Package 'SIMAT'

November 13, 2024

Type Package

Title GC-SIM-MS data processing and alaysis tool

Version 1.38.0

Date 2018-08-02

Author M. R. Nezami Ranjbar <nranjbar@vt.edu>

Maintainer M. R. Nezami Ranjbar <nranjbar@vt.edu>

Depends R (>= 3.5.0), Rcpp (>= 0.11.3)

Imports mzR, ggplot2, grid, reshape2, grDevices, stats, utils

Suggests RUnit, BiocGenerics

Description This package provides a pipeline for analysis of GC-MS data acquired in selected ion monitoring (SIM) mode. The tool also provides a guidance in choosing appropriate fragments for the targets of interest by using an optimization algorithm. This is done by considering overlapping peaks from a provided library by the user.

License GPL-2

biocViews ImmunoOncology, Software, Metabolomics, MassSpectrometry

URL http://omics.georgetown.edu/SIMAT.html

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

git_branch RELEASE_3_20

git_last_commit a091a2f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-12

Contents

`SIMAT-package` *This package is for processing GC-SIM-MS data*

Description

This package provides a pipeline for analysis of GC-MS data acquired in selected ion monitoring (SIM) mode. The tool also provides a guidance in choosing appropriate fragments for the targets of interest by using an optimization algorithm. This is done by considering overlapping peaks from a provided library by the user.

Details

Package:	SIMAT
Type:	Package
Version:	0.99.2
Date:	2015-03-01
License:	GPL-2

Author(s)

Mo R. Nezami Ranjbar,

References

M. R. Nezami Ranjnar, C. Di Poto, Y. Wang, and H. W. Ressom, "SIMAT: GC-SIM-MS Data Analysis Tool", submitted to BMC Bioinformatics.

See Also

http://omics.georgetown.edu/SIMAT.html

getEIC *Get the EIC of a peak*

Description

This function retrives the EIC of one peak in one run.

Usage

```
getEIC(Run = list(), compound = "Analyte", ms0 = numeric(), sp0 = numeric(),
                    rt0 = numeric(), drt = 10/60, dsc = 10/2, ri0 = 0,
                    weight = 2/3, deltaRI = 20, calibRI = NULL)
```

Arguments

Run	a list containing the information of one run obtained by readCDf funciton.
compound	a character vector for the name of the target.
ms0	a numeric vector of mass of fragments of the target.
sp0	a numeric vector of intensities of fragments of the target.
rt0	a numeric value of the expected retention time of the target.
drt	a numeric value of the retention time window width in seconds, optional but recommended
dsc	a numeric value of average half peak width based on the TICs, optional but recommended
ri0	a numeric value of the retention index of the target from library, optional but recommended
deltaRI	a numeric value for the penalty on the retention index similarity score
weight	a numeric value in [0,1] interval to calculated a combined weighted similarity scores based on Apex and area under EIC curve
calibRI	a function to calculate retention index based on RI calibration information, can be obtained by getRI function, optional. It is also used to estimate the retention time related to a retention index.

Details

This function accepts several parameters including the expected retention, the mass and intensity of several fragments, the retention index of the peak, the extracted data from a raw netCDF file in peak table format, a certain range for searching the retention time, i.e. the retention time window, average half peak width. The function uses a similarity score based on a combined measure from spectral matching and RI similarity, if RI is available, and finds the most appropriate peak in considering the target information.

Value

A list containing the peak information:

rtApex	a numeric value of retention time of the apex based on the quantifier fragment
intApex	a numeric value of the intensity of the quantifier fragment at its apex

RI	a numeric value of retention index related to the rtApex
area	area under EIC for all fragments of the related target
EIC	intensity of EIC profile of all fragments
RT	retention times of EIC profiles
ms	mass of fragments
sp	intensity of the fragments based on the reference spectrum
rt0	a numeric value of library retention time
ri0	a numeric value of library retention index
compound	a character vector cintaining the name of the target.

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getPeak

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# load targets information
data(Targets)

# get the corresponding peak of a target
peakEIC <- getEIC(Run = Run, compound = Targets$compound[1],
                ms0 = Targets$ms[[1]], sp0 = Targets$sp[[1]],
                rt0 = Targets$rt[1], ri0 = Targets$ri[1])
```

---

getPeak *Get all peaks corresponding to targets in several runs*

Description

This function retrives all the corresponding peaks of targets of interest in several runs.

Usage

```
getPeak(Run = list(), file.name = character(), Targets = list(),
            target.file.name = character(), drt = 10/60, dsc = 14/2,
                weight = 2/3, deltaRI = 20, calibRI = NULL, rt.sort = FALSE)
```

Arguments

Run	a list including a single run information, if provided, the next argument, i.e. file.name, is ignored.
file.name	a character vector of file names, i.e. the run file names generated by readCDf.
Targets	a list including the Targets information, e.g. acquired by getTarget function.
target.file.name	a character object, i.e. string, of the name of the file generated by getTarget.
drt	a numeric value of the retention time window width in seconds, optional but recommended
dsc	a numeric value of average half peak width based on the TICs, optional but recommended
weight	a numeric value in [0,1] interval to calculated a combined weighted similarity scores based on Apex and area under EIC curve.
deltaRI	a numeric value for the penalty on the retention index similarity score.
calibRI	a function to calculate retention index based on RI calibration information, can be obtained by getRI function, optional.
rt.sort	a boolean value to sort the targets based on retention time before performing peak detection, optional

Details

This function performs peak detection for a list of targets in several runs by calling getEIC. The list of targets should include the information on retention times, retention indexes, mass and intensity of the fragments. It can also include the name of the targets, i.e. compounds.

Value

a list of list objects, one for each run. Each inner list represents a single run containing peaks information, i.e. the same as getEIC output.

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getEIC

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# load targets information
data(Targets)

# get all the corresponding peaks of the target list
runPeaks <- getPeak(Run = Run, Targets = Targets)
```

getPeakScore *Calculate the similarity score for all peaks*

Description

This function calculates and output the similarity scores based on spectral matching and RI similarity for multiple peaks.

Usage

```
getPeakScore(runPeaks = list(), deltaRI = 20, weight = 2/3, plot = FALSE)
```

Arguments

runPeaks	a list of peaks of runs, e.g. generated by calling getPeak.
deltaRI	a numeric value for the penalty on the retention index similarity score
weight	a numeric value in [0,1] to calculated a combined weighted similarity scores based on Apex and area under EIC curve.
plot	a logical value to generate the histogram of the scores, default is FALSE.

Details

By calling getScore(), this function calculates the similarity score for multiple peaks in multiple runs. This is performed by using spectral and retention index information together.

Value

a matrix of scores with rows as compounds and columns as runs

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getScore

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# load targets information
data(Targets)

# get all the corresponding peaks of the target list
runPeaks <- getPeak(Run = Run, Targets = Targets)
```

```
# get the scores for all analytes in all runs
Scores <- getPeakScore(runPeaks = runPeaks)
```

---

getRI *Generate a calibration function*

Description

This function generates a calibration function which can be used to calculate the retention index of a compound, given its retention time together with the retention indexes of RI standards and their retention times.

Usage

```
getRI(RItable = data.frame())
```

Arguments

RItable	a data frame of retention times and indexes of RI standardsPeaks, e.g. generated by calling getRIStandard.

Details

The input of this function is an RItable. RItable is the table of RTs and correponding RIs of RI standards measured using an RI run. The user can get the RItable using the getRIStandard function. The output of this function is a function which can be used for retention index calculation providing a retention time. It also enables the user to estimate the retention time of a provided retention index.

Value

A calibration function which accepts a retention time and outputs corresponding retention index. The user can also provide a retention index and estimate the related retention time based on RI calibration.

Author(s)

Mo Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getRIStandard

Examples

```
# load retention index table from RI standards
data(RItable)

# create a calibration function
calibRI <- getRI(RItable)

# perform RI calibration for a certain RT = 12.32min
calibRI(12.32)

# estimate the RT of an RI based RI calibration
calibRI(ri = 1150)
```

`getRIStandard` *Get the RI standard information*

Description

This function generates a table which can be used for RI calibration when calling the `getRI` function It retrieves the retention indexes of RI standards and their retention times in a table, e.g. data frame.

Usage

```
getRIStandard(file.name = character())
```

Arguments

`file.name`	a string including the full name of a `csv` file including the information of RI standards in two columns, one for retention time and one for retention index.

Details

The input of this function is a table in `.csv` format RItable. The file should include the information of RI standards in two columns, one for retention times and one for retention index of the standards. The information is usually extracted by running a mixture of RI standards for calibration. The output is a table, i.e. a data frame, including the retention time and retention index of the RI standards which can be further used when calling `getRI` function.

Value

A data frame including retention times and retention idexes of the RI standards

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

`getRI`

Examples

```
# load an example data set
extdata.path <- system.file("extdata", package = "SIMAT")
datafile = file.path(extdata.path, "RIStandards.csv")

# read RItable from file
RItable <- getRIStandard(file.name = datafile)
```

`getScore` *Calculate similarity score for a peak*

Description

This function calculates and output the similarity score for a peak while a reference spectrum is provided. The score is based on spectral matching and RI difference between the true peak and the reference.

Usage

```
getScore(trueSpec = numeric(), refSpec = numeric(),
                        trueRI = 0, refRI = 0, deltaRI = 30)
```

Arguments

`trueSpec`	a numeric vector of intensity values for the measure, i.e. true, spectrum.
`refSpec`	a numeric vector of the intensity values for the reference spectrum.
`trueRI`	a numeric value of the measured, i.e. true, retention index
`refRI`	a numeric value of the reference retention index
`deltaRI`	a numeric value for the penalty on the retention index similarity score

Details

By calling getScore(), this function calculates the similarity score for multiple peaks in multiple runs. This is performed by using spectral and retention index information together.

Value

numeric

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getPeakScore

Examples

```
getScore(trueSpec = c(97, 995, 278, 343), refSpec = c(100, 1000, 250, 390))
```

`getTarget` *Get target information*

Description

Asssuming that the targets are provided in a NIST mass spectral library, i.e. MSL, format, this function reads the list of targets and the related information.

Usage

```
getTarget(Method = "target", target.file = character(),
            library.file = character(), path = getwd(),
            library.path = getwd(), Library = list(), target.table = list(),
            deltaRI = numeric(), deltaRT = numeric(), Save = FALSE)
```

Arguments

`Method`	a srting with three possible values: "target", "library", and "combined". The "target" case means the user only provides a `.MSL` file as a list of targets. In the "library" mode, the user provides only a library in `.MSL` format, but it is required to provide a `target.table` to determine the name of the targets and the selected mass. The `target.table` can be obtained by using `getTargetTable` function. In the "combined" method, the user provides a list of targets as `target.table` and a library in `.MSL` format where the fragments are selected using optimization.
`target.file`	a string including the full name of a `.MSL` file including the target compounds information.
`library.file`	a string including the full name of a `.MSL` file including the library compounds information. The library can include background compounds, e.g. an in-house library. A library can also directly imported using the `Library` argument.
`path`	a string including the full path to the location of the file.name, optional
`library.path`	a string inlcuding the full path to the location of the library file, optional
`Library`	a list of library information, this is used when the library file is not provided.
`target.table`	a list including the name of the compounds, this is required when a library is provided to extract the targets information from the library. This value can be obtained using `getTargetTable` function.
`deltaRI`	a numeric value for the penalty on the retention index similarity score
`deltaRT`	a numeric value for the penalty on the retention time similarity score
`Save`	if `FALSE`, the function returns the targets as a list, otherwise, it saves the list as an `rds` object.

Details

By calling `readMSL`, this function reads the target list and retrieves the reltated informatin such as retention time, retention index, mass and intensity of the fragments, and compound names. The user can determine the quantifier mass by using a single value for all targets, or a vector of values pointing to the index of the fragment of ineterest for each target. Also, the targets can be chosen by omptimization based on overlapping compounds if a library is provided. The library should not include any targets, but it can be an in-house library built from background compounds detected in measurements of the same type of samples.

Value

A list containing:

`compound`	a character vector containing the names of the targets
`ms`	a list of numeric vectors of fragment mass of the targets
`sp`	a list of numeric vectors of fragment intensities of the targets
`rt`	a numeric vector of retention times of the targets
`ri`	a numeric vector of the retention indexed of the targets
`quantFrag`	a numeric vector showing the index of quantifier fragment in `ms` and `sp` fields.
`sortedFrag`	a list of numeric vectors showing the order of fragments from the most favorable to the list favorable choice for a quantifier.

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

`getTargetTable`, `optFrag`

Examples

```
# load the target table information
data(target.table)

# load the background library to be used with fragment selection
data(Library)

# get targets info using target table and provided library
Targets <- getTarget(Method = "library", Library = Library,
                                          target.table = target.table)
```

`getTargetTable` *Get target table information*

Description

Asssuming that the targets are provided in a table format as a txt file, this function reads the list of targets and the related information.

Usage

```
getTargetTable(target.table.file = character())
```

Arguments

`target.table.file`
: a string including the full name of a text file including the target compounds information.

Details

This function gets the targets table from user by reading the targets info including the names of the compounds together with the the mass of selected fragments, the names should be clear enough to be searched by getTarget function when required. Each line of the target table file icludes two keywords, "name", and "mass" or "numfrag", where the full name of the compound comes after "name" and the selected masses after "mass" or number of desired fragments after "numfrag". The "mass" is used when no optimization is expected and the corresponding fragments are defined by users. The "numfrag" is used when the user does not provide the fragments for monitoring and asks them to be selected by optimization. The function is not case-sensitive and the items can be separated by tabs or spaces. Also, it is not required to have the same number of masses for different compounds. The table can be created using any tool with any format, e.g. txt, while it is saved as a text file with a unicode format. Below is an example:

Name L-valine mass 55 72 118 Name urea mass 66 74 189 190

Value

A list containing:

`compound`	a character vector of compound names
`ms`	a list of numeric vectors of selected mass for each target
`numFrag`	a numeric vector of the desired number of fragments for each target

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

`getTarget`

Examples

```
# load an example data set
extdata.path <- system.file("extdata", package = "SIMAT")
datafile = file.path(extdata.path, "TargetTable.txt")

# read target table information form file
target.table <- getTargetTable(target.table.file = datafile)
```

Library *Extracted compound information from an MSL library.*

Description

The information in this data set, was obtained by using `readMSL` function. It includes a variable called `Library` which contains compound names, their retention time and retention index, the mass and the intensity of the fragments in the spectrum of each compound.

Usage

```
data(Library)
```

Format

A list with the same number of observations on the following 5 variables:

`rt` a numeric vector of retention times of the compounds.

`ri` a numeric vector of retention indexes of the compounds.

`compound` a character vector of name of the compounds.

`ms` a list of numeric vectors of mass of fragments for each compound.

`sp` a list of numeric vectors of intensity of fragments for each compound.

Details

This is the compound information extracted from an `MSL` file. It was obtained by using `readMSL` function. This library is provided as an example to be used for functions in the package.

Value

A list

References

http://omics.georgetown.edu/SIMAT.html

Examples

```
data(Library)
```

optFrag *Fragment selection through optimization*

Description

This function can be used to select the quantifier fragments through optimization. The optimization criteria are less overlapping targets or compounds with higher intensities.

Usage

```
optFrag(Targets = list(), Library = list(), target.table = list(),
            deltaRI = 20, deltaRT = 4, numFrag.default = 4, forceOpt = FALSE)
```

Arguments

`Targets`	a list of targets and their information including retention time and index, mass and intensity of the fragments extracted from a target list
`Library`	a list of compounds and their information including retention time and index, mass and intensity of the fragments extracted from an .MSL library.
`target.table`	a list including the name of the compounds, this is required when a library is provided to extract the targets information from the library. This value can be obtained using `getTargetTable` function.
`deltaRI`	a numeric value for the penalty on the retention index similarity score
`deltaRT`	a numeric value for the penalty on the retention time similarity score
`numFrag.default`	a numeric value, where neither the number of fragments, i.e. `numFrag`, nor the mass of suggested fragments are not provided, the algorithm chooses `numFrag.default` fragments from the library.
`forceOpt`	a logical value to force the optimization where the default is `FALSE`. When it is set to TRUE, even when fragment masses are provided by user, the algorithm picks the omptimized fragments. The number of selected fragments is then decided by the number of suggested fragments or `numFrag.default`.

Details

This function is called in `getTarget` function to optimize the selection of the fragments where fragments with higher intensity and less overlapping profile with neigbour compounds are preferred. Therefore, it is required that the user provides a library, e.g. stablished, in-house, or a combination of both. It is important that the library does not include any targets. The function sorts the fragments based on their intensities and checks if there are any overlapping compounds, e.g. based on retention time or retention index, which shares the same fragment, i.e. the same mass in the spectra. By penalizing the shared fragments based on their retention time or index difference with overlapping compounds, and also considering the intensity of the fragments, fragments are priorotized. As a result the quantifier mass can be selected from the top. It is recommended to provide the RI information, but if it is not available, the code uses retention times for this purpose.

Value

list

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getTarget, getTargetTable

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# load the target table information
data(target.table)

# load the background library to be used with fragment selection
data(Library)

# forcing optimization to get the targets info
optTargets <- optFrag(Library = Library, target.table = target.table,
                                                        forceOpt = TRUE)
```

---

plotEIC *Plotting EIC of one peak*

Description

This function can be used to plot the EIC profile of a peak, where the peak has been optianed by getPeak or getEIC functions. The plot also shows a pseudo peak, which is an illustratiion based on reference spectrum.

Usage

```
plotEIC(peakEIC = list(), fig.name = character())
```

Arguments

peakEIC	a list inclduing the EIC information of a peak
fig.name	a character vector (string), if provided, the figure is saved in pdf format with this name.

Details

This function plots the EIC profile of one peak. The peak information can be obtained using `getPeak` or `getEIC` finctions. The peak profile includes the measured intensities in the raw data correponding to a certain target in on specific run. The intensities are plotted versus recorded retention time in the sample. Different colors are used for different fragments and the mass of each fragment is included in the legend of the plot. Also, to make the visual comparison between the true and reference spcetra available, a pseudo peak based on the reference spectra is provided in a subplot showing the ratios between fragments. Finally, the expected and actual retention times of the target is shown in gray and red respectively.

Value

A logical value which is TRUE if the resulted plot is saved

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

plotTIC

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# load targets information
data(Targets)

# get all the corresponding peaks of the target list
runPeaks <- getPeak(Run = Run, Targets = Targets)

# plot the EIC of the first target
plotEIC(runPeaks[[1]][[1]])
```

`plotTIC` *Plotting TIC of one run*

Description

This function can be used to plot the total ion chromatogram (TIC) profile of a run, showing the accumulated measurements for all fragments versus time. The run can be optianed by `getPeak`.

Usage

```
plotTIC(Run = list(), file.name = character())
```

Arguments

Run	a list inclduing the information of all detected peaks in one run.
file.name	a character vector (string) of names of the runs which is used to name the figure file when saving, if the Run is provided, this argument is ignored.

Details

This function plots the TIC profile of one run. The run information can be obtained using getPeak finction. The TIC is the sum of the intensities from all masses at each scan. If argument Run is provided, the funtion plots the TIC for that run. If a list of file names is provided, the function saves a corresponding plot for each run with the related name.

Value

A logical value which is TRUE if the resulted plot is saved

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

plotEIC

Examples

```
# load an RData file including a single run data acquired by readCDF
data("Run")

# plot TIC of the run
plotTIC(Run = Run)
```

---

putTargetTable *Put target table information*

Description

Asssuming that the targets are provided in a NIST mass spectral library, i.e. MSL, format, this function reads the list of targets and the related information.

Usage

```
putTargetTable(target.table = list(), target.table.file = character())
```

Arguments

target.table	a list containing the target table information, e.g. obtained using getTargetTable.
target.table.file	a string including the full name of a text file including the target compounds information.

Details

This function writes the target table information in a csv file so the user can open the results in text and table editors.

Value

A logical value

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

getTargetTable

Examples

```
# load the target table information
data(target.table)

# create a subset of the table
target.table.sub <- list()
target.table.sub$compound <- target.table$compound[1:2]
target.table.sub$ms <- target.table$ms[1:2]

# write the subset into a csv file
putTargetTable(target.table = target.table.sub,
               target.table.file = "TargetTableSub.csv")
```

---

readCDF *Read raw mass spectrometry data in netCDF files*

Description

This function read raw netCDF files and provides a list of peaks and retention time of scans.

Usage

```
readCDF(path = getwd())
```

Arguments

path	a string including the full path of the CDF files, optional.

Details

This function gets a path which is optional. In the path, it searches for CDF files and reads them one by one. An RData file is created for each file separately, keeping the original file names. Each file includes a list of peaks, i.e. a combination of mass and intensities of scans, and the retention times of the scans.

Value

a character vector inlcuding the names of the imported netCDF files. The output then can be used with getPeak function.

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

readMSL

Examples

```
# load an example data set
extdata.path <- system.file("extdata", package = "SIMAT")

# read CDF files
file.name <- readCDF(path = extdata.path)
```

---

readMSL *Read mass spectral library (MSL) files*

Description

This funtion read data in NIST mass spectral library (MSL) format and returns a list of compounds, with their names, retention times, retention indexes (if provided), together with mass and intensity of all fragments.

Usage

```
readMSL(file.name = character(), path = getwd(), Save = FALSE)
```

Arguments

file.name	a string including the full name of a .MSL file including the library information.
path	a string including the full path to the location of the file.name, optional
Save	if TRUE, the function also saves the list as an RDdata object. It always returns the targets as list.

Details

NIST mass spectral library, i.e. MSL, data files include compound names and their spectral information. The spectral information consist of number of fragments for each compound, together with the mass and relative and usually normalized intensity of each fragment. Also, the retention time and retention index of the compounds are provided which helps improve the identification by avoding false positives. This functions usually is used indirectly by calling getTarget function.

Value

A list including target information:

rt	a numeric vector of retention times
rt	a numeric vector of retention indexes
ms	a list including vectors of fragment masses of each target
sp	a list including vectors of fragment intensities of each target
compound	a list including character vectors of the target chemical names

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

writeMSL, readCDF

Examples

```
# load an example data set
extdata.path <- system.file("extdata", package = "SIMAT")

# get the list of targets in a file with MSL format from Example Data Set
Targets1 <- readMSL(file.name="Targets_1.MSL", path = extdata.path)
```

RItable — *Extracted RI standard information from a* csv *file.*

Description

The information in this data set, was obtained by using `getRIStandards` function. It includes a variable called `RItable` which contains the retention index and measured retention times of the RI standards.

Usage

```
data(RItable)
```

Format

A data frame including retention times and retention idexes of the RI standards

`rt` a numeric vector of retention times of the RI standards.

`ri` a numeric vector of retention indexes of the RI standards.

Details

This is the data extracted from a RI standard information table. It was obtained by using `getRIStandards` function. The data set is provided as an example to be used for functions in the package. More examples can be found in the reference.

Value

A data frame

References

http://omics.georgetown.edu/SIMAT.html

Examples

```
data(RItable)
```

Run — *Extracted peaks from one SIM run.*

Description

The information in this data set, was obtained by using `readCDF` function. It includes a variable called `Run` which contains scans, i.e. mass and intensity pairs, and retention time of the scans for the run.

Usage

```
data(Run)
```

Format

A list with variable number of observations on the following 4 variables:

`rt` a numeric vector of retention times of the scans.

`sc` a numeric vector of scan indexes from 1 to the number of scans.

`tic` a numeric vector of the total ion chromatogram (TIC) of the run.

`pk` a list including same number of items as `sc` field where each item of the list includes a matrix with two columns containing the mass and related intensity of each mass

Details

This is the data extracted from one SIM run. It was obtained by using `readCDF` function. The data set is provided as an example to be used for functions in the package. More examples can be found in the reference.

Value

A list

References

http://omics.georgetown.edu/SIMAT.html

Examples

```
data(Run)
```

---

target.table *Extracted target information from a* csv *file.*

Description

The information in this data set, was obtained by using `getTargetTable` function. It includes a variable called `target.table` which contains compound names, and the mass of the fragments in the spectrum of each compound selected for SIM analysis.

Usage

```
data(target.table)
```

Format

A list with the same number of observations on the following 2 variables:

`compound` a character vector of name of the compounds.

`ms` a list of numeric vectors of mass of fragments for each compound.

Details

This is the data extracted from a target information table. It was obtained by using `getTargetTable` function. The data set is provided as an example to be used for functions in the package. More examples can be found in the reference.

Value

A list

References

http://omics.georgetown.edu/SIMAT.html

Examples

```
data(target.table)
```

`Targets` *Targets information in a list.*

Description

The information in this data set, was obtained by using `getTarget` function. It includes a variable called `Targets` which contains targets information, including the compound name, retention time, retention index, together with mass and intensity of the fragments.

Usage

```
data(Targets)
```

Format

A list with variable number of observations on the following 7 variables:

compound a character vector containing the names of the targets

ms a list of numeric vectors of fragment mass of the targets

sp a list of numeric vectors of fragment intensities of the targets

rt a numeric vector of retention times of the targets

ri a numeric vector of the retention indexed of the targets

quantFrag a numeric vector showing the index of quantifier fragment in `ms` and `sp` fields.

sortedFrag a list of numeric vectors showing the order of fragments from the most favorable to the list favorable choice for a quantifier.

Details

This is the data extracted from a library using a target.table. It was obtained by using `getTarget` function. The data set is provided as an example to be used for functions in the package. More examples can be found in the reference.

Value

A list

References

http://omics.georgetown.edu/SIMAT.html

Examples

```
data(Targets)
```

`writeMSL` *Write targets information into a file with mass spectral library (MSL) format.*

Description

This funtion gets the targets information and writes the data in the NIST mass spectral library (MSL) format.

Usage

```
writeMSL(Targets = list(), target.file.name = character())
```

Arguments

`Targets`	a list including the Targets information, e.g. acquired by `getTarget` function.
`target.file.name`	a character object, i.e. string, of the name of the output file.

Details

NIST mass spectral library, i.e. MSL, data files include compound names and their spectral information. The spectral information consist of number of fragments for each compound, together with the mass and relative and usually normalized intensity of each fragment. Also, the retention time and retention index of the compounds are provided which helps improve the identification by avoding false positives. This functions is used to write the targets information into a file with MSL format.

Value

A logical value

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

`readMSL`, `getTarget`

Examples

```
# load targets information
data(Targets)

# write the targets into a file with MSL format
writeMSL(Targets = Targets, target.file.name = "myTargets.MSL")
```

writeResult *Write analysis results to file.*

Description

This funtion gets the anaylsis results and writes them in a csv format.

Usage

```
writeResult(runPeaks = list(), output.file.name = 'results.csv')
```

Arguments

runPeaks	a list including the analysis results and targets information, e.g. acquired by getPeak function.
output.file.name	a character object, i.e. string, of the name of the output file in csv format.

Details

As an input, the runPeaks object is a list of lists where each list, is the information for one run. The information for each run includes target information, and analysis results, e.g. apex and area location in time and their related intensities as well as estimated RI.

Value

A logical value

Author(s)

Mo R. Nezami Ranjbar

References

http://omics.georgetown.edu/SIMAT.html

See Also

readCDF, getPeak

Examples

```
# load an RData file including a single run data acquired by readCDF
data(Run)

# load targets information
data(Targets)

# get all the corresponding peaks of the target list
runPeaks <- getPeak(Run = Run, Targets = Targets)

# write analysis results into a csv file
writeResult(runPeaks = runPeaks)
```

Index