Package 'Autotuner'

September 29, 2021

Type Package

Title Automated parameter selection for untargeted metabolomics data processing

Version 1.7.0

Author Craig McLean

Maintainer Craig McLean <craigmclean23@gmail.com>

Description This package is designed to help faciliate data processing in untargeted metabolomics. To do this, the algorithm contained within the package performs statistical inference on raw data to come up with the best set of parameters to process the raw data.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0), methods, Biobase, MSnbase (>= 2.14.2)

RoxygenNote 7.1.1

biocViews MassSpectrometry, Metabolomics

Imports RColorBrewer, mzR, assertthat, scales, entropy, cluster, grDevices, graphics, stats, utils

Suggests testthat (>= 2.1.0), covr, devtools, knitr, rmarkdown, mtbls2

VignetteBuilder knitr

BugReports https://github.com/crmclean/Autotuner/issues

URL https://github.com/crmclean/Autotuner/

LazyData true

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

git_branch master

git_last_commit 87b3b61

git_last_commit_date 2021-05-19

Date/Publication 2021-09-29

R topics documented:

Autotuner-class *Autotuner*

Description

This file contains the skeleton to the Autotuner class used through out Autoutuner.

This object is a generic object designed to run the different functions of the Autotuner package. The slots represent content or data that the package uses throughout the different functions.

Slots

`time` A list containing vectors of scan time points from each sample.

`intensity` A list containing vectors of scan intensity points from each sample.

`peaks` Regions within each sample identified as peaks by slidingwindow analysis.

`peak_table` A data.frame containing information on each peak after further processing is done to the data.

`peak_difference` A data.frame containing information on how peaks are eluted differently over time.

`metadata` A data.frame containing metadata for all samples to be run on Autotuner.

`file_paths` A string path that leads to the samples to be run on Autotuner.

`file_col` A string for the column name of the column within the metadata that has specific sample names.

`factorCol` A string for the column name of the column within the metadata that has specific sample class names.

checkBounds *checkBounds*

Description

Recursive function used to find how far a binned feature might extend beyond the boundary of the originally defined TIC peak.

Usage

```
checkBounds(
  mass,
  upper = TRUE,
  mzDb,
  currentIndex,
  intensityStorage,
  ppmEst,
  scans,
  origBound,
  header
)
```

Arguments

mass	Specific mass being checked against adjacent scans.
upper	A boolean value that tells the algorithm to check indices greater than the entered one. If false, it will check values less thant the entered one.
mzDb	A list of data.frames containing the m/z and intensity values from each scan's mass spectra.
currentIndex	Numerical index indicating which scan contains feature specific information.
intensityStorage	A vector used during recursion to store intensity values as they are added to peak expansion.
ppmEst	Scalar numerical value meant to represent the ppm of the instrument.
scans	Set of all possible ms1 scans for the sample.
origBound	The original scan bound location of the peak.
header	A data.fame containing metadata on the sample like spectra type (MS1 vs MS2), retention time, and scan count.

Value

This function returns the last index the feature is detected.

---

checkEICPeaks *checkEICPeaks*

Description

This function is the outer most function used to check for individual EIC peak specific parameters.

Usage

```
checkEICPeaks(
  mzDb,
  header,
  observedPeak,
  massThresh = 0.005,
  useGap,
  varExpThresh,
  returnPpmPlots,
  plotDir,
  filename
)
```

Arguments

mzDb	A list of data.frames containing the m/z and intensity values from each scan's mass spectra.
header	A data.fame containing metadata on the sample like spectra type, retention time, and scan count.
observedPeak	A list with names 'start' and 'end' containing scalar values representing the calculated peak boundary points
massThresh	A generous exact mass error threshold used to estimate PPM for features.
useGap	Parameter carried into checkEICPeaks that tells Autotuner whether to use the gap statustic to determine the proper number of clusters to use during ppm parameter estimation.
varExpThresh	Numeric value representing the variance explained threshold to use if useGap is false.
returnPpmPlots	Boolean value that tells R to return plots for ppm distributions.
plotDir	Path where to store plots.
filename	A string containing the name of the current data file being analyzed.

Value

This function returns a peak specific set of processign parameters.

---

createAutotuner *createAutotuner*

Description

This function will create a Autotuner used to extract ms2s.

Usage

```
createAutotuner(data_paths, runfile, file_col, factorCol)
```

Arguments

data_paths	A string path pointing at data files to load in Autotuner.
runfile	A data.frame of sample metadata.
file_col	Character string of the column name of the column within the runfile that contains sample names.
factorCol	Character string of the column name of the column within the runfile that contains sample type factor.

Value

This function returns an Autotuner object

Examples

```
library(mtbls2)
rawPaths <- c(
system.file("mzData/MSpos-Ex2-cyp79-48h-Ag-1_1-B,3_01_9828.mzData",
package = "mtbls2"),
system.file("mzData/MSpos-Ex2-cyp79-48h-Ag-2_1-B,4_01_9830.mzData",
package = "mtbls2"),
system.file("mzData/MSpos-Ex2-cyp79-48h-Ag-4_1-B,4_01_9834.mzData",
package = "mtbls2"))

metadata <- read.table(system.file(
"a_mtbl2_metabolite_profiling_mass_spectrometry.txt", package = "mtbls2"),
header = TRUE, stringsAsFactors = FALSE)
metadata <- metadata[sub("mzData/", "",
metadata$Raw.Spectral.Data.File) %in% basename(rawPaths),]

Autotuner <- Autotuner::createAutotuner(rawPaths,
metadata,
file_col = "Raw.Spectral.Data.File",
factorCol = "Factor.Value.genotype.")
```

`dissectScans` *dissectScans*

Description

This function is designed to extract all MS1 scan features observed within the bounds of the current TIC peak.

Usage

```
dissectScans(mzDb, observedPeak, header)
```

Arguments

`mzDb`	This is a list of two column data frames containing information on each mass spectra within the data.
`observedPeak`	A list with 'start' and 'stop' boundaries of the current peak.
`header`	This is the header file containing all the metadata for the currently loaded sample.

Value

A Peak Matrix from Scan Data

EICparams *EICparams*

Description

This function is designed to calculate the recommended parameters from EIC peaks. It is the main holder function for a lot of different ones involved in calculating EIC parameters.

Usage

```
EICparams(
  Autotuner,
  massThresh,
  useGap = TRUE,
  varExpThresh = 0.8,
  returnPpmPlots = TRUE,
  plotDir = ".",
  verbose = TRUE
)
```

Arguments

Autotuner	An Autotuner objected containing sample specific raw data.
massThresh	A generous exact mass error threshold used to estimate PPM for features.
useGap	Parameter carried into checkEICPeaks that tells Autotuner whether to use the gap statustic to determine the proper number of clusters to use during ppm parameter estimation.
varExpThresh	Numeric value representing the variance explained threshold to use if useGap is false.
returnPpmPlots	A boolean value that tells R to return plots for ppm distributions.
plotDir	Path where to store plots.
verbose	Boolean value used to indicate whether checkEICPeaks function returns messages to the console.

Details

The function CheckEICPeaks handles all the peak specific computations.

Value

A data.frame of all peak specific estimates.

eicParamsEsts *eicParamsEsts*

Description

This object contains a data.frame of parameter estimates generated by running the Autotuner algorithm of the Autotuner object.

Usage

```
eicParamsEsts
```

Format

A data.frame representing the output of the EICparams function.

Source

inst/script/makeTestData.R

estimatePPM *estimatePPM*

Description

This function provides a convenient way to measure ppm between two exact masses.

Usage

```
estimatePPM(first, second)
```

Arguments

first	Numeric of length 1 respresenting the first mass entered
second	Numeric of length 1 representing second mass entered

Value

The ppm error between the first and second entered mass

estimateSNThresh *estimateSNThresh*

Description

This function is responsible for computing an estimated s/n threshold.

Usage

```
estimateSNThresh(no_match, sortedAllEIC, approvedPeaks)
```

Arguments

no_match	This is a vector of numerical indicies within the raw data mapping to scan data considered to come from noise.
sortedAllEIC	This is the raw data from a single TIC peak.
approvedPeaks	This is a data.frame that contains information on which peaks come from TIC data.

Value

returns an estimated s/n threshold value

---

extract_peaks *extract_peaks*

Description

This function is designed to extract peaks observed within the TIC from each sample, and return their indicies for further processing.

Usage

```
extract_peaks(Autotuner, returned_peaks = 10, signals)
```

Arguments

Autotuner	An Autotuner objected containing sample specific raw data.
returned_peaks	A scalar number of peaks to return for visual inspection. Five is the minimum possible value. is the standard.
signals	A list containing traces and locations where signals are detected across all samples being checked by the algorithm.

Value

peak_table_list - a list of data.frame tables containing information on where each where peaks are located within each sample.

filterPeaksfromNoise *filterPeaksfromNoise*

Description

This function is desgined to perform the binning of potential features. At this point in the algorithm, a potential feature is 2+ m/z values that are within the generous exact mass error windown provided by the user.

Usage

```
filterPeaksfromNoise(matchedMasses)
```

Arguments

matchedMasses	A data.frame containing information on the retained bins.

Value

A list with entries for noise peaks and true peaks.

filterPpmError *filterPpmError*

Description

This function computes an estimate for the ppm error threshold.

Usage

```
filterPpmError(
  approvedPeaks,
  useGap,
  varExpThresh,
  returnPpmPlots,
  plotDir,
  observedPeak,
  filename
)
```

Arguments

approvedPeaks	This is a data.frame with information on bins retained after filtering with user input mz error threshold and continuity checks.
useGap	Parameter carried into checkEICPeaks that tells Autotuner whether to use the gap statustic to determine the proper number of clusters to use during ppm parameter estimation.
varExpThresh	Numeric value representing the variance explained threshold to use if useGap is false.
returnPpmPlots	Boolean value that tells R to return plots for ppm distributions.
plotDir	Path where to store plots.
observedPeak	A list with names 'start' and 'end' containing scalar values representing the calculated peak boundary points
filename	A string containing the name of the current data file being analyzed.

Details

A distribution is created from the set of all ppm values identified. The most dense peak of this distribution is assumed to represent the standard ppm error of the data.

Value

This function returns a scalar value representing ppm error estimate.

---

findPeakWidth *findPeakWidth*

Description

This function is designed to find the maximum peakwidth of an EIC observed within a given TIC peak. It does so by using checkBounds to estimate width in time of a peak and countMaxima to determine if a peak may be made up from two similar structural isomers.

Usage

```
findPeakWidth(approvScorePeaks, mzDb, header, sortedAllEIC, boundaries, ppmEst)
```

Arguments

approvScorePeaks	A data.frame containing information on the retained bins.
mzDb	A list of data.frames containing the m/z and intensity values from each scan's mass spectra.
header	A data.fame containing metadata on the sample like spectra type (MS1 vs MS2), retention time, and scan count.

sortedAllEIC	A data.frame containing observed EIC values along with their corresponsing scan ID.
boundaries	A numeric vector with indicies representing the scans bounding the original TIC peak.
ppmEst	A scalar value representing the calculated ppm error used to generate data.

Value

This function returns a scalar value representing an estimate for the maximal peak width across samples.

`findTruePeaks` *findTruePeaks*

Description

This function is designed to filter out bins that don't come from continuous scans. The idea is that after this stage, the data is ready for parameter estimation.

Usage

```
findTruePeaks(truePeaks, sortedAllEIC)
```

Arguments

truePeaks	A list containing indicies representing each bin.
sortedAllEIC	All the raw ms1 data extracted from the EIC peak.

Value

a list of candidate EIC regions

`getAutoFactorCol` *getAutoFactorCol*

Description

This function returns the character string stored within the 'factorCol' slot of the Autotuner Object

Usage

```
getAutoFactorCol(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the factorCol slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoFactorCol(Autotuner)
```

---

getAutoFile_col *getAutoFile_col*

Description

This function returns the character string stored within the 'file_col' slot of the Autotuner Object

Usage

```
getAutoFile_col(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the file_col slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoFile_col(Autotuner)
```

---

getAutoFile_paths *getAutoFile_paths*

Description

This function returns the character string stored within the 'file_paths' slot of the Autotuner Object

Usage

```
getAutoFile_paths(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the file_paths slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoFile_paths(Autotuner)
```

---

getAutoIntensity *getAutoIntensity*

Description

This function is designed to return the list intensities obtained from applying the sliding window analysis to the raw data stored within an AutoTuner object.

Usage

```
getAutoIntensity(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the intensity slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoIntensity(Autotuner)
```

getAutoMetadata *getAutoMetadata*

Description

This function returns the data.frame stored within the 'meatadata' slot of the Autotuner Object

Usage

```
getAutoMetadata(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the meatadata slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoMetadata(Autotuner)
```

getAutoPeaks *getAutoPeaks*

Description

This function returns the list of numerics stored within the 'peaks' slot of the Autotuner Object

Usage

```
getAutoPeaks(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the peaks slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoPeaks(Autotuner)
```

getAutoPeak_difference *getAutoPeak_difference*

Description

This function returns the data.frame stored within the 'peak_difference' slot of the Autotuner Object

Usage

```
getAutoPeak_difference(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the peak_difference slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoPeak_difference(Autotuner)
```

getAutoPeak_table *getAutoPeak_table*

Description

This function returns the data.frame stored within the 'peak_table' slot of the Autotuner Object

Usage

```
getAutoPeak_table(Autotuner)
```

Arguments

Autotuner	An AutoTuner object.

Value

The content of the peak_table slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoPeak_table(Autotuner)
```

`getAutoTime` *getAutoTime*

Description

This function returns the list of numerics stored within the 'time' slot of the Autotuner Object

Usage

```
getAutoTime(Autotuner)
```

Arguments

`Autotuner`	An AutoTuner object.

Value

The content of the time slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoTime(Autotuner)
```

`initialize,Autotuner-method`
Update an `Autotuner` *object*

Description

This method updates an `Autotuner` object to the latest definition.

Usage

```
## S4 method for signature 'Autotuner'
initialize(.Object, data_paths, runfile, file_col, factorCol)
```

Arguments

.Object	The Autotuner object to update.
data_paths	A string path pointing at data files to load in Autotuner.
runfile	A data.frame of sample metadata.
file_col	Character string of the column name of the column within the runfile that contains sample names.
factorCol	Character string of the column name of the column within the runfile that contains sample type factor.

Value

An updated Autotuner containing all data from the input object.

Author(s)

Craig McLean

---

isolatePeaks *isolatePeaks*

Description

This function is designed to handle the isolation of TIC peak regions to throw into AutoTuner.

Usage

```
isolatePeaks(Autotuner, returned_peaks, signals)
```

Arguments

Autotuner	An Autotuner object. Ideally generated right after signal processing peak identification is complete.
returned_peaks	A numerical value representing the number of peaks that are to be returned to the user for downstream parameter optimization.
signals	A list of list containing data from sliding window analysis.

Value

Returns an Autotuner object with selected TIC regions.

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
lag <- 25
threshold<- 3.1
influence <- 0.1
signals <- lapply(getAutoIntensity(Autotuner), ThresholdingAlgo,
lag, threshold, influence)
isolatePeaks(Autotuner, returned_peaks = 10, signals)
```

`mzDb` *mzDb*

Description

A list containing all scan information from one of the mmetspData package files used for examples, tests, and vignettes in this package.

Usage

```
mzDb
```

Format

A list with length 926

Source

inst/script/makeAutotunerTestData.R

`observedPeak` *observedPeak*

Description

A list containing peak start and end times for one peak of the mmetspData package files used for examples, tests, and vignettes in this package.

Usage

```
observedPeak
```

Format

A list with length 2

start 56.18

end 97.73

Source

inst/script/makeAutotunerTestData.R

`peakwidth_est` *peakwidth_est*

Description

This function is designed to generate peak width estimates for each TIC peak detected by sliding window analysis.

Usage

```
peakwidth_est(
  peak_vector,
  time,
  intensity,
  start = NULL,
  end = NULL,
  old_r2 = NULL
)
```

Arguments

`peak_vector`	A numeric vector with names of specific time points of the chromatography data measured. The numeric values correspond to indiciess within the total chromatographic data that span the peak width.
`time`	This vector contains the time measurements during the chromatography. This vector is used to match the values in peak_vector to the names in the intensity vector.
`intensity`	A measured intensity values for chromatorgraphy
`start`	A numeric index indicating where peak starts. Leave null.
`end`	The same as above, leave null.
`old_r2`	A previous fit of model used to judge recursion of fit.

Details

This function takes in one peak vector at a time and runs a linear model on the selected start and end points of a peak. By measuring the change of the fit of the model, the function returns an index of values corresponding to a peak. This function works recursively to estimate the width of the peak. Ultimately, it returns the names of the final points in the peak.

Value

This function returns a scalar value representing the estimated peak width for a given peak.

---

peakwidth_table *peakwidth_table*

Description

This function is designed to generate estimates of peakwidth for each peak within the peakList and some properties of each peak. After this is done, the table of estimates is exported.

Usage

```
peakwidth_table(Autotuner, returned_peaks = 10)
```

Arguments

`Autotuner`	An Autotuner objected containing sample specific raw data.
`returned_peaks`	A scalar number of peaks to return for visual inspection. Five is the minimum possible value.

Details

The actual calculations used to estimate peakwidth are done within the function "peakwidth_est".

Value

This function will return a peak table with information on the peak width for each detected peak across samples, the name attribute for when the peak starts and ends, and the time points associated with each of those parameters and for the midpoint.

peak_time_difference *peak_time_difference*

Description

This function is designed to return a data.frame containing info on how

Usage

```
peak_time_difference(Autotuner)
```

Arguments

Autotuner	An Autotuner object containiing a table of peak width values extracted with the function peak_width_table.

Details

This function is designed to determine what are the retention time differences between peaks that are effectively the same between samples. The similarity in peaks is determined by a threshold in retention time similarity between peaks. This function returns the max peak width between samples, and the time difference between peaks across samples in a data frame object. The current and next row indexes are given to go back to the peaktable object to plot peaks.

Value

This function returns a data.frame of peaks matched over time.

plot_peaks @*title plot_peaks*

Description

@description This funciton plots the peak identified within chromatography.

Usage

```
plot_peaks(Autotuner, boundary = 10, peak, showLegend = TRUE)
```

Arguments

Autotuner	An Autotuner objected containing sample specific raw data.
boundary	UI input value that defines the boundary around the peak to visualize it.
peak	A Numeric index obtained from UI that indicates which peak should be visualized.
showLegend	A boolean dictating if a legend should be shown or not. Resolves issue where legend can cover chromatographic data

Details

This function plots individual peaks selected by signal processing and expanded with a regression to allow the user to validate the selected signal processing parameters.

Value

This function outputs plots that are meant to go into the peakVis UI.

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
plot_peaks(Autotuner = Autotuner, boundary = 100, peak = 1)
```

---

plot_signals *plot_signals*

Description

this funciton plots the peak identified within chromatography.

Usage

```
plot_signals(Autotuner, threshold, sample_index, signals)
```

Arguments

Autotuner	Autotuner object created following Create_Autotuner() initialization function.
threshold	User input scalar value for the number of standard deviations required to consider a peak to be significant.
sample_index	which of all of the samples should the user plot. Entered in as a numerical index value with length 1.
signals	A vector containing information on where signals are located between samples.

Details

This function plots the chromatography and the matching sliding window signal processing results for each sample. Signal processing functions will be the same color as the chromatography spectra, just a lighter shade and a different type of line. The chromatography will be a solid line, the signal will be a dashed line (lty = 2) with a .5 alpha, and the thersholds will be dotted lines (lty = 3) with alpha values of 3.

Value

Plots signal

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
lag <- 25
threshold <- 3.1
influence <- 0.1

signals <- lapply(getAutoIntensity(Autotuner),
ThresholdingAlgo, lag, threshold, influence)

plot_signals(Autotuner, threshold, sample_index = seq_len(3), signals = signals)
```

`returnParams` *returnParams*

Description

This function is designed to return a list of data.frames containing parameter estimates obtained from the EIC and TIC parameter estimation.

Usage

```
returnParams(eicParamEsts, Autotuner)
```

Arguments

`eicParamEsts`	The objection containing all parameter estimates obtained from running Autotuner's EICparam function.
`Autotuner`	An Autotuner object used to return the TIC estimated parameters

Value

A list of data.frames with parameter estimates.

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))

eicParamEsts <- readRDS(system.file("extdata/eicParamsEsts.rds",
package="Autotuner"))
outParams <- returnParams(eicParamEsts = eicParamEsts, Autotuner = Autotuner)
```

setAutoFactorCol *setAutoFactorCol*

Description

This function fills the factorCol slot within an Autotuner object.

Usage

```
setAutoFactorCol(factorCol, Autotuner)
```

Arguments

factorCol	A character vector representing factorCol
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled factorCol slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
factorCol <- getAutoFactorCol(Autotuner)
Autotuner <- setAutoFactorCol(factorCol, Autotuner)
```

setAutoFile_col *setAutoFile_col*

Description

This function fills the file_col slot within an Autotuner object.

Usage

```
setAutoFile_col(file_col, Autotuner)
```

Arguments

file_col	A character vector representing file_col
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled file_col slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
file_col <- getAutoFile_col(Autotuner)
Autotuner <- setAutoFile_col(file_col, Autotuner)
```

`setAutoFile_paths` *setAutoFile_paths*

Description

This function fills the file_paths slot within an Autotuner object.

Usage

```
setAutoFile_paths(file_paths, Autotuner)
```

Arguments

`file_paths`	A character vector representing file_paths
`Autotuner`	An AutoTuner object.

Value

An Autotuner object with a filled file_paths slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
file_paths <- getAutoFile_paths(Autotuner)
Autotuner <- setAutoFile_paths(file_paths = file_paths, Autotuner)
```

`setAutoIntensity` *setAutoIntensity*

Description

This function fills the "intensity" slot within an Autotuner object.

Usage

```
setAutoIntensity(intensity, Autotuner)
```

Arguments

intensity	A list of numeric values representing intensity
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled intensity slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
intensity <- getAutoIntensity(Autotuner)
Autotuner <- setAutoIntensity(intensity, Autotuner)
```

---

setAutoMetadata *setAutoMetadata*

Description

This function fills the metadata slot within an Autotuner object.

Usage

```
setAutoMetadata(metadata, Autotuner)
```

Arguments

metadata	A data.frame representing metadata
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled metadata slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
metadata <- getAutoMetadata(Autotuner)
Autotuner <- setAutoMetadata(metadata, Autotuner)
```

setAutoPeaks *setAutoPeaks*

Description

This function fills the peaks slot within an Autotuner object.

Usage

```
setAutoPeaks(peaks, Autotuner)
```

Arguments

peaks	A list of numeric values representing peaks
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled peaks slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
peaks <- getAutoPeaks(Autotuner)
Autotuner <- setAutoPeaks(peaks, Autotuner)
```

setAutoPeak_difference *setAutoPeak_difference*

Description

This function fills the peak_difference slot within an Autotuner object.

Usage

```
setAutoPeak_difference(peak_difference, Autotuner)
```

Arguments

peak_difference	A data.frame representing peak_difference
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled peak_difference slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
peak_difference <- getAutoPeak_table(Autotuner)
Autotuner <- setAutoPeak_difference(peak_difference, Autotuner)
```

---

setAutoPeak_table *setAutoPeak_table*

Description

This function fills the peak_table slot within an Autotuner object.

Usage

```
setAutoPeak_table(peak_table, Autotuner)
```

Arguments

peak_table	A data.frame representing peak_table
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled peak_table slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
peak_table <- getAutoPeak_table(Autotuner)
Autotuner <- setAutoPeak_table(peak_table, Autotuner)
```

setAutoTime *setAutoTime*

Description

This function fills the "time" slot within an Autotuner object.

Usage

```
setAutoTime(time, Autotuner)
```

Arguments

time	A list of numeric values representing time
Autotuner	An AutoTuner object.

Value

An Autotuner object with a filled time slot

Examples

```
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
time <- getAutoTime(Autotuner)
Autotuner <- setAutoTime(time, Autotuner)
```

ThresholdingAlgo *ThresholdingAlgo*

Description

This function performs a sliding window analysis on the chromatograms in order to identify peaks within the data. I would recommend to keep influence low in order to use adjacent peak lengths as a measure of peak width.

Usage

```
ThresholdingAlgo(y, lag, threshold, influence)
```

Arguments

y	A numerical vector of measured chromatographic intensity values
lag	A scalar value of number of observations to calculate intensity prior to peak selection.
threshold	A number of standard deviations above chromatogram. Used to detect significantly observed peaks.
influence	A scalar values between 0-1 that describes how much the value of a peak (measured index value above threshold) should contribute to the sliding window analysis of downstream peaks.

Value

A list of calculated sliding window values.

Examples

```
lag <- 25
threshold<- 3.1
influence <- 0.1
Autotuner <- readRDS(system.file("extdata/Autotuner.rds",
package="Autotuner"))
signals <- lapply(getAutoIntensity(Autotuner), ThresholdingAlgo,
lag, threshold, influence)
```

TIC_params *TIC_params*

Description

This function is designed to return the parameters related to chromatography gathered during the TIC peak selection steps. An estimate is given for maximum and minimum peak width as well as the bandwidth parameter used in grouping.

Usage

```
TIC_params(peak_table, peak_difference)
```

Arguments

peak_table	A data.frame containing information on peak width values extracted with the function peak_width_table.
peak_difference	A data.frame containing information on retention time differences between peaks.

Value

Returns a set of parameters to run xcms.

Index