Package 'TargetSearch'

June 16, 2019

Type Package

Title A package for the analysis of GC-MS metabolite profiling data

Version 1.41.1

Date 2019-05-08

Author Alvaro Cuadros-Inostroza <acuadros+bioc@gmail.com>,
Jan Lisec,
Henning Redestig,
Matt Hannah

Maintainer Alvaro Cuadros-Inostroza <acuadros+bioc@gmail.com>

Depends ncdf4

Imports graphics, grDevices, methods, stats, tcltk, utils

Suggests TargetSearchData, BiocStyle, knitr

VignetteBuilder knitr

Description This packages provides a targeted pre-processing method for GC-MS data.

biocViews MassSpectrometry, Preprocessing, DecisionTree,
ImmunoOncology

License GPL (>= 2)

URL https://github.com/acinostroza/TargetSearch

BugReports https://github.com/acinostroza/TargetSearch/issues

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

git_branch master

git_last_commit d687f26

git_last_commit_date 2019-06-11

Date/Publication 2019-06-15

R topics documented:

`TargetSearch-package` *A targeted approach for GC-MS data.*

Description

This packages provides a targeted method for GC-MS data analysis. The workflow includes a peak picking algorithm to convert from netcdf files to tab delimited files, retention time correction using retention time markers provided by the user, and a library search using multiple marker masses and retention time index optimisation.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

Maintainer: Alvaro Cuadros-Inostroza <inostroza@mpimp-golm.mpg.de>

`baselineCorrection` *Baseline correction algorithm*

Description

Functions for baseline correction of GC-MS chromatograms.

Usage

```
baselineCorrection(int, threshold = 0.5, alpha = 0.95, bfraction = 0.2,
       segments = 100, signalWindow = 10, method = "linear")

baseline(ncData, baseline.opts = NULL)
```

Arguments

`int`	A matrix object of spectra peak intensities to be baseline corrected, where the columns are retention times and rows mass traces.
`threshold`	A numeric value between 0 and 1. A value of one sets the baseline above the noise, 0.5 in the middle of the noise and 0 below the noise.
`alpha`	The alpha parameter of the high pass filter.
`bfraction`	The percentage of the fragments with the lowest intensities of the filtered signal that are assumed to be baseline signal.
`segments`	The number of segments in which the filtered signal is divided.
`signalWindow`	The window size (number of points) used in the signal windowing step.
`method`	The method used to approximate the baseline. `"linear"` (default) uses linear interpolation. `"spline"` fits a cubic smoothing spline (warning: really slow).
`ncData`	A list returned by the function `TargetSearch:::.open.ncdf`.
`baseline.opts`	A list with parameters to be passed to `baselineCorrection` function. For example `baseline.opts = list(threshold = 0.5, alpha = 0.95)`.

Details

The baseline correction algorithm is based on the work of Chang et al, and it works as follows. For every mass trace, i.e., rows of matrix `int`, the signal intensity is filtered by a first high pass filter: *y[i] = alpha * (y[i-1] + x[i] - x[i-1]).* The filtered signal is divided into evenly spaced segments (`segments`) and the standard deviation of each segment is calculated. A percentage (`bfraction`) of the segments with the lowest values are assumed to be baseline signal and the standard deviation (*stdn*) of the points within those segments is calculated.

Once *stdn* has been determined, the points with absolute filtered values larger than *2 * stdn* are considered signal. After that, the signal windowing step takes every one of the points found to be signal as the center of a signal window (`signalWindow`) and marks the points within that window as signal. The remaining points are now considered to be noise.

The baseline signal is obtained by either using linear interpolation (default) or fitting a cubic smoothing spline taking only the noise. The baseline can be shifted up or down by using the parameter (threshold), which is done by the formula: *B' = B + 2*(threshold - 0.5)*2*stdn*, where *B* is the fitted spline, *stdn* the standard deviation of the noise, and `threshold` a value between 0 and 1. Finally, the corrected signal is calculated by subtracting *B'* to the original signal.

The baseline function is called by the function NetCDFPeakFinding before the peak picking algorithm is performed. Since it is an internal function, it is not intended to be executed directly, but it is exported for debugging purposes.

Value

A matrix of the same dimensions of int with the baseline corrected intensities.

Author(s)

Alvaro Cuadros-Inostroza

References

David Chang, Cory D. Banack and Sirish L. Shah, Robust baseline correction algorithm for signal dense NMR spectra. *Journal of Magnetic Resonance 187 (2007) 288-292*

See Also

RIcorrect

---

checkRimLim *Visually check retention index marker limits*

Description

A function to visually check if the retention time limites of the retention index markers (aka FAMEs) is correct.

Usage

```
checkRimLim(samples, rim, layout, show = TRUE, extend = 0.5,
  rect.col = "#e7e7e7", mar = c(2, 2, 2, 2), oma = c(3, 3, 2, 0.5),
  cex.main = 1, type = "l", ...)
```

Arguments

samples	A tsSample object created by ImportSamples.
rim	A tsRim object describing the retention index markers. See ImportFameSettings.
layout	A vector of the form c(nr,nc) to arrange the panel by nr rows and nc columns. If missing then the layout is created automatically.
show	Logical. If FALSE the plot is not shown, but the data points can be used for further inspection or for custom plots.
extend	a numeric coefficient to extend the time window search of the respective time marker. Defaults to 0.5.
rect.col	the color for the background rectangle which indicates the current retention time limits.
mar	the subplots margins, passed to par().
oma	the outer plot margins, passed to par().
cex.main	The magnification to be used for main titles, passed to par().

type	A character vector indicating the type of plots. Default "l" for lines. Passed to plot().
...	extra plotting arguments passed to plot() such as col, lty, pch, lwd.

Details

The function takes a random CDF file from your tsSample object and creates a panel plot of the m/z traces around the area in which a marker is expected to be. Repeated calls to this function can be used to check other samples. It is also possible to check a specific sample by indexing the tsSample object.

Value

A list of n times 2 matrices or invisible. Each element correspond to a marker. Columns are retention time and intensities of the respective marker's m/z. The rows can be as many data points are within the search window.

See Also

tsSample, tsRim, ImportFameSettings

Examples

```
require(TargetSearchData)

# get the cdf path TargetSearchData
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# import samples (see ImportSamples() for details)
samples <- ImportSamples(file.path(cdfpath, "samples.txt"), CDFpath = cdfpath)

# Import RI markers (see ImportFameSettings())
rim <- ImportFameSettings(file.path(cdfpath, "rimLimits.txt"))

# choose a sample at random and plot the m/z traces around the retention
# time window
ret <- checkRimLim(samples, rim)
```

FAMEoutliers *FAME outlier detection*

Description

A function to detect retention time marker (FAME) outliers.

Usage

```
FAMEoutliers(samples, RImatrix, pdffile = NA, startDay = NA, endDay = NA,
             threshold = 3, group.threshold = 0.05)
```

Arguments

samples	A tsSample object created by ImportSamples function.
RImatrix	A retention time matrix of the found retention time markers.
pdffile	A character string naming a PDF file where the FAMEs report will be saved.
startDay	A numeric vector with the starting days of your day groups.
endDay	A numeric vector with the ending days of your day groups.
threshold	A standard deviations cutoff to detect outliers.
group.threshold	A numeric cutoff to detect day groups based on hierarchical clustering. Must be between 0..1.

Details

If no pdffile argument is given, the report will be saved on a file called "TargetSearch-YYYY-MM-DD.FAME-report.pd where YYYY-MM-DD is a date.

If both startDay and endDay are not given, the function will try to detect day groups using a hierarchical clustering approach by cutting the tree using group.threshold as cutoff height.

Retention time markers that deviate more than threshold standard deviations from the mean of their day group will be identified as outliers.

Value

A logical matrix of the same size of RImatrix. A TRUE value indicates that the retention time marker in that particular sample is an outlier.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

RIcorrect, ImportSamples, TSExample

Examples

```
# load pre-calculated example data and objects
data(TSExample)

# find the retention marker outliers of the example data and save it in "outlier.pdf"
outliers <- FAMEoutliers(sampleDescription, RImatrix, pdffile = "outlier.pdf")

# find the outliers (although they are reported in the output PDF file)
apply(outliers, 1, which)
```

FindAllPeaks *Extract peaks from chromatogram files - low level function*

Description

This function extracts all peaks of a given metabolite in a given RI window. It is intented to fine-tune metabolite search parameters.

Usage

```
FindAllPeaks(samples, Lib, libID, dev=NULL, mz=NULL, RI=NULL,
         mz_type = c('selMass', 'quantMass', 'topMass'),
         columns = c("SPECTRUM", "RETENTION_TIME_INDEX", "RETENTION_TIME"))
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function.
libID	An index (integer or character) value representing the respective metabolite in the reference library Lib.
dev	The allowed retention index (RI) deviation or NULL.
mz	A list of m/z values to search or NULL.
RI	The expected retention index or NULL.
mz_type	whether to search for the selective, quant or top masses of the respective metabolite.
columns	A numeric vector with the positions of the columns SPECTRUM, RETENTION_TIME_INDEX, and RETENTION_TIME or a character vector with the header names of those columns.

Details

The function searches for all peaks of a metabolite in all samples within a RI window. The parameters dev, mz, and RI have preference over the settings of the metabolite indexed by libID. In fact, if all of these parameters are not NULL, then refLib and libID are not used.

The columns parameter is only needed for custom text RI files. There is no need to change it.

Value

It returns a matrix in which each row represent a hit. Note that there can be zero rows if no hits are found. The columns are named and these are:

Int	Peak intensity
RI	Retention Index
RI	Retention Time
mz	the searched m/z value
fid	the respective file or sample index. An integer value.

Note

This is an internal function not intended to be invoked directly, but it is exposed for convenience and advanced used.

In the future it will replace FindPeaks.

Author(s)

Alvaro Cuadros-Inostroza

See Also

FindPeaks

Examples

```
# load pre-calculated example data files and objects
require(TargetSearchData)
data(TSExample)

# get and set the RI file path
RIpath(sampleDescription) <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# search all peaks of Valine (GC.3) and selective masses
peaks <- FindAllPeaks(sampleDescription, refLibrary, 'GC.3')
head(peaks)

# a numeric index is also allowed
peaks <- FindAllPeaks(sampleDescription, refLibrary, 3)
head(peaks)

# search arbitrary masses at arbitrary RI. the reference library and ID
# must be set set to NULL.
peaks <- FindAllPeaks(sampleDescription, NULL, NULL, dev=3000, RI=270000, mz=c(144, 100))
head(peaks)
```

FindPeaks *Extract peaks from chromatogram files*

Description

This function extracts the maximum intensity of a list of masses in a given RI window.

Usage

```
FindPeaks(my.files, refLib,
    columns = c("SPECTRUM", "RETENTION_TIME_INDEX", "RETENTION_TIME"),
    showProgressBar = FALSE)
```

Arguments

my.files	A character vector naming RI files to be searched.
refLib	A numeric matrix with three columns or a list of three column matrices. The second column contains the masses and the first and third column contains the RI limits.
columns	A numeric vector with the positions of the columns SPECTRUM, RETENTION_TIME_INDEX, and RETENTION_TIME or a character vector with the header names of those columns.
showProgressBar	Logical. Should the progress bar be displayed?

Details

The reference library parameter refLib can be either a single three-column matrix or a list of such matrices. If it is a list, the length must match the length of my.files. In this case, every component will be used to iteratively search in the corresponding file.

The RI files format can be either "text" or "binary". The type is detected dinamically.

Value

A tsMSdata object.

Note

This is an internal function not intended to be invoked directly.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

medianRILib, sampleRI, peakFind, tsMSdata

Examples

```
# load example CDF files
require(TargetSearchData)
# load pre-calculated example data and objects
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path

my.files <- RIfiles(sampleDescription)
# make a three column matrix: lower RI, mass, upper RI
refLib   <- refLib(refLibrary)
head(refLib)

# extract the peaks
peaks <- FindPeaks(my.files, refLib)
```

fixRI *Fixing Retention Time Index Correction*

Description

This function can be used to correct the detected retention time index (RI) markers or to manually force their location to specific retention times if, for example, the RI markers were not co-injected with the biological samples.

Usage

```
fixRI(samples, rimLimits, RImatrix=NULL, sampleNames=NULL)
```

Arguments

samples	A tsSample object created by ImportSamples function.
rimLimits	A tsRim object. See ImportFameSettings.
RImatrix	Optional. A retention time matrix of the found retention time markers that was obtained after running RIcorrect.
sampleNames	Optional. A character vector naming the samples that are to be RI corrected.

Details

Sometimes the retention index limits (see ImportFameSettings) are not set correctly and you will have to run the peak detection and RI correction function (RIcorrect) again, which may take a long time specially if there are many samples.

Instead, a simple approach is to fix the RI limits and use this function to correct the generated RI files. Since these files are much smaller than CDF files (chromatograms), this runs much faster.

Other possibility is that the time positions of one or more RI markers were wrongly detected because there was just simply no peak or the RI markers where not co-injected in some samples. You could manually force the locations of the RI markers. This case is discussed in the "RICorrection" vignette.

The behavior of this function depends on whether the parameter RImatrix is NULL or not. If NULL, the RI markers will be searched again (using the settings of rimLimits parameters, which you should have already fixed) and the resulting values will be used to correct the RI files. If it is a numeric matrix, then these values will be used to correct the RI files. Note that this matrix dimmensions are exactly *m* samples (rows) times *n* (columns) RI markers.

sampleNames controls which samples will be corrected. If NULL then all samples will be corrected. It could be character vector (sample names) or a numeric vector representing the sample indexes.

Value

Invisible NULL. It prints the corrected samples.

Author(s)

Alvaro Cuadros-Inostroza

See Also

RIcorrect, FAMEoutliers,ImportSamples, ImportFameSettings

Examples

```
require(TargetSearchData)
# import refLibrary, rimLimits and sampleDescription.
data(TSExample)
# get the CDF files
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# select a subset of samples
smp <- sampleDescription[1:4]

# update the CDF path
CDFpath(smp) <- cdfpath

# make a copy of the RI markers object
rim <- rimLimits

# mess up the limits of marker 3 (real value is 369 seconds app.)
rimLimits(rim)[3,] <- c(375, 400)

# run RIcorrect
RImat <- RIcorrect(smp, rim, massRange = c(85,320),
           Window = 15, pp.method = "ppc", IntThreshold = 50)

# fix the limits of marker 3
rimLimits(rim)[3,] <- c(360, 400)

# you could run again RIcorrect, but this is faster
fixRI(smp, rim)

# get the RI matrix
RImat <- riMatrix(smp, rim)

# compare the values with the real ones (previously stored in RImatrix)
stopifnot( all.equal(RImat, RImatrix[,1:4], tolerance=1e-8) )

# manual adjusment or RI markers for sample 3.
# Warning: this is just an example to illustrate how to use this function.
#          don't do this unless you know what you're doing.
RImat[,3] <- c(252, 311, 369)
fixRI(smp, rim, RImat, 3)
```

ImportFameSettings *Retention time markers settings*

Description

This function imports a list of retention standard markers.

Usage

```
ImportFameSettings(tmp.file = NA, mass = NA, ...)
```

Arguments

tmp.file	A character string naming a file with standard markers.
mass	The m/z standard marker.
...	Other options passed to read.delim function.

Details

The standard marker file is a tab-delimited text file with 3 or 4 columns. Column names doesn't matter. They must be in the following order.

- LowerLimit - The Retention time lower limit in seconds.
- UpperLimit - The Retention time upper limit in seconds.
- RIstandard - The RI value of that standard.
- mass - The m/z standard marker. This column is optional since it could be set by the mass parameter.

If no arguments are given, a default object will be returned.

Value

A tsRim object.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

RIcorrect, tsRim

Examples

```
# get the RI marker definition file
cdfpath  <- file.path(find.package("TargetSearchData"), "gc-ms-data")
rim.file <- file.path(cdfpath, "rimLimits.txt")

# set the mass marker to 87
mass <- 87

# load the definition
rimLimits <- ImportFameSettings(rim.file, mass = mass)

# sometimes you need to change the limits of a particular standard
rimLimits(rimLimits)[2,] <- c(410, 450)

# to change the mass value
rimMass(rimLimits) <- 85
```

`ImportLibrary` *Library import*

Description

These functions import a metabolite library file that will be used to processed the GC-MS data. Two file formats are supported: a tab-delimited format and the more common NIST MSP format.

Usage

```
ImportLibrary(x, type = "auto", ...)

ImportLibrary.tab(libfile, fields = NULL, RI_dev = c(2000,1000,200),
    SelMasses = 5, TopMasses = 15, ExcludeMasses = NULL,
    libdata, file.opt=NULL)

ImportLibrary.msp(libfile, fields = NULL, RI_dev = c(2000,1000,200),
    SelMasses = 5, TopMasses = 15, ExcludeMasses = NULL)
```

Arguments

`x`	A character string or a `data.frame`. If `data.frame`, it will be passed to `ImportLibrary.tab` as parameter `libdata`. If `character`, it will be passed as `libfile` to either `ImportLibrary.tab` or `ImportLibrary.msp` according to the file type (option `type`).
`libfile`	A character string naming a library file. See details.
`type`	The library file format. Posible options are `"tab"` for a tab-delimited file, `"msp"` for NIST MSP format, or `"auto"` for autodetection. Default to `"auto"`.
`fields`	A two component list. Each component contains a regular expression used to parse and extract the fields for retention index and selection masses. Only meaningful for MSP format.
`RI_dev`	A three component vector with RI windows.
`SelMasses`	The number of selective masses that will be used.
`TopMasses`	The number of most intensive masses that will be taken from the spectrum, if no `TOP_MASS` is provided.
`ExcludeMasses`	Optional. A vector containing a list of masses that will be excluded.
`libdata`	Optional. A data frame with library data. The format is the same as the library file. It is equivalent to importing the library file first with `read.table` and calling `ImportLibrary.tab` after. This might be preferable for "fine tuning", for example, if the library file is in CSV format instead of tab-delimited.
`file.opt`	Optional. A list containing arguments to be passed to `read.table`.
`...`	Further arguments passed to `ImportLibrary.tab` or `ImportLibrary.msp`

Details

ImportLibrary is a wrapper for functions ImportLibrary.tab and ImportLibrary.msp which detects automatically which function should be called.

ImportLibrary.tab reads a tab delimited text file by calling the function read.table which will be parsed and converted to a tsLib object. The following arguments are used by default (which are not exactly the defaults for read.table):

header=TRUE,sep="\t",quote="",dec=".",fill=TRUE,comment.char="#"

The argument file.opt can be used to change these options. Other alternative is to import first the file with read.table and friends, and call ImportLibrary with the resulting data.frame. This allows more flexibility with libraries with unusual characters, for example.

These columns are needed:

- libID - A unique identifier for the metabolite.
- Name - The metabolite name.
- RI - The expected RI.
- SEL_MASS - A list of selective masses separated with semicolon.
- TOP_MASS - A list of the most abundant masses to be searched, separated with semicolons.
- Win_k - The RI windows, k = 1,2,3. Mass search is perfomed in three steps. A RI window required for each one of them.
- SPECTRUM - The metabolite spectrum. m/z and intensity are separated by spaces and colons.
- QUANT_MASS - A list of masses that might be used for quantification. One value per metabolite and it must be one of the selective masses. (optional)

The columns Name and RI are mandatory. At least one of columns SEL_MASS, TOP_MASS and SPECTRUM must be given as well. By using the parameters SelMasses or TopMasses it is possible to set the selective masses or the top masses from the spectra. The parameter ExcludeMasses is used only when masses are obtained from the spectra. The parameter RI_dev can be used to set the RI windows. Note that in this case, all metabolites would have the same RI windows.

A unique identifier may be provided as well (libID). This could be a external database identifier. If it is not provided, a random identifier will be created of the form GC.k, k = 1,...,N.

The MSP format is a text file that can be imported/exported from NIST. A typical MSP file looks like this:

```
Name: Pyruvic Acid
Synon: Propanoic acid, 2-(methoxyimino)-, trimethylsilyl ester
Synon: RI: 223090
Synon: SEL MASS: 89|115|158|174|189
Formula: C7H15NO3Si
MW: 189
Num Peaks: 41
  85    8;  86   13;  87    5;  88    4;  89  649;
  90   55;  91   28;  92    1;  98   13;  99  257;
 100  169; 101   30; 102    7; 103   13; 104    1;
 113    3; 114   35; 115  358; 116   44; 117   73;
 118   10; 119    4; 128    2; 129    1; 130   10;
 131    3; 142    1; 143   19; 144    4; 145    1;
 157    1; 158   69; 159   22; 160    4; 173    1;
 174  999; 175  115; 176   40; 177    2; 189   16;
```

```
 190    2;

Name: another metabolite
...
```

Different entries must be separated by empty lines. In order to parse the retention time index (RI) and selective masses (SEL MASS), a two component list containing the field names of RI and SEL_MASS must be provided by using the parameter `fields`. In this example, use `field = list("RI: ","SEL MASS: ")`. Note that `ImportLibrary` expects to find those fields next to "Synon:". Alternatively, you could provide the RI and SEL_MASS using the `tsLib` methods.

Libraries for TargetSearch and for different retention index systems, such as VAR5 or MDN35, can be downloaded from http://gmd.mpimp-golm.mpg.de/.

Value

A `tsLib` object.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportSamples`, `tsLib`

Examples

```
# get the reference library file
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
lib.file  <- file.path(cdfpath, "library.txt")

# Import the reference library
refLibrary <- ImportLibrary(lib.file)

# set new names for the first 3 metabolites
libName(refLibrary)[1:3] <- c("Metab01", "Metab02", "Metab03")

# change the retention time deviations of Metabolite 3
RIdev(refLibrary)[3,] <- c(3000,1500,150)
```

ImportSamples *Sample definitions*

Description

This function imports a sample list that will be processed from a tab delimited file.

Usage

```
ImportSamples(sampfile, CDFpath=".", RIpath=".", ftype=c("binary", "text"), ...)

ImportSamplesFromDir(CDFpath=".", RIfiles=FALSE, ignore.case=TRUE,
ftype=c("binary", "text"))
```

Arguments

sampfile	A character string naming a sample file or a data.frame See details.
CDFpath	A character string naming a directory where the CDF files are located.
RIpath	A character string naming a directory where the RI corrected text files are/will be located.
RIfiles	Logical. If TRUE, the function will look for for RI files (RI_*) instead of CDF files (the default).
ftype	A character string giving the file type for RI files. Options are "binary" and "text".
ignore.case	Logical. Should pattern-matching be case-insensitive?
...	Other options passed to read.delim function.

Details

The sample file is a tab-delimited text file or, alternatively, a data.frame like the one would get by calling read.delim. The following two columns are expected:

- CDF_FILE - The list of baseline corrected CDF files.
- MEASUREMENT_DAY - The day when the sample was measured.

The function first looks for columns matching those names. If they are not found, then it looks for columns with the substrings 'cdf' and 'day' respectively. If the 'cdf' column is not found, the function raises an error, but if 'day' is not found, then it assumes the measurement day is the same for all. In case of multiple matches, then it takes the first column. The column order does not matter. Other columns could be included in that file. They won't be used by the script, but will be included in the "sample" R object.

The ftype parameter sets the file format of the RI files, which are created by the function RIcorrect. "text" is the *classic* text format and "binary" is a binary version, designed for speed, of the text format (TargetSearch >= 1.12.0). The file format can be identified by the file extension (".txt" for text, ".dat" for binary), but this is just an indicator: the file format is detected dinamically during file reading. Use the method fileFormat to set or get this parameter.

Value

A tsSample object.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

ImportLibrary, tsSample

Examples

```
# get the sample definition definition file
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
sample.file  <- file.path(cdfpath, "samples.txt")

# set a path where the RI files will be created
```

```
RIpath <- "."

# import samples
sampleDescription <- ImportSamples(sample.file, CDFpath = cdfpath, RIpath = RIpath)

# change the sample names. It is required that the names must be unique.
sampleNames(sampleDescription) <- paste("Sample", 1:length(sampleDescription), sep = "_")

# change the file paths (relative to the working path)
CDFpath(sampleDescription) <- "my_cdfs/"
RIpath(sampleDescription)  <- "my_RIs/"
```

`medianRILib` *Median RI library correction*

Description

Return a `tsLib` object with the median RI of the selective masses across samples.

Usage

```
medianRILib(samples, Lib, makeReport = FALSE, pdfFile = "medianLibRep.pdf",
columns = c("SPECTRUM", "RETENTION_TIME_INDEX", "RETENTION_TIME"),
    showProgressBar = FALSE)
```

Arguments

`samples`	A `tsSample` object created by `ImportSamples` function.
`Lib`	A `tsLib` object created by `ImportLibrary` function.
`makeReport`	Logical. If `TRUE` will report the RI deviations for every metabolite in the library.
`pdfFile`	The file name where the report will be saved.
`columns`	A numeric vector with the positions of the columns `SPECTRUM`, `RETENTION_TIME_INDEX`, and `RETENTION_TIME` or a character vector with the header names of those columns.
`showProgressBar`	Logical. Should the progress bar be displayed?

Value

A `tsLib` object. It will update the slot `med_RI` which contains the median RI of every searched metabolite.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportSamples`, `ImportLibrary`, `tsLib-class`

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path
# Import Library
refLibrary       <- ImportLibrary(file.path(RI.path,'library.txt'))
# update median RI
refLibrary       <- medianRILib(sampleDescription, refLibrary)

# perhaps you need to adjust the library RI of one metabolite and the allowed time
# deviation (first time deviation window)
libRI(refLibrary)[5] <- 306500
RIdev(refLibrary)[5,1] <- 2000

refLibrary       <- medianRILib(sampleDescription, refLibrary)
```

`NetCDFPeakFinding` *Peak picking algorithm from CDF files*

Description

This function reads a netcdf chromatogram file, finds the apex intensities and returns a list containing the retention time and the intensity matrices.

Usage

```
NetCDFPeakFinding(cdfFile, massRange = NULL, Window = 15, IntThreshold = 10,
                  pp.method = "ppc", baseline = FALSE, baseline.opts = NULL)
```

Arguments

`cdfFile`	A character string naming a netcdf file.
`massRange`	Deprecated. It is completely ignored but it is kept for compatibility with old scripts.
`Window`	The window used by peak picking method. The number of points actually used is `2*Window + 1`.
`IntThreshold`	Apex intensities lower than this value will be removed from the RI files.
`pp.method`	The pick picking method to be used. Options are `"smoothing"`, `"gaussian"` and `"ppc"`.
`baseline`	Logical. Should baseline correction be performed?
`baseline.opts`	A list of options passed to `baselineCorrection`.

Details

The function expects the following NetCDF variables: `intensity_values`, `mass_values`, `scan_index`, `point_count` and `scan_acquisition_time`. Otherwise, an error will be displayed.

Formerly, the `massRange` parameter was a numeric vector with two components: lower and higher masses. Now, the mass range is detected automatically and it has no effect if it is set. It is kept only for compatibility reasons.

There are three peak picking algorithms that can be used. The `"smoothing"` method smooths the m/z curves by a moving average and then looks for a change of sign of the intensity difference between two consecutive points. The `"gaussian"` is exactly the same, but it uses a gaussian smoother instead of a moving average.

The `"ppc"` uses a sliding window and looks for the local maxima. This method is based on R-package ppc.

Value

A three component list.

`Time`	The retention time vector.
`Peaks`	The intensity matrix. Rows are the retention times and columns are masses. The first column is the lower mass value and the last one is the higher mass.
`massRange`	The mass range.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`peakCDFextraction`

Examples

```
require(TargetSearchData)
CDFpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
CDFfiles <- dir(CDFpath, pattern = ".cdf$", full.names = TRUE)
CDFfiles

# extrac peaks of first chromatogram
peaks.1 <- NetCDFPeakFinding(CDFfiles[1], Window = 15, IntThreshold = 10, pp.method = "smoothing")
# scan acquisition times
head(peaks.1$Time)
# peaks in matrix form. first column is mass 85, last one is mass 320.
head(peaks.1$Peaks)
```

peakCDFextraction *NetCDF to R*

Description

This function reads a netcdf chromatogram file and returns a list containing the retention time and the intensity matrices.

Usage

```
peakCDFextraction(cdfFile, massRange)
```

Arguments

cdfFile	A character string naming a netcdf file.
massRange	Deprecated. The mass range is extracted automatically.

Details

The function expects the following NetCDF variables: intensity_values, mass_values, scan_index, point_count and scan_acquisition_time. Otherwise, an error will be displayed.

The massRange parameter used to be a numeric vector with two components: lower and higher masses. Now, the mass rangeis determined by scanning the CDF file. All masses in the CDF file will be extracted.

Value

A two component list.

Time	The retention time vector.
Peaks	The intensity matrix. Rows are the retention times and columns are masses. The first column is the lower mass value and the last one is the higher mass.
massRange	The mass range.

Note

This function does not look for peaks, just extracts all the raw intensity values of the chromatogram file. Use NetCDFPeakFinding instead.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

NetCDFPeakFinding

peakFind *Intensities and RI matrices*

Description

This function returns a list of the intensities and RI matrices that were searched.

Usage

```
peakFind(samples, Lib, cor_RI,
    columns = c("SPECTRUM", "RETENTION_TIME_INDEX", "RETENTION_TIME"),
    showProgressBar = FALSE)
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function with corrected RI values. See medianRILib.
cor_RI	A matrix of correlating selective masses RI for every sample. See sampleRI.
columns	A numeric vector with the positions of the columns SPECTRUM, RETENTION_TIME_INDEX, and RETENTION_TIME or a character vector with the header names of those columns.
showProgressBar	Logical. Should the progress bar be displayed?

Value

A tsMSdata object.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

ImportSamples, ImportLibrary, medianRILib, sampleRI, tsMSdata, tsLib, tsSample

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
refLibrary <- ImportLibrary(file.path(RI.path,"library.txt"))

# update RI file path
RIpath(sampleDescription) <- RI.path

peakData <- peakFind(sampleDescription, refLibrary, corRI)
# show peak Intensities.
head(Intensity(peakData), 2)
```

```
# How to get intensities for a particular metabolite
# just select the identifier. Here extract the intensities
# for the first metabolite in the library
IntMatrix <- Intensity(peakData)[[1]]
```

---

plotFAME *Plot a standard marker*

Description

Plots a given standard marker.

Usage

```
plotFAME(samples, RImatrix, whichFAME)
```

Arguments

samples	A tsSample object created by ImportSamples function.
RImatrix	A retention time matrix of the found retention time markers.
whichFAME	The retention marker to plot. Must be a number between 1 and the number of markers.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

RIcorrect, FAMEoutliers, tsSample

Examples

```
require(TargetSearchData)
data(TSExample)
# plot Retention index standards 1 to 3
plotFAME(sampleDescription, RImatrix, 1)
plotFAME(sampleDescription, RImatrix, 2)
plotFAME(sampleDescription, RImatrix, 3)
```

plotPeak *Plot peaks*

Description

Plot a peak of a given metabolite in a given sample showing the search windows.

Usage

```
plotPeak(samples, Lib, metProf, rawpeaks, which.smp=1, which.met=1,
massRange=NULL, corMass=FALSE)
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function.
metProf	A metabolite profile object. See Profile
rawpeaks	A thre component list containing the retention time, the intensity matrix, and the mass range. See peakCDFextraction.
which.smp	A numeric value indicating the sample.
which.met	A numeric value indicating the metabolite.
massRange	A two component numeric vector with the scan mass range to extract. or NULL for automatic detection.
corMass	Logical. If TRUE, show only correlating masses for the selected metabolite. Show all masses otherwise.

Value

A two component list containing the retention time and the intensity matrices. This list can be recycled as the 'rawpeaks' parameter for further plots (for example in a loop), so the CDF file doesn't need to be read again.

Note

This function was completely rewritten. For the old function, see plotPeakSimple

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

plotPeakSimple, RIcorrect, tsMSdata, tsRim, peakCDFextraction, matplot

Examples

```
require(TargetSearchData)
data(TSExample)

# update CDF and RI paths
CDFpath(sampleDescription) <- file.path(find.package("TargetSearchData"), "gc-ms-data")
RIpath(sampleDescription) <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# Plot the peak "Valine" for sample number 1
grep("Valine", libName(refLibrary)) # answer: 3

# plot peak from the cdf file. The rawpeaks object can be recycled in order to plot
# other metabolites.
rawpeaks <- plotPeak(sampleDescription, refLibrary, metabProfile, which.smp=1,
        which.met=3, massRange=c(85,500), corMass=FALSE)
```

---

plotPeakRI *Plot peak RI across samples*

Description

Plot peak RI of the quant mass of a given metabolite across samples. This function can be used to visualize the elution time of a metabolite in the RI and RT dimension, and in combination with the function sampleRI for fine tunning.

Usage

```
plotPeakRI(samples, Lib, libID, dev=NULL, mz=NULL, RI=NULL,
           method=c('RI', 'Intensity'), useRI=TRUE, main=NULL,
           col=NULL, int_range=c(2,6), cex_range=c(.7,6), key_width=2)
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function.
libID	An index (integer or character) value representing the respective metabolite in the reference library Lib.
dev	The allowed retention index (RI) deviation or NULL.
mz	A list of m/z values to search or NULL.
RI	The expected retention index or NULL.
method	A character vector used to decided what peak should be chosen in case there are ambiguous peaks. If 'RI', then the closest peak to the expected RI is chosen. If 'Intensity', then the highest is taken.
useRI	Logical. Should the RI or RT be displayed in the y-axis?
main	The title for the plot. If NULL, then the metabolite name is displayed.
col	A color vector (length > 2) to show different levels of peak intensity.
int_range	A length-two vector. The limits of the intensity for the color key. Note the the intensity is expressed in log10. Eg, a value of 2 represents a intensity of 100.
cex_range	The 'cex' range value of the points. lower-intense peaks are represented as points of smaller size.
key_width	The width in cm of the area allocated for the color key.

Details

This function uses internally FindAllPeaks, so the same rules apply, i.e., the parameters dev, mz, and RI have preference over the settings of the metabolite indexed by libID.

In the plot, the x-axis are samples as defined by the object samples. The y-axis is retention index (RI) as shown on the left-hand-size. On the right-hand-size y-axis the approximate retention time (RT) is shown. This is because the RT varies across samples, therefore it is averaged for displays purposes. If useRI==FALSE, then the RT is displayed on the left hand size and the RI is averaged and shown on the left. Note that in either case, the RI is used for searching.

The point size is proportional to the log10 of the peak intensity. Their size is controlled by the parameters int_range and cex_range. By default, intensities of 100 (log10 => 2) or lower are shown with cex=0.7, while intensities greater than 1000000 (log10 => 6) as displaued with cex=6. This also affects the scaling of the color key.

The best peaks, selected acording to method, are shown with a black border, while the other are shown with no border and slightly transparent.

The output is the RI of the best peaks or invisible. Note that if no peak is found, then no plot is drawn.

Value

Returns invisible or a numeric vector with the corresponding RI of the best peak chosen by method.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

FindAllPeaks, sampleRI, ImportSamples, ImportLibrary

Examples

```
def.par <- par(no.readonly = TRUE) # save parameters for resetting

# load pre-calculated example data files and objects
require(TargetSearchData)
data(TSExample)

# get and set the RI file path
RIpath(sampleDescription) <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# search all peaks of Valine (GC.3) and selective masses. Retention index
ri <- plotPeakRI(sampleDescription, refLibrary, 'GC.3')

# increase deviation, change m/z to search, change colors and title
main <- 'Valine'
cols <- c('red', 'blue', 'green')
ri <- plotPeakRI(sampleDescription, refLibrary, 'GC.3', dev=4000, mz=144,
                 main=main, col=cols)

# plot by RT instead. Note the RI is still returned
ri <- plotPeakRI(sampleDescription, refLibrary, 'GC.3', useRI=FALSE)

par(def.par) # reset to default
```

plotPeakSimple *Plot peaks - simple interface*

Description

Plot selected ions in a given time range.

Usage

```
plotPeakSimple(rawpeaks, time.range, masses, cdfFile = NULL, useRI = FALSE,
 rimTime = NULL, standard = NULL, massRange = NULL, ...)
```

Arguments

rawpeaks	A three component list containing the retention time, the intensity matrix, and the mass range. See peakCDFextraction.
time.range	The time range to plot in retention time or retention time index units to plot.
masses	A vector containing the ions or masses to plot.
cdfFile	The name of a CDF file. If a file name is specified, the ions will be extracted from there instead of using rawpeaks.
useRI	Logical. Whether to use Retention Time Indices or not.
rimTime	A retention time matrix of the found retention time markers. It is only used when useRI is TRUE.
standard	A numeric vector with RI values of retention time markers. It is only used when useRI is TRUE.
massRange	A two component numeric vector with the scan mass range to extract or NULL for automatic detection.
...	Further options passed to matplot.

Note

This function used to be named 'plotPeak'. This function was completely rewritten so we kept the old version and renamed it 'plotPeakSimple'.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

plotPeak, RIcorrect, tsMSdata, tsRim, peakCDFextraction, matplot

Examples

```
require(TargetSearchData)
data(TSExample)

# update CDF path
CDFpath(sampleDescription) <- file.path(find.package("TargetSearchData"), "gc-ms-data")
```

```
# Plot the peak "Valine" for sample number 1
grep("Valine", libName(refLibrary)) # answer: 3
# select the first file
cdfFile  <- CDFfiles(sampleDescription)[1]

# select "Valine" top masses
top.masses <- topMass(refLibrary)[[3]]

# plot peak from the cdf file
plotPeakSimple(cdfFile = cdfFile, time.range = libRI(refLibrary)[3] + c(-2000,2000),
    masses = top.masses, useRI = TRUE, rimTime = RImatrix[,1],
    standard = rimStandard(rimLimits), massRange = c(85, 500))

# the same, but extracting the peaks into a list first. This may be better if
# you intend to loop through several peaks.
rawpeaks <- peakCDFextraction(cdfFile, massRange = c(85,500))
plotPeakSimple(rawpeaks, time.range = libRI(refLibrary)[3] + c(-2000,2000),
    masses = top.masses, useRI = TRUE, rimTime = RImatrix[,1],
    standard = rimStandard(rimLimits), massRange = c(85, 500))
```

`plotRIdev` *Plot Retention Time Index Deviation*

Description

`plotRIdev` plots the Retention Time Index Deviation of a given set of metabolites. `plotAllRIdev` saves the plots of the RI deviations of all the metabolites in the library object into a PDF file.

Usage

```
plotRIdev(Lib, peaks, libId = 1)

plotAllRIdev(Lib, peaks, pdfFile, width = 8, height = 8, ...)
```

Arguments

`Lib`	A `tsLib` object created by `ImportLibrary` function.
`peaks`	A `tsMSdata` object. See `peakFind`.
`libId`	A numeric vector providing the indices of the metabolites to plot.
`pdfFile`	A file name where the plot will be saved. Only `plotAllRIdev`.
`width, height`	The width and height of the plots in inches. Only `plotAllRIdev`.
`...`	Further options passed to `pdf`.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportLibrary`, `tsLib`, `tsMSdata`, `pdf`

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path

peakData <- peakFind(sampleDescription, refLibrary, corRI)

# Plot RI deviation of metabolite "Valine"
grep("Valine", libName(refLibrary)) # answer: 3
plotRIdev(refLibrary, peakData, libId = 3)

# Plot an RI deviation overview of the first nine metabolites
plotRIdev(refLibrary, peakData, libId = 1:9)

# Save all RI deviation into a pdf file
plotAllRIdev(refLibrary, peakData, pdfFile = "RIdeviations.pdf")
```

---

plotSpectra *Plot a Spectra Comparison*

Description

`plotSpectra` plots a contrast between the reference spectra and the median spectra of a given metabolite in the library. `plotAllRIdev` saves the plots of the median-reference spectra comparisons of all the metabolites in the reference library into a PDF file.

Usage

```
plotSpectra(Lib, peaks, libId = 1, type = "ht")

plotAllSpectra(Lib, peaks, type = "ht", pdfFile, width = 8, height = 8, ...)
```

Arguments

`Lib`	A `tsLib` object created by `ImportLibrary` function.
`peaks`	A `tsMSdata` object. See `peakFind`.
`libId`	A numeric vector providing the indices of the metabolites to plot.
`type`	The type of the plot. Options are `"ht"`, head-tail plot, `"ss"`, side by side plot, and `"diff"`, spectrum difference plot.
`pdfFile`	A file name where the plot will be saved. Only `plotAllRIdev`.
`width, height`	The width and height of the plots in inches. Only `plotAllRIdev`.
`...`	Further options passed to `pdf`.

Details

The median spectra is obtained by computing the median intensity of every ion across the samples. The median and the reference spectra values are scaled to vary between 0 and 999 in order to make them comparable.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

tsLib, tsMSdata,pdf

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path

peakData <- peakFind(sampleDescription, refLibrary, corRI)

# Plot a comparison RI deviation of metabolite "Valine"
grep("Valine", libName(refLibrary)) # answer: 3
plotSpectra(refLibrary, peakData, libId = 3, type = "ht")

# Plot the spectra "side by side"
plotSpectra(refLibrary, peakData, libId = 3, type = "ss")

# Plot the spectra difference
plotSpectra(refLibrary, peakData, libId = 3, type = "diff")
```

Profile *Average the correlating masses for each metabolite*

Description

This function makes a profile from the masses that correlate for each metabolite.

Usage

```
Profile(samples, Lib, peakData, r_thres = 0.95, method = "dayNorm", minPairObs = 5)
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function with corrected RI values. See medianRILib.
peakData	A tsMSdata object. See peakFind.

r_thres	A correlation threshold.
method	Normalisation method. Options are "dayNorm", a day based median normalisation, "medianNorm", normalisation using the median of all the intensities of a given mass, and "none", no normalisation at all.
minPairObs	Minimum number of pair observations. Correlations between two variables are computed using all complete pairs of observations in those variables. If the number of observations is too small, you may get high correlations values just by chance, so this parameters is used to avoid that. Cannot be set lower than 5.

Value

A tsProfile object. The slots are:

Info	A data frame with a profile of all masses that correlate.
Intensity	A list containing peak-intensity matrices, one matrix per metabolite.
RI	A list containing RI matrices, one matrix per metabolite.
profInt	A matrix with the averaged intensities of the correlating masses.
profRI	A matrix with the averaged RI of the correlating masses.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

ImportSamples, ImportLibrary, medianRILib, peakFind, tsProfile

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path
# Import Library
refLibrary        <- ImportLibrary(file.path(RI.path,'library.txt'))
# update median RI
refLibrary        <- medianRILib(sampleDescription, refLibrary)
# get the sample RI
corRI             <- sampleRI(sampleDescription, refLibrary, r_thres = 0.95)
# obtain the peak Intensities of all the masses in the library
peakData          <- peakFind(sampleDescription, refLibrary, corRI)
# make a profile of the metabolite data
metabProfile      <- Profile(sampleDescription, refLibrary, peakData, r_thres = 0.95)

# same as above, but with different thresholds.
metabProfile      <- Profile(sampleDescription, refLibrary, peakData,
                     r_thres = 0.9, minPairObs = 5)
```

`ProfileCleanUp` *Reduce redundancy of the profile*

Description

This function reduces/removes redundancy in a profile.

Usage

```
ProfileCleanUp(Profile, timeSplit=500, r_thres=0.95, minPairObs=5,
    prioritization=c('mass','score'), corMass=1, score=0,
    show=c('unidentified','knowns','full'))
```

Arguments

`Profile`	A `tsProfile` object. See `Profile`.
`timeSplit`	A RI window.
`r_thres`	A correlation threshold.
`minPairObs`	Minimum number of pair observations. Correlations between two variables are computed using all complete pairs of observations in those variables. If the number of observations is too small, you may get high correlations values just by chance, so this parameters is used to avoid that. Cannot be set lower than 5.
`prioritization`	Selects whether the metabolite suggestion should be based on the number of correlation masses (`mass`) or the score (`score`).
`corMass`	Metabolites with a number of correlation masses lower than `score` will be marked as 'Unidentified RI'
`score`	Metabolites with a score lower than `score` will be marked as unidentified.
`show`	A character vector. If `unidentified`, all non-redundant metabolites will be returned; if `knowns`, only returns those metabolites with correlation masses and score greater than the given values; and if `full`, it shows all redundant metabolites, which may be useful to retrieve the data from misidentified metabolites.

Details

Metabolites that are inside a `timeSplit` window will be correlated to see whether the metabolites are potentially the same or not, by using `r_thres` as a cutoff. If so, the best candidate will be chosen according to the value of `prioritization`: If 'mass', then metabolites will be suggested based on number of correlating masses, and if 'score', then the score will be used. Metabolites that don't have al least `corMass` correlating masses and `score` score will be marked as 'unidentified' and not will be suggested, unless all the metabolites in group are unidentified.

For example, suppose that three metabolites A (CM=3, S=900), B (CM=6, S=700), C (CM=5, S=800) correlate within the same time group, where CM is the number of correlating masses and S is the score.

- If prioritization='mass', corMass=3, score=650, then the suggested order is B, C, A.
- If prioritization='mass', corMass=3, score=750, then the suggested order is C, A, B.
- If prioritization='mass', corMass=3, score=850, then the suggested order is A, B, C.
- If prioritization='score', corMass=3, score=650, then the suggested order is A, C, B.

- If prioritization='score', corMass=4, score=650, then the suggested order is C, B, A.
- If prioritization='score', corMass=4, score=850, then the suggested order is C, A, B.

Note that by chosing `prioritization='mass'`, `score=0`, and `corMass=1` you will get the former behavior (`TargetSearch <= 1.6`).

Value

A `tsProfile` object with a non-redundant profile of the masses that were searched and correlated, and intensity and RI matrices of the correlating masses.

`slot "Info"`	A data frame with a profile of all masses that correlate and the metabolites that correlate in a `timeSplit` window.
`slot "profInt"`	A matrix with the averaged intensities of the correlating masses.
`slot "profRI"`	A matrix with the averaged RI of the correlating masses.
`slot "Intensity"`	A list containing peak-intensity matrices, one matrix per metabolite.
`slot "RI"`	A list containing RI matrices, one matrix per metabolite.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

Profile, tsProfile

Examples

```
# load example data
require(TargetSearchData)
data(TSExample)

RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
refLibrary <- ImportLibrary(file.path(RI.path,"library.txt"))
# update RI file path
RIpath(sampleDescription) <- RI.path
# Import Library
refLibrary       <- ImportLibrary(file.path(RI.path,'library.txt'))
# update median RI
refLibrary       <- medianRILib(sampleDescription, refLibrary)
# get the sample RI
corRI            <- sampleRI(sampleDescription, refLibrary, r_thres = 0.95)
# obtain the peak Intensities of all the masses in the library
peakData         <- peakFind(sampleDescription, refLibrary, corRI)
metabProfile     <- Profile(sampleDescription, refLibrary, peakData, r_thres = 0.95)

# here we use the metabProfile previously calculated and return a "cleaned" profile.
metabProfile.clean <- ProfileCleanUp(metabProfile, timeSplit = 500,
                      r_thres = 0.95)

# Different cutoffs could be specified
metabProfile.clean <- ProfileCleanUp(metabProfile, timeSplit = 1000,
                      r_thres = 0.9)
```

quantMatrix *Create an intensity matrix using quantification masses*

Description

Create an intensity matrix using quantification masses. The quantification masses can be specified when importing the library file or by manually setting its values (see example).

Usage

```
quantMatrix(Lib, metabProfile, value = "maxint")
```

Arguments

Lib	A tsLib object created by ImportLibrary function.
metabProfile	A tsProfile object. The final result of the package. This object is generated by either Profile or ProfileCleanUp.
value	The default method to select automatically the quantification mass, in case it is not given by the user. 'maxint' selects the selective mass with the highest intensity. 'maxobs' selects the most observed mass, i.e., the one with less missing values.

Value

An intensity matrix with metabolites as rows and samples as columns. The matrix has two attributes: 'quantMass' a numeric vector that contains the quantification masses that were selected; 'isSelMass' a logical vector that indicates whether a quantification mass is also a selected mass.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

tsLib, tsMSdata

Examples

```
require(TargetSearchData)
data(TSExample)

# process chormatograms and get a profile
RI.path    <- file.path(find.package("TargetSearchData"), "gc-ms-data")
RIpath(sampleDescription) <- RI.path
refLibrary <- ImportLibrary(file.path(RI.path,"library.txt"))
refLibrary    <- medianRILib(sampleDescription, refLibrary)
corRI         <- sampleRI(sampleDescription, refLibrary, r_thres = 0.95)
peakData      <- peakFind(sampleDescription, refLibrary, corRI)
metabProfile  <- Profile(sampleDescription, refLibrary, peakData, r_thres = 0.95)

# show quant Matrix
quantMass(refLibrary)
```

```
# no quantMass have been defined yet, so are all zeros
# 0  0  0  0  0  0  0  0  0  0  0  0

# get a Matrix using use default values, ie, select the masses
# with the highest intensity
quantMat <- quantMatrix(refLibrary, metabProfile)
quantMat

# set the quantification Masses
quantMass(refLibrary)[1:3] <- c(89,86,100)
quantMat <- quantMatrix(refLibrary, metabProfile)
quantMat
```

`ri2rt` *Retention Time Index to Retention Time convertion*

Description

Convert retention time indices to retention times indices based on observed FAME RI and their standard values.

Usage

```
ri2rt(riTime, rt.observed, ri.standard)
```

Arguments

`riTime`	And RI vector or matrix to convert to Retention Time.
`rt.observed`	The observed FAME RT's. It could be a vector or a matrix.
`ri.standard`	The standard RI for each FAME

Details

This function is the inverse of `rt2ri`.

Value

The converted RT

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`RIcorrect`, `FAMEoutliers`

Examples

```
# RI standards
standard <- c(100, 200, 300, 400, 500)
# observed standard retention times
observed <- c(10.4, 19.3, 32.4, 40.2, 50.3)
# a random set of retention times
RI       <- runif(100,90,600)
# the corrected RIs
RT       <- ri2rt(RI, observed, standard)
```

RIcorrect *Peak picking from CDF files and RI correction*

Description

This function reads from CDF files, finds the apex intensities, converts the retention time to retention time index (RI), and writes RI corrected text files (a.k.a. RI files).

Usage

```
RIcorrect(samples, rimLimits = NULL, massRange = NULL, Window, IntThreshold,
pp.method = "ppc", showProgressBar = FALSE, baseline = FALSE,
baseline.opts = NULL )
```

Arguments

samples	A tsSample object created by ImportSamples function.
rimLimits	A tsRim object. If set to NULL, no retention time will be performed. See ImportFameSettings.
massRange	Deprecated. It is completely ignored but it is kept for compatibility with old scripts.
Window	The window used for smoothing. The number of points actually used is 2*Window + 1. It must be an integer. See details.
IntThreshold	Apex intensities lower than this value will be removed from the RI files.
pp.method	Peak picking method. Options are "smoothing", "gaussian" and "ppc". See details.
showProgressBar	Logical. Should the progress bar be displayed?
baseline	Logical. Should baseline correction be performed?
baseline.opts	A list of options passed to baselineCorrection.

Details

There are three pick picking methods available: "ppc", "smoothing", "gaussian".

The "ppc" method (default) implements the peak detection method described in the ppc package. It looks for the local maxima within a 2*Window + 1 scans for every mass trace.

The "smoothing" method calculates a moving average of 2*Window + 1 points for every mass trace. Then it looks for a change of sign (from positive to negative) of the difference between two consecutive points. Those points will be returned as detected peaks.

The "gaussian" method behaves similar to the "smoothing" method, but instead a gaussian smoother is used instead of the moving average.

To work out a suitable `Window` value, the following might be useful: `Window = (SR * PW -1) / 2`, where SR is the scan rate of the MS instrument and PW is the peak width. Because `Window` is an integer, the resulting value must be rounded. For example, for SR = 20 scans per second, a PW = 1.5 seconds, then `Window` = 14.5, which can be rounded to 15.

The RI file type is determined by the output of `fileFormat` method applied to the `tsSample` input object. To choose between the available formats ("binary" and "text"), select it with `fileFormat` method before calling `RIcorrect`.

Value

A retention time matrix of the found retention time markers. Every column represents a sample and rows RT markers.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportSamples`, `ImportFameSettings`, `NetCDFPeakFinding`, `FAMEoutliers`, `tsSample`, `tsRim`.

Examples

```
require(TargetSearchData)
# import refLibrary, rimLimits and sampleDescription.
data(TSExample)
# get the CDF files
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
cdfpath
list.files(cdfpath)
# update the CDF path
CDFpath(sampleDescription) <- cdfpath
# run RIcorrect (massScanRange = 85-320; Intensity Threshold = 50;
# peak detection method = "ppc", window = 15)
RImatrix <- RIcorrect(sampleDescription, rimLimits,
            Window = 15, pp.method = "ppc", IntThreshold = 50)

# you can try other parameters and other peak picking algorithm.
RImatrix <- RIcorrect(sampleDescription, rimLimits,
            Window = 15, pp.method = "smoothing", IntThreshold = 10)

RImatrix <- RIcorrect(sampleDescription, rimLimits,
            Window = 15, pp.method = "ppc", IntThreshold = 100)
```

---

riMatrix *Retention Time Index Matrix*

Description

A function to search for retention index RI markers.

Usage

```
riMatrix(samples, rim)
```

Arguments

samples	A tsSample object created by ImportSamples function.
rim	A tsRim object. See ImportFameSettings.

Details

This function works similar to RIcorrect, but searches for RI markers in RI files (not in CDF files). Can be used to retrieve the retention times of RI markers in already processed files.

Note that it does not perform any RI adjusment. See fixRI.

Value

A retention time matrix of the found retention time markers. Every column represents a sample and rows RT markers.

Author(s)

Alvaro Cuadros-Inostroza

See Also

RIcorrect, FAMEoutliers,ImportSamples, ImportFameSettings, fixRI

Examples

```
require(TargetSearchData)
# import refLibrary, rimLimits and sampleDescription.
data(TSExample)
# get the CDF files
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")

# select a subset of samples
smp <- sampleDescription[1:4]

# update the CDF path
CDFpath(smp) <- cdfpath

# make a copy of the RI markers object
rim <- rimLimits

# run RIcorrect
RImat <- RIcorrect(smp, rim, massRange = c(85,320),
           Window = 15, pp.method = "ppc", IntThreshold = 50)

# extract the retention times of the markers
RImat2 <- riMatrix(smp, rim)

# both matrices should be equal
stopifnot( all.equal(RImat, RImat2, tolerance=1e-8) )
```

rt2ri *Retention Time to Retention Time Index convertion*

Description

Convert retention times to retention indices based on observed FAME RI and their standard values.

Usage

```
rt2ri(rtTime, observed, standard)
```

Arguments

rtTime	The extracted RT's to convert
observed	The observed FAME RT's
standard	The standard RI for each FAME

Details

Linear interpolation, interpolation outside bounds are done with continued linear interpolation from the last two FAME's

Value

The converted RI

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

RIcorrect, FAMEoutliers

Examples

```
# RI standards
standard <- c(100, 200, 300, 400, 500)
# observed standard retention times
observed <- c(10.4, 19.3, 32.4, 40.2, 50.3)
# a random set of retention times
RT       <- runif(100,1,60)
# the corrected RIs
RI       <- rt2ri(RT, observed, standard)
```

sampleRI *Sample especific RI detection*

Description

Return a matrix of the sample specific RIs based on the correlating selective masses.

Usage

```
sampleRI(samples, Lib, r_thres = 0.95,
         columns = c("SPECTRUM", "RETENTION_TIME_INDEX", "RETENTION_TIME"),
         method = "dayNorm", minPairObs = 5, showProgressBar = FALSE,
         makeReport = FALSE, pdfFile = "medianLibRep.pdf")
```

Arguments

samples	A tsSample object created by ImportSamples function.
Lib	A tsLib object created by ImportLibrary function with corrected RI values. See medianRILib.
r_thres	A correlation threshold.
columns	A numeric vector with the positions of the columns SPECTRUM, RETENTION_TIME_INDEX, and RETENTION_TIME or a character vector with the header names of those columns.
method	Normalisation method. Options are "dayNorm", a day based median normalisation, "medianNorm", normalisation using the median of all the intensities of a given mass, and "none", no normalisation at all.
minPairObs	Minimum number of pair observations. Correlations between two variables are computed using all complete pairs of observations in those variables. If the number of observations is too small, you may get high correlations values just by chance, so this parameters is used to avoid that. Cannot be set lower than 5.
showProgressBar	Logical. Should the progress bar be displayed?
makeReport	Logical. If TRUE will report the RI deviations for every metabolite in the library.
pdfFile	The file name where the report will be saved.

Value

A matrix of correlating selective masses RI. Columns represent samples and rows the median RI of the selective masses.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

ImportSamples, ImportLibrary, medianRILib, tsLib, tsSample

Examples

```
require(TargetSearchData)
data(TSExample)

# get RI file path
RI.path <- file.path(find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path
# Import Library
refLibrary       <- ImportLibrary(file.path(RI.path,'library.txt'))

# get the sample RI
corRI <- sampleRI(sampleDescription, refLibrary, r_thres = 0.95)

# same as above, but changing the correlation threshold and the minimum number
# of observations
corRI <- sampleRI(sampleDescription, refLibrary, r_thres = 0.9,
minPairObs = 10)
```

TargetSearch-defunct *Defunct functions in package 'TargetSearch'*

Description

Functions listed here are defunct and no longer available.

Details

These function have been removed from 'TargetSearch' and no longer available. Use the replacement if any.

- `fixRIcorrection` has been replaced by `fixRI`.

TargetSearch-deprecated *Deprecated functions in package 'TargetSearch'*

Description

The functions listed here have been deprecated and may be removed as soon as of the next release.

Details

These functions are deprecated. Use the replacement if available.

- `TargetSearchGUI` has been deprecated. There are neither equivalent or replacements because everything can be achived by using common 'TargetSearch' functions.

`TargetSearchGUI` *A GUI for TargetSearch*

Description

Opens a Graphical User Interface (GUI, written using Tcl/Tk) to allow easy setting and manipulation of most processing parameters which control GC-MS Data Evaluation with *TargetSearch*.

NOTE: This function has been deprecated. Use the command line instead.

Usage

```
TargetSearchGUI()
```

Details

The GUI is intended to facilitate the use of *TargetSearch* for users unfamiliar with R otherwise. Many parameters that would be set calling the individual *TargetSearch* Functions as described in the manual can be set here 'in one go' before running the complete analysis.

Important Note: Please select the folder where you store your GC-MS Data (NetCDF or Apex) as the `Working Directory`. It is not yet possible to process data files from other/different locations.

The parameters:

- `Working Directory`: Use the *Browse*-button to select the folder on your hard drive containing all your GC-MS data files. The output of *TargetSearch* will be written to this folder too.
- `File Import`: Clicking *NetCDF Data* or *Apex Data* radio buttons will open a file select dialog. Choose the files you would like to be processed. You may check your selection pressing the *Show*-button.
- `Baseline Correction`: Clicking *on/off* button will perform baseline correction before peak detection. If selected, the `threshold` parameter is a numeric value between 0 and 1. A value of one returns a baseline above the noise, 0.5 in the middle of the noise and 0 below the noise. See `baselineCorrection` for further details.
- `Retention Index Correction`: Retention Index Correction is neccessary and applied only if you supply NetCDF Data (Apex Data contain already Retention Indices). You may *Load* or *Create* the search windows for your RI-Markers here.
- `Peak Detection`: *Search Windows* refers to the allowed RI deviation of your metabolites which are narrowed in 3 consecutive searches. *Intensity Counts threshold* defines the minimum apex intensity incorporated in the analysis. A value of 1 would include all peaks. *Mass Range* allows to limit the mass values (m/z) to be included in the analysis. *Smoothing* averages raw data to eliminate some inherent noise leading to multiple peaks otherwise.
- `Library`: A Library (to detect metabolites) usable by *TargetSearch* contains at least information about the metabolite 'Name', its expected 'RI' and the selective masses in its spectrum 'SEL_MASS'. You may *Load* or *Create* one yourself using the respective buttons. The parameter *no. of top masses* is the number of most intensive masses that will be taken from the spectrum, and *excluded masses* is a list of masses that will be excluded. A more detailed description of the file formats can be found in `ImportLibrary`.
- `Normalization`: This selects how the data will be normalized during the metabolite search. Options are `"dayNorm"`, a day based median normalization, `"medianNorm"`, normalization using the median of all the intensities of a given mass, and `"none"`, no normalization at all.

- `Final Profiles`: Here you may set the parameters used by the functions `Profile` and `ProfileCleanUp`. *timesplit* sets an RI window that will be used to look for metabolites that could have been redundanly identified. *correl. thr.* is the correlation threshold and *min. number of correlation samples* is a threshold used to make sure that correlations are computed with at least said number of observations.
- `Parameters`: You may *Save* the current parameters as an `*.RData` file or *Load* previously saved parameters to compare the outcome of different settings or just repeat the analysis.
- `Program`: *Run* starts to process all currently selected files using the current parameters and saving output to *Working Directory*. *Quit* closes the GUI.

Author(s)

Alvaro Cuadros, Jan Lisec

`text2bin` *Convert RI files from text to binary format and viceversa.*

Description

This function converts a list of RI files (peak list files) in text (binary) format to binary (text) format.

Usage

```
text2bin(in.files, out.files=NULL,
columns=c("SPECTRUM","RETENTION_TIME_INDEX","RETENTION_TIME"))

bin2text(in.files, out.files=NULL)
```

Arguments

`in.files`	A character string naming the input files.
`out.files`	A character string naming the output files. If `NULL`, the input file extensions will be changed accordingly ("txt" to "dat" and viceversa).
`columns`	A numeric vector with the positions of the columns `SPECTRUM`, `RETENTION_TIME_INDEX`, and `RETENTION_TIME` or a character vector with the header names of those columns. This parameter is required only for `text2bin`.

Details

The format of the input files is detected dinamically and error will be issued if it is incorrect.

Note that the respective `tsSample` object may need to be updated by using the method `fileFormat`.

Value

A character vector of the created files.

Author(s)

Alvaro Cuadros-Inostroza

See Also

ImportSamples, tsSample, RIcorrect

TSExample *Example GC-MS data for TargetSearch Package*

Description

A TargetSearch example GC-MS data. This datasets contains TargetSearch object examples generated from a E.coli salt stress experiment (See package TargetSearchData).

Usage

```
data(TSExample)
```

Format

The data contains the following objects:

sampleDescription a tsSample object. The sample description.

refLibrary a tsLib object. The reference library.

rimLimits a tsRim object. The RI markers definition.

RImatrix a matrix object. The retention time of the RI markers.

corRI a matrix object. The sample RI.

peakData a tsMSdata object. The intensities and RIs of all the masses that were searched for.

metabProfile a tsProfile object. The metabolite profile.

Details

This dataset contain only the objects. The actual source files are provided by the package TargetSearchData.

See Also

ImportLibrary, ImportSamples, ImportFameSettings,

Examples

```
require(TargetSearchData)

## The directory with the NetCDF GC-MS files
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
cdfpath
list.files(cdfpath)
samp.file <- file.path(cdfpath, "samples.txt")
rim.file  <- file.path(cdfpath, "rimLimits.txt")
lib.file  <- file.path(cdfpath, "library.txt")

# import files from package
sampleDescription <- ImportSamples(samp.file, CDFpath = cdfpath, RIpath = ".")
refLibrary        <- ImportLibrary(lib.file)
```

```
rimLimits         <- ImportFameSettings(rim.file, mass = 87)
# perform RI correction
RImatrix          <- RIcorrect(sampleDescription, rimLimits, massRange = c(85,320),
                     IntThreshold = 25, pp.method = "ppc", Window = 15)
# update median RI
refLibrary        <- medianRILib(sampleDescription, refLibrary)
# get the sample RI
corRI             <- sampleRI(sampleDescription, refLibrary, r_thres = 0.95)
# obtain the peak Intensities of all the masses in the library
peakData          <- peakFind(sampleDescription, refLibrary, corRI)
# make a profile of the metabolite data
metabProfile      <- Profile(sampleDescription, refLibrary, peakData, r_thres = 0.95)

# show the metabolite profile
profileInfo(metabProfile)
# show the matrix intensities
Intensity(metabProfile)
```

`tsLib-class` *Class for representing a reference library*

Description

This is a class representation of a reference library.

Objects from the Class

Objects can be created by the function `ImportLibrary`.

Slots

`Name`: `"character"`, the metabolite or analyte names.

`RI`: `"numeric"`, the expected retention time indices (RI) of the metabolites/analytes.

`medRI`: `"numeric"`, the median RI calculated from the samples.

`RIdev`: `"matrix"`, the RI deviation windows, k = 1,2,3. A three column matrix

`selMass`: `"list"`, every component is a numeric vector containing the selective masses.

`topMass`: `"list"`, every component is a numeric vector containing the top masses.

`quantMass`: `"numeric"`, the mass used for quantification.

`libData`: `"data.frame"`, additional library information.

`spectra`: `"list"`, the metabolite spectra. Each component is a two column matrix: m/z and intensity.

Methods

`[ signature(x = "tsLib")`: Selects a subset of metabolites from the library.

`$name signature(x = "tsLib")`: Access column `name` of `libData` slot.

`libId signature(obj = "tsLib")`: Returns a vector of indices.

`length signature(x = "tsLib")`: returns the length of the library. i.e., number of metabolites.

`libData signature(obj = "tsLib")`: gets/sets the `libData` slot.

`libName signature(obj = "tsLib")`: gets the `Name` slot.

`libRI signature(obj = "tsLib")`: gets the `RI` slot.

`medRI signature(obj = "tsLib")`: gets the `medRI` slot.

`refLib signature(obj = "tsLib")`: Low level method to create a matrix representation of the library.

`RIdev signature(obj = "tsLib")`: gets the RI deviations.

`RIdev<- signature(obj = "tsLib")`: sets the RI deviations.

`quantMass signature(obj = "tsLib")`: gets the quantification mass.

`quantMass<- signature(obj = "tsLib")`: sets the quantification mass.

`selMass signature(obj = "tsLib")`: gets the selective masses.

`show signature(object = "tsLib")`: show method.

`spectra signature(obj = "tsLib")`: gets the spectra.

`topMass signature(obj = "tsLib")`: gets the top masses.

Note

Some care is needed when using the methods `quantMass<-`, `selMass<-`, `topMass<-`. In order to be consistent, the first m/z value of the slot `topMass` and `selMass` are equal to `quantMass`, and the values of `selMass` are equal to the first couple of values of `topMass`. In other words, the following constrain is applied.

```
quantMass : x[0]
selMass   : x[0], x[1], ..., x[k]
topMass   : x[0], x[1], ..., x[k], x[k+1], ..., x[n]
```

where 1 <= k <= n and `x[i]` is a m/z value. Thus, using one these methods will change the values of the other slots. In the future, these methods will be deprecated, so it is better to not rely on them.

See the last example on how this can lead to unexpected results.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportLibrary`

Examples

```
showClass("tsLib")

# define some metabolite names
libNames   <- c("Metab1", "Metab2", "Metab3")
# the expected retention index
RI         <- c(100,200,300)
# selective masses to search for. A list of vectors.
selMasses  <- list(c(95,204,361), c(87,116,190), c(158,201,219))
# define the retention time windows to look for the given selective masses.
RIdev      <- matrix(rep(c(10,5,2), length(libNames)), ncol = 3, byrow = TRUE)
# Set the mass spectra. A list object of two-column matrices, or set to
```

```
# NULL if the spectra is not available
spectra    <- NULL
# some extra information about the library
libData    <- data.frame(Name = libNames, Lib_RI = RI)
# create a reference library object
refLibrary <- new("tsLib", Name = libNames, RI = RI, medRI = RI, RIdev = RIdev,
            selMass = selMasses, topMass = selMasses, spectra = spectra, libData = libData)

# get the metabolite names
libName(refLibrary)
# set new names
libName(refLibrary) <- c("Metab01", "Metab02", "Metab03")

# get the expected retention times
libRI(refLibrary)
# set the retention time index for metabolite 3 to 310 seconds
libRI(refLibrary)[3] <- 310
# change the seleccion and top masses of metabolite 3
selMass(refLibrary)[[3]] <- c(158,201,219,220,323)
topMass(refLibrary)[[3]] <- c(158,201,219,220,323)
# change the retention time deviations
RIdev(refLibrary)[3,] <- c(8,4,1)

######################################################################
######################################################################
# These examples show how changing a quantitaive or selective mass
# could lead to unexpected results.

# show quantMasses
quantMass(refLibrary)

# suposse that we want to change the quant mass of metabolite 1 to 96 due
# to a typo in the library. We could do just
quantMass(refLibrary)[1] <- 96

# however, we still see the mass 95 in the selective and top masses.
selMass(refLibrary)[[1]]
topMass(refLibrary)[[1]]

# to remove the mass 95, set the topMass and selMass explicitely, noting that
# the first masses coincides with 96 (the quantMass)
selMass(refLibrary)[[1]] <- c(96, 204, 361)
topMass(refLibrary)[[1]] <- c(96, 204, 361)
```

`tsMSdata-class` *Class for representing MS data*

Description

This is a class to represent MS data obtained from the sample.

Details

The method `as.list` converts every slot (`RI`, `RT`, and `Intensity`) of a `tsMSdata` object into a matrix. The converted matrices are stored in a list. Each converted matrix has an attribute called 'index' that relates the metabolite index with the respective rows. The components of the resulting list are named as the slots. If the slot `RT` is not defined or empty, then the output list will have only two components. ('RT' and 'Intensity').

Objects from the Class

Objects be created by calls of the form

Slots

`RI`: `"list"`, a list containing an RI matrix, one matrix per metabolite

`RT`: `"list"`, a list containing an RT matrix, one matrix per metabolite

`Intensity`: `"list"`, a list containing a peak intensity matrix, one matrix per metabolite

Methods

Intensity `signature(obj = "tsMSdata")`: gets the peak intensity list.

Intensity<- `signature(obj = "tsMSdata")`: gets the peak intensity list.

retIndex `signature(obj = "tsMSdata")`: gets RT list.

retIndex<- `signature(obj = "tsMSdata")`: sets the RI list.

retTime `signature(obj = "tsMSdata")`: gets the RT list.

retTime<- `signature(obj = "tsMSdata")`: sets the RT list.

show `signature(object = "tsMSdata")`: show function.

as.list `signature(object = "tsMSdata")`: coerce a list object. See details

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`FindPeaks`, `peakFind`

Examples

```
showClass("tsMSdata")
```

`tsProfile-class` *Class for representing a MS profile*

Description

This class is to represent a MS profile

Objects from the Class

Objects can be created by the function `Profile` or by

`new("tsMSdata",RI = [retention time index matrix],RT = [retention time matrix],Intensity = [peak intensity])`

Slots

`info`: `"data.frame"`, the profile information.

`RI`: `"list"`, a list containing RI matrices, one matrix per metabolite

`RT`: `"list"`, a list containing RT matrices, one matrix per metabolite

`Intensity`: `"list"`, a list containing peak-intensity matrices, one matrix per metabolite

`profRI`: `"matrix"`, the profile RI matrix.

`profRT`: `"matrix"`, the profile RT matrix.

`profInt`: `"matrix"`, the profile Intensity matrix.

Extends

Class `tsMSdata`, directly.

Methods

profileInfo `signature(obj = "tsProfile")`: get the profile information.

profileInfo<- `signature(obj = "tsProfile")`: set the profile information.

profileInt `signature(obj = "tsProfile")`: get the profile intensity matrix.

profileInt<- `signature(obj = "tsProfile")`: set the profile intensity matrix.

profileRI `signature(obj = "tsProfile")`: get the profile RI matrix.

profileRI<- `signature(obj = "tsProfile")`: set the profile RI matrix.

profileRT `signature(obj = "tsProfile")`: get the profile RT matrix.

profileRT<- `signature(obj = "tsProfile")`: set the profile RT matrix.

show `signature(object = "tsProfile")`: the show function.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`Profile`, `ProfileCleanUp`, `tsMSdata`

Examples

```
showClass("tsProfile")
```

tsRim-class *Class for representing retention index markers*

Description

This is a class to represent retention index markers.

Objects from the Class

Objects can be created by the function ImportFameSettings or by calls of the form new("tsRim",limits = [two column matrix with time limits],standard = [a vector with RI standards],mass = [m/z marker]).

Slots

limits: "matrix", two column matrix with lower and upper limits where the standards will be search. One row per standard.

standard: "numeric", the marker RI values.

mass: "numeric", the m/z marker.

Methods

rimLimits signature(obj = "tsRim"): gets the time limits.

rimLimits<- signature(obj = "tsRim"): sets the time limits.

rimMass signature(obj = "tsRim"): gets the m/z marker.

rimMass<- signature(obj = "tsRim"): sets the m/z marker.

rimStandard signature(obj = "tsRim"): gets the standars.

rimStandard<- signature(obj = "tsRim"): sets the standars.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

ImportFameSettings

Examples

```
showClass("tsRim")

# create a rimLimit object:
#  - set the lower (first column) and upper (second column) time limites to
#    search for standards.
Lim <- rbind(c(200, 300), c(400,450), c(600,650))
#  - set the retention indices of the standard
Std <- c(250000, 420000, 630000)
#  - set the mass marker
mass <- 87
#  - create the object
rimLimits <- new("tsRim", limits = Lim, standard = Std, mass = mass)
```

```
# sometimes you need to change the limits of a particular standard
rimLimits(rimLimits)[2,] <- c(410, 450)

# to change the mass value
rimMass(rimLimits) <- 85
```

---

tsSample-class *Class for representing samples*

Description

This is a class to represent a set of samples.

Objects from the Class

Objects can be created by the function ImportSamples or by calling the object generator function.

new("tsSample",Names = [sample names],CDFfiles = [list of CDF file names],RIfiles = [list of RI file names],CDFpath = [CDF files path],RIpath = [RI files path],days = [measurement days],data = [additional sample information],ftype = [RI file format])

Slots

Names: "character", the sample names.

CDFfiles: "character", the list of CDF file names.

RIfiles: "character", the list of RI file names.

CDFpath: "character", CDF files path.

RIpath: "character", RI file path.

days: "character", measurement days.

data: "data.frame", additional sample information.

Methods

[signature(x = "tsSample"): Selects a subset of samples.

$name signature(x = "tsSample"): Access column name of sampleData slot.

CDFfiles signature(obj = "tsSample"): list of CDF files.

RIfiles signature(obj = "tsSample"): list of RI files.

RIpath signature(obj = "tsSample"): The RI file path.

CDFpath signature(obj = "tsSample"): The CDF file path.

length signature(x = "tsSample"): number of samples.

sampleData signature(obj = "tsSample"): additional sample information.

sampleDays signature(obj = "tsSample"): measurement days.

sampleNames signature(obj = "tsSample"): sample names. The names must be unique

show signature(object = "tsSample"): the show funtion.

fileFormat signatureobj = "tsSample": Sets or gets the RI file format. Options are either "binary" or "text". See note below.

Notes

The method `fileFormat` only changes the internal information of the file type and not the files themselves. To actually change the files, use the functions `bin2text` and `text2bin`.

Note that the slot `Names` (i.e., the sample names/identifiers) must be unique. This allows sample selection by using sample identifiers as well as indices. Also, if columns are selected, the output will be either a vector or a `data.frame` depending on whether one or more columns were selected.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`ImportSamples`

Examples

```
showClass("tsSample")

# get a list of CDF files from a directory
require(TargetSearchData)
CDFpath <- system.file("gc-ms-data", package = "TargetSearchData")
cdffiles <- dir(CDFpath, "cdf")

# define the RI file path
RIpath  <- "."

# create the sample object
sampleDescription <- new("tsSample", CDFfiles = cdffiles, CDFpath = CDFpath, RIpath = RIpath)

##

# More parameters could be defined:
# define the RI files and the RI path
RIfiles <- sub("cdf$", "txt", paste("RI_", cdffiles, sep = ""))
RIpath  <- "."

# get the measurement days (the four first numbers of the cdf files, in this
# example)
days <- substring(cdffiles, 1, 4)

# sample names
smp_names <- sub(".cdf", "", cdffiles, fixed = TRUE)

# add some sample info
smp_data <- data.frame(CDF_FILE =cdffiles, GROUP = gl(5,3))

# create the sample object
sampleDescription <- new("tsSample", Names = smp_names, CDFfiles = cdffiles, CDFpath = CDFpath,
    RIpath = RIpath, days = days, RIfiles = RIfiles, data = smp_data)

# change the sample names (they must be unique)
sampleNames(sampleDescription) <- paste("Sample", 1:length(sampleDescription), sep = "_")

# chang the file paths (relative to the working path)
```

```
CDFpath(sampleDescription) <- "my_cdfs/"
RIpath(sampleDescription)  <- "my_RIs/"

## sample subsetting.
# select samples 1, 3 and 5
(sampleset <- sampleDescription[c(1, 3, 5)])

# or use sample IDs
(sampleset <- sampleDescription[c("Sample_1", "Sample_3", "Sample_5")])

## extract columns
# select column 'GROUP'
(group <- sampleDescription$GROUP)
# or
(group <- sampleDescription[, 'GROUP'])
```

`Write.Results` *Save TargetSearch result objects into files*

Description

This is a convenient function to save the TargetSearch result into text files.

Usage

```
Write.Results(Lib, metabProfile, quantMatrix=c('maxint','maxobs','none'),
prefix = NA)
```

Arguments

`Lib`	A `tsLib` object.
`metabProfile`	A `tsProfile` object. The final result of the package. This object is generated by either `Profile` or `ProfileCleanUp`.
`quantMatrix`	Should an intensity matrix using quantification masses be created? This parameter will be passed to `quantMatrix` and indicates whether the quantification mass should be chosen based on intensity or observations. The file will have the extension '.profile.quantmatrix.txt'. If `none`, then the file is not created.
`prefix`	A character string. This is used as a name prefix for the written files. `"TargetSearch-"` is used by default.

Value

This function doesn't return anything. Just print a message with the saved files.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`peakFind`, `Profile`, `ProfileCleanUp`, `tsLib`, `tsMSdata`, `tsProfile`, `quantMatrix`

writeLibText *Save a library object in text format*

Description

This function creates tab delimited text file with library information. The created file can be re-imported with the ImportLibrary function.

Usage

```
writeLibText(lib, file)
```

Arguments

lib	A tsLib object. A metabolite library.
file	A string naming the output file.

Author(s)

Alvaro Cuadros-Inostroza

See Also

tsLib, ImportLibrary

Examples

```
# get the reference library file
cdfpath <- file.path(find.package("TargetSearchData"), "gc-ms-data")
lib.file  <- file.path(cdfpath, "library.txt")

# Import the reference library
refLibrary <- ImportLibrary(lib.file)

# save it to a file
writeLibText(refLibrary, file="libraryCopy.txt")
```

writeMSP *Save spectra in MSP format to be visualized in NIST*

Description

This function creates MSP format file from peak intensities that can be viewed with NIST.

Usage

```
writeMSP(metlib, metprof, file, append = FALSE)
```

Arguments

metlib	A tsLib object. A metabolite library.
metprof	A tsProfile object. Usually the ouput of Profile or ProfileCleanUp functions.
file	A string naming the output file.
append	Logical. If TRUE the results will be appended to file. Otherwise, it will overwrite the contents of file.

Author(s)

Alvaro Cuadros-Inostroza

See Also

peakFind, Profile, ProfileCleanUp, tsLib, tsMSdata, tsProfile

Index