TargetSearch

November 11, 2009

R topics documented:

`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.pdf"`, 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`

Examples

```
require(TargetSearchData)
data(TargetSearchData)

# 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)
```

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"),
          showProgressBar = FALSE)
```

Arguments

`my.files`	A character vector naming files to be searched.
`refLib`	A numeric matrix with three columns. 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` and `RETENTION_TIME_INDEX` or a character vector with the header names of those columns.
`showProgressBar`	Logical. Should the progress bar be displayed?

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

```
require(TargetSearchData)
data(TargetSearchData)

# 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)
```

`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 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.

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

This function imports a metabolite library file that will be used to processed the GC-MS data.

Usage

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

Arguments

`libfile`	A character string naming a library file. See details.
`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_MASSES` is provided.
`ExcludeMasses`	Optional. A vector containing a list of masses that will be excluded.

Details

The library file is a tab delimited text file with the following column names.

- `Name` - The metabolite name.
- `RI` - The expected RI.
- `SEL_MASSES` - A list of selective masses separated with semicolon.
- `TOP_MASSES` - 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.

The columns `Name` and `RI` are mandatory. At least one of columns `SEL_MASSES`, `TOP_MASSES` 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.

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 = ".", ...)
```

Arguments

`sampfile`	A character string naming a sample file. 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.
`...`	Other options passed to `read.delim` function.

Details

The sample file is a tab-delimited text file with at least two columns:

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

The column names must be exactly those indicated, but the column order doesn't 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.

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
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"), 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` and `RETENTION_TIME_INDEX` 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(TargetSearchData)

# get RI file path
RI.path <- file.path(.find.package("TargetSearchData"), "gc-ms-data")
# update RI file path
RIpath(sampleDescription) <- RI.path
# 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 = c(85, 500), Window = 5, IntThreshold = 10
                  pp.method = "smoothing")
```

Arguments

`cdfFile`	A character string naming a netcdf file.
`massRange`	A two component numeric vector with the scan mass range to extract.
`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"` and `"ppc"`.

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 is a numeric vector with two components: lower and higher masses. All masses in that range will be extracted. Note that it is not possible to extract a discontinuous mass range.

There are two peak picking algorithms that can be used. The `"smoothing"` method smooths the m/z curves and then looks for a change of sign of the intensity difference between two consecutive points. The `"ppc"` uses a sliding window and looks for the local maxima. This method is based on R-package `ppc`.

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.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`peakCDFextraction`

Examples

```
require(TargetSearchData)
data(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], massRange = c(85, 320), 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 = c(85, 500))
```

Arguments

`cdfFile`	A character string naming a netcdf file.
`massRange`	A two component numeric vector with the scan mass range to extract.

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 is a numeric vector with two components: lower and higher masses. All masses in that range will be extracted. Note that it is not possible to extract a discontinuous mass range.

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.

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"),
          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 column positions of `SPECTRUM` and `RETENTION_TIME_INDEX` 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(TargetSearchData)

# 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)
# show peak Intensities.
head(Intensity(peakData))

# How to get intensities for a particular metabolite
#
# make a library index using top masses
```

```
libId <- libId(refLibrary, sel = FALSE)
# get the peak intensities of Metabolite 1, for example, of every mass
int.1 <- Intensity(peakData)[libId == 1,]
# this assigns the mass values to the row names of int.1
rownames(int.1) <- topMass(refLibrary)[[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(TargetSearchData)
# plot Retention index standards 1 to 3
plotFAME(sampleDescription, RImatrix, 1)
plotFAME(sampleDescription, RImatrix, 2)
plotFAME(sampleDescription, RImatrix, 3)
```

plotPeak *Plot peaks*

Description

Plot selected ions in a given time range.

Usage

```
plotPeak(rawpeaks, time.range, masses, cdfFile = NULL,  useRI = FALSE,
         rimTime = NULL, standard = NULL, massRange = c(85, 500), ...)
```

Arguments

`rawpeaks`	A two component list containing the retention time and the intensity matrices. 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.
`...`	Further options passed to `matplot`.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`RIcorrect`, `tsMSdata`, `tsRim`, `peakCDFextraction`, `matplot`

Examples

```
require(TargetSearchData)
data(TargetSearchData)

# 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
plotPeak(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))
plotPeak(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(TargetSearchData)

# 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(TargetSearchData)

# 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")
```

`ProfileCleanUp` *Reduce redundancy of the profile*

Description

This function reduces/removes redundancy in a profile.

Usage

```
ProfileCleanUp(Profile, timeSplit = 500, r_thres = 0.95)
```

Arguments

`Profile`	A `tsProfile` object. See `Profile`.
`timeSplit`	A RI window.
`r_thres`	A correlation threshold.

Details

Metabolites that are inside a `timeSplit` window will be correlated to see whether the metabolites are the same or not, by using `r_thres` as a cutoff. If so, the intensities and RI will be averaged and the metabolite with more correlating masses will be suggested.

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 "Intensity"`	A matrix with the averaged intensities of the correlating masses.
`slot "RI"`	A matrix with the averaged RI of the correlating masses.

Author(s)

Alvaro Cuadros-Inostroza, Matthew Hannah, Henning Redestig

See Also

`Profile`, `tsProfile`

Examples

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

# 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)
```

`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 =
```

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.

Value

A tsProfile object. The slots are:

Info	A data frame with a profile of all masses that correlate.
Intensity	A matrix with the averaged intensities of the correlating masses.
RI	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(TargetSearchData)

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

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.

Usage

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

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`	A two component vector of m/z range used by the GC-MS machine.
`Window`	The window used for smoothing. 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`	Peak picking method. Options are either "smoothing" or "ppc". See details.
`showProgressBar`	Logical. Should the progress bar be displayed?

Details

There are two pick picking methods available: "smoothing" and "ppc".

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 "ppc" method 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.

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(TargetSearchData)
# 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, massRange = c(85,320),
            Window = 15, pp.method = "ppc", IntThreshold = 50)

# you can try other parameters and other peak picking algorithm.
RImatrix <- RIcorrect(sampleDescription, rimLimits, massRange = c(85,320),
            Window = 15, pp.method = "smoothing", IntThreshold = 10)

RImatrix <- RIcorrect(sampleDescription, rimLimits, massRange = c(85,320),
            Window = 15, pp.method = "ppc", IntThreshold = 100)
```

`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"),
         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` and `RETENTION_TIME_INDEX` 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.
`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(TargetSearchData)

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

# 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` *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>

`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.

`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 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.

`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.

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
# an empty list if the spectra is not available
spectra    <- list()
# 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, libD

# 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)
```

---

tsMSdata-class *Class for representing MS data*

Description

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

Objects from the Class

Objects be created by calls of the form

Slots

`RI`: `"matrix"`, an RI matrix.

`RT`: `"matrix"`, an RT matrix.

`Intensity`: `"matrix"`, an peak intensity matrix.

Methods

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

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

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

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

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

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

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

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`: `"matrix"`, an RI matrix.

`RT`: `"matrix"`, an RT matrix.

`Intensity`: `"matrix"`, an peak intensity matrix.

Extends

Class `tsMSdata`, directly.

Methods

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

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

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])`

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")`: additiona sample information.

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

sampleNames `signature(obj = "tsSample")`: sample names.

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

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 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)

# 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 = CD
    RIpath = RIpath, days = days, RIfiles = RIfiles, data = smp_data)

# changing the sample names
sampleNames(sampleDescription) <- paste("Sample", 1:length(sampleDescription), sep = "_")

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

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

`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, peakData, finalProfile, prefix = NA)
```

Arguments

`Lib`	A `tsLib` object.
`peakData`	A `tsMSdata` object.
`finalProfile`	A `tsProfile` object. The final result of the package. This object is generated by either `Profile` or `ProfileCleanUp`.
`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`

Index