spectra, and the final database obtained with `createSTDdbGC` for three chemical standards.

Usage

```
data(threeStdsDB)
data(threeStdsInfo)
data(threeStdsNIST)
```

Details

Raw GC-MS data for the three standards, linalool, methyl salicylate and ethyl hexanoate, are available in package metaMSdata. Manual information required to build up an in-house database should be presented in the form of a data.frame, an example of which is `stdInfo`. Presenting this information to `createSTDdbGC` leads to processing of the raw data, and cross-checking with an external database containing mass spectra. An excerpt of the NIST database, containing only spectra for these three compounds, is available in `smallDB`. The final database that is then obtained can be inspected in object `DB`, which is a simple list of tags and values. For use as a reference database, several of these fields are mandatory. Currently these are `Name`, `monoMW`, `pspectrum` and `std.rt`.

Source

Georg Weingart

Examples

```
data(threeStdsNIST)
length(smallDB)

data(threeStdsInfo)
stdInfo

data(threeStdsDB)
par(mfrow = c(3,1))
sapply(DB, function(x) plotPseudoSpectrum(x$pspectrum, main = x$Name))
```

---

treat.DB *Scaling of pseudospectra in an msp object*

Description

This function transforms the "raw" data in an msp DB object into preprocessed data. Even if no scale and no mass.weight is applied, the intensities are still changed: scaled to unit length.

Usage

```
treat.DB(DB, scale.p = c("sqrt", "none"), mass.weight = TRUE, isMSP = TRUE)
```

Arguments

DB	A database of spectra in the original intensity units.
scale.p	scale intensities with a square-root function, or leave them as they are. Default is to use scaling.
mass.weight	Logical: if TRUE, higher masses are given more weight.
isMSP	Logical: if TRUE, then the spectra are stored in slot pspectrum; otherwise the spectra are simply the list elements of DB.

Value

The function returns the database, where intensities are scaled.

Author(s)

Ron Wehrens

Examples

```
data(threeStdsNIST)  ## provides object smallDB, excerpt from NIST DB
smallDB.scaled <- treat.DB(smallDB)
```

Index

∗ **classes**
metaMSsettings-class, 29
∗ **datasets**
errf, 14
exptable, 15
FEMsettings, 17
GCresults, 20
LCDBtest, 24
LCresults, 25
threeStdsDB, 45
∗ **manip**
addRI, 4
alignmentLC, 5
AnnotateTable, 6
annotations2tab, 8
constructExpPseudoSpectra, 9
createSTDdbGC, 10
createSTDdbLC, 12
getAnnotationLC, 20
getAnnotationMat, 21
getFeatureInfo, 22
getPeakTable, 23
match2ExtDB, 25
matchExpSpec, 26
matchSamples2DB, 27
matchSamples2Samples, 28
msp, 33
peakDetection, 35
plotPseudoSpectrum, 36
processStandards, 37
readStdInfo, 38
removeDoubleMasses, 39
runCAMERA, 40
runGC, 41
runLC, 43
treat.DB, 46
∗ **methods**
metaSetting-methods, 32
∗ **package**
metaMS-package, 3
∗ **print**
printString, 37

addRI, 4, *42*
alignmentLC, 5
annotate, *17*, *41*
AnnotateFeature *(*AnnotateTable*)*, 6
AnnotateTable, 6, *21*
annotations2tab, 8

construct.msp *(*msp*)*, 33
constructExpPseudoSpectra, 9
createSTDdbGC, 10
createSTDdbLC, 12, *16*, *24*

DB *(*threeStdsDB*)*, 45

errf, 14
exptable, 15, *24*

FEMsettings, *6*, 17, *21*, *24*, *31*, *32*
filter.msp *(*msp*)*, 33
findPeaks, *17*, *29*

GCresults, 20
generateStdDBGC, *11*
generateStdDBGC *(*createSTDdbGC*)*, 10
generateStdDBLC *(*createSTDdbLC*)*, 12
getAnnotationLC, 20
getAnnotationMat, 21, *42*
getFeatureInfo, 22
getPeakTable, 23
groupFWHM, *41*

LCDBtest, *21*, 24
LCresults, 25
LCxset *(*LCresults*)*, 25

makeAnnotation *(*annotations2tab*)*, 8

match.unannot.patterns
(matchSamples2Samples), 28
match2ExtDB, 25
matchExpSpec, 26
matchSamples2DB, *9*, 27, *42*
matchSamples2Samples, *9*, 28, *42*
metaMS (metaMS-package), 3
metaMS-package, 3
metaMSsettings (metaMSsettings-class), 29
metaMSsettings-class, 29
metaSetting (metaSetting-methods), 32
metaSetting,metaMSsettings-method
(metaSetting-methods), 32
metaSetting-methods, 32
metaSetting<- (metaSetting-methods), 32
metaSetting<-,metaMSsettings-method
(metaSetting-methods), 32
msp, 33, *42*

Orbitrap.RP (FEMsettings), 17

peakDetection, 35, *42*
plotPseudoSpectrum, 36
printInfo (printString), 37
printString, 37
printWarning (printString), 37
processStandards, *11*, 37

read.msp (msp), 33
readStdInfo, 38
relInt (getAnnotationMat), 21
removeDoubleMasses, 39
runCAMERA, *38*, 40, *42*
runGC, *20*, 41
runLC, 43

SearchNIST (msp), 33
show,metaMSsettings-method
(metaMSsettings-class), 29
smallDB (threeStdsDB), 45
stdInfo (threeStdsDB), 45
Synapt.NP (FEMsettings), 17
Synapt.RP (FEMsettings), 17

threeStdsDB, 45
threeStdsInfo (threeStdsDB), 45
threeStdsNIST (threeStdsDB), 45
to.msp (msp), 33

treat.DB, *42*, 46
TSQXLS.GC (FEMsettings), 17

write.msp (msp), 33

xcmsSet, *38*
xset2msp (msp), 33