Package ‘rMSIproc’

February 13, 2020

Type Package

Title MSI Processing in C++

Version 0.3

Date 2018-03-16

Author Pere Rafols[aut], Lluc Semente [aut]

Maintainer Pere Rafols Soler<`pere.rafols@urv.cat`>

Description Processing methods for MSI using rMSI data format and C++ methods

License GPL-3

LazyData TRUE

SystemRequirements C++11, fftw3 (>= 3.3.5)

Imports rMSI,
Rcpp (>= 0.12.2),
gWidgets2 (>= 1.0),
RGtk2 (>= 2.20.25),
gWidgets2RGtk2 (>= 1.0)

Suggests ggplot2 (>= 3.0.0),
gridExtra (>= 2.3)

LinkingTo Rcpp

RoxygenNote 7.0.2

R topics documented:

`adductAnnotation` *adductAnnotation*

Description

Given the monoisotopic ions found by the isotopes test, founds the possible adduct pairs considering the elements in the adductDataFrame.

Usage

```
adductAnnotation(isotopeObj, PeakMtx, adductDataFrame, tolerance)
```

Arguments

`isotopeObj`	List. Results from the isotopes test.
`PeakMtx`	List. An rMSIprocPeakMatrix. Must contain at least the following categories: • PeakMtx$intensity. A matrix containg the intensities of the peaks for each pixel (rows = pixels, cols = peaks). • PeakMtx$mass. A vector containg the masses of each peak. Must be in the same order with the columns of the intensity marix. • PeakMtx$numPixels. Number of pixels (rows in your matrix).
`adductDataFrame`	Data frame with two columns. $name, with the names of adducts to be found, $mass, with each masses.
`tolerance`	Integer. Mass error tolerance in ppm.

Value

A list containing the results of the test and other kind of information.

ArrangeMultipleImg2Plot *ArrangeMultipleImg2Plot*

Description

Prepare multiple images to be plotted with arbitrary data that must be specified as a vector using the values parameter. The values vector must be sorted according the pos matrix of the provided peak matrix object (peakMat). The images will be plotted laidout in a matrix according the parameters nrow, ncol and byrow which follows the same conventions as the R matrix() function. A different order of the images can be specified with the img_name parameter. Just provide the images names to plotted in the desierd order. Additionally each single image can be flip horizontally and vertically and/or rotated to any angle using the parameters mirror_x, mirror_y and rotations.

Usage

```
ArrangeMultipleImg2Plot(
  peakMat,
  values,
  nrow,
  ncol,
  byrow = T,
  margin = 20,
  img_names = peakMat$names,
  rotations = rep(0, length(img_names)),
  mirror_x = rep(F, length(img_names)),
  mirror_y = rep(F, length(img_names))
)
```

Arguments

`peakMat`	an rMSIproc peak matrix.
`values`	a vector of pixel values following the same order as pixels appear in the peak matrix.
`nrow`	number of rows of the plotted matrix layout (only used if byrow == F)
`ncol`	number of cols of the plotted matrix layout (only used if byrow == T)
`byrow`	a bool specifing if images must be arranged in rows.
`margin`	the separation between plotted images.
`img_names`	a character vector with img names in the desierd plotting order.
`rotations`	a vector with the rotation of each images specified in degrees.
`mirror_x`	a bool vector specifing if each image must be flipped or not in X direction prior to rotation.
`mirror_y`	a bool vector specifing if each image must be flipped or not in Y direction prior to rotation.

Value

a list object conaining the arranged position matrix, the pixel values and the labels positions.

`AverageSpectrum` *AverageSpectrum.*

Description

Calculates the dataset average spectrum. The average spectrum is the spectrum produced by weighting all the dataset spectra.

Usage

```
AverageSpectrum(img, NumOfThreads = parallel::detectCores())
```

Arguments

`img`	An rMSI object.
`NumOfThreads`	Number of threads.The Default value is the number of cores of the machine.

Value

The average spectrum.

`buildImgIdVectorFromPeakMatrix` *buildImgIdVectorFromPeakMatrix.*

Description

Builds a integer vector containing all rMSI objects ID's accroding the peak matrix row order. The resulting ID vector can be used to locate a spectrum ID in a peak matrix of multiple data sets.

Usage

```
buildImgIdVectorFromPeakMatrix(pkMat)
```

Arguments

`pkMat`	an rMSIproc peak matrix object.

Value

an integer vector with all ID's of each image in the pkMat object.

`CalibrateImage` *Apply a new mass axis to a complete image. A GUI to calibrate the mean spectrum will be shown.*

Description

Apply a new mass axis to a complete image. A GUI to calibrate the mean spectrum will be shown.

Usage

```
CalibrateImage(img, output_fname, newMzAxis = NULL)
```

Arguments

`img`	A image in rMSI data format .
`output_fname`	full path to store the output image.
`newMzAxis`	if a new mz axis specified it is used for the calibration window
`useZoo`	if zoo interpolation must be used (caution this may introduce m/z error if large compensations are made)

`CalibrationWindow` *CalibrationWindow.*

Description

CalibrationWindow.

Usage

```
CalibrationWindow(
  mass,
  intensity,
  peak_win_size = 20,
  win_title = "",
  CalibrationSpan = 0.75,
  NotPerformCalibration = F
)
```

Arguments

`mass`	The mass vector of spectrum to calibrate.
`intensity`	The intensity vector of spectrum to calibrate.
`CalibrationSpan`	the span of the loess method for calibration.
`NotPerformCalibration`	a boolean to disbale calibration. The list of target masses will be returned instead.

Value

a the calibrated mass axis.

`calMzAxis` *calMzAxis.*

Description

calMzAxis.

Usage

```
calMzAxis(avgSpc_mz, ref_mz, target_mz, method = "loess", CalSpan = 0.75)
```

Arguments

`avgSpc_mz`	The mass axis to calibrate.
`ref_mz`	a vector of reference masses (for exaple the theorical gold peaks).
`target_mz`	manually slected masses to be fittet to ref_masses (must be the same length than ref_mz).
`method`	a string with the method used for interpolation, valid methods are loess and linear
`CalSpan`	the span of loess method.

Value

a list containing the calibrated mass axis and the interpolated mass error repect the original mass axis.

`chemFormula2Expression` *chemFormula2Expression.*

Description

Converts a string containing a chemical forumla to an R expression that allows plotting the formula with sub-indices properly.

Usage

```
chemFormula2Expression(strFormula)
```

Arguments

`strFormula`	a string containing a chemical formula.

Value

a formated R expression that must be used with parse().

`CPeakList2PeakMatrix` *CPeakList2PeakMatrix.*

Description

Convert's an R peak list into a peak matrix.

Usage

```
CPeakList2PeakMatrix(
  RpeakList,
  BinTolerance = 5,
  BinFilter = 0.1,
  BinToleranceUsingPPM = TRUE
)
```

Arguments

`RpeakList`	R peak list.
`BinFilter`	the peaks bins non detected in at least the BinFitler*TotalNumberOfPixels spectra will be deleted.
`BinToleranceUsingPPM`	if True the peak binning tolerance is specified in ppm, if false the tolerance is set using scans.
`the`	tolerance used to merge peaks to the same bin. It is recomanded to use the half of peak width in ppm units.

Value

peak matrix.

`DeisotopingOutputFormat` *DeisotopingOutputFormat*

Description

Gives format to the output.

Usage

```
DeisotopingOutputFormat(r, ScoreThreshold)
```

`DetectPeaks` *DetectPeaks'*

Description

DetectPeaks'

Usage

```
DetectPeaks(mass, intensity, SNR = 5, WinSize = 20, OverSampling = 10)
```

Arguments

`mass`	a vector containing the mass axis.
`intensity`	a vector containinf the spectrum intensities.
`SNR`	the minimum signal to noise ratio of retained peaks
`WinSize`	the used windows size for peak detection
`OverSampling`	the used oversampling value for interpolating the peak shape and improve mass and area calculation.

Value

a list containing mass, intensity, SNR, area and the binSize arround peak fields of detected peaks.

`DetectPeaks_C` *DetectPeaks_C.*

Description

Detect peaks from a Rcpp::NumericVector object and returns data in a R matrix. This method is only exported to be use by R function DetectPeaks which is an actual R function. The returned peak positions follows C indexing style, this is starts with zero.

Usage

```
DetectPeaks_C(mass, intensity, SNR = 5, WinSize = 20L, UpSampling = 10L)
```

Arguments

`mass`	a NumericVector containing the mass axis of the spectrum.
`intensity`	a NumericVector where peaks must be detected.
`SNR`	Only peaks with an equal or higher SNR are retained.
`WinSize`	The windows used to detect peaks and caculate noise.
`UpSampling`	the oversampling used for acurate mass detection and area integration.

Value

a NumerixMatrix of 5 rows corresponding to: mass, intensity of the peak, SNR, area and binSize.

`ExportROIAverages` *ExportROIAverages.*

Description

Exports the average spectrum of a group of pixels in a ROI as an ASCII text file. ROI's are defined using Bruker XML files. A single spectrum will be exported for each ROI found in XML file.

Usage

```
ExportROIAverages(roi_xml_file, img, out_path = getwd(), normalization = NULL)
```

Arguments

`roi_xml_file`	a Bruker region XML file.
`img`	an rMSI object.
`out_path`	disk path where the results will be stored.
`normalization`	a text string with the name of desired normalization to apply at output files.

Value

a list with the data set UUID and the pixels ID's exported for each ROI.

`ExportROIAveragesMultiple` *ExportROIAveragesMultiple.*

Description

Exports the ROI average spectrum for a data set of multiple images. If pkMat != NULL the CSV summary peak matrices will be exported as well.

Usage

```
ExportROIAveragesMultiple(
  img_list,
  xml_list,
  pkMat = NULL,
  out_path = getwd(),
  normalization = NULL
)
```

Arguments

`img_list`	a lis of rMSI objects.
`xml_list`	a vector of XML ROI files in the same order as img_list.
`pkMat`	an rMSIproc peak matrix object.
`out_path`	disk path where the results will be stored.
`normalization`	a text string with the name of desired normalization to apply at output files.

`ExportROIPeaks` *ExportROIPeaks.*

Description

Export a peak matrix summary as CSV files. An average and standard deviation of each ROI is calculated for intensity, area and SNR peak matrices. The results are stored as tables in CSV files.

Usage

```
ExportROIPeaks(pkmat, idlist, out_path = getwd(), normalization = NULL)
```

Arguments

`pkmat`	an rMSIproc peak matrix object.
`idlist`	a list of pixel ID's in each ROI using the format returned by ExportROIAverages.
`out_path`	disk path where the results will be stored.
`normalization`	a text string with the name of desired normalization to apply at output files.

`export_imzMLpeakList` *export_imzMLpeakList.*

Description

Export an rMSIproc peak list as an imzML processed dataset.

Usage

```
export_imzMLpeakList(
  peakList,
  posMatrix,
  pixel_size_um,
  filename,
  normalization = rep(1, length(peakList))
)
```

Arguments

`peakList`	an rMSIproc peak list object.
`posMatrix`	an rMSI pos matrix with the pixel coordinates in the image.
`pixel_size_um`	the image pixel size in microns.
`filename`	complete path where the imzML and ibd files will be stored (the .imzML extensions must be omited).
`normalizations`	a numeric vector contaning the normalization value for each pixel (1 by default).

`FormatPeakMatrix` *FormatPeakMatrix. Formats a C style peak matrix generated by MTPeakPicking::BinPeaks() to a rMSIprocPeakMatrix. If the original motors matrix posMotors is not provided, a copy of posMat will be used.*

Description

FormatPeakMatrix. Formats a C style peak matrix generated by MTPeakPicking::BinPeaks() to a rMSIprocPeakMatrix. If the original motors matrix posMotors is not provided, a copy of posMat will be used.

Usage

```
FormatPeakMatrix(cPeakMatrix, posMat, numPixels, names, uuid, posMotors = NULL)
```

Arguments

`cPeakMatrix`	a peak matrix with the same format as retured by MTPeakPicking::BinPeaks().
`posMat`	a rMSI image pos matrix.
`numPixels`	a vector including the number of pixels of each sample.
`names`	a vector of strings with the name of each sample.
`uuid`	a vector of img UUID to be also stored in peak matrices
`posMotors`	a rMSI image original motros coordinates matrix.

Value

the formated matrix.

`getImgIdsFromPeakMatrixRows` *getImgIdsFromPeakMatrixRows.*

Description

Calculate the rMSI object ID's corresponding to a subset of row of a rMSIproc peak matrix.

Usage

```
getImgIdsFromPeakMatrixRows(pkMat, rows)
```

Arguments

`pkMat`	an rMSIproc peak matrix object.
`rows`	a vector of peak matrix rows.

Value

a list with the rMSI object ID's and image that correpond the specified rows.

`getPeakMatrixRowsFromImgIds` *getPeakMatrixRowsFromImgIds.*

Description

Obtains a vector of rMSIproc peak matrix rows corresponding to a vector of rMSI obj ID's.

Usage

```
getPeakMatrixRowsFromImgIds(pkMat, img_num, ids)
```

Arguments

`pkMat`	an rMSIproc peak matrix object.
`img_num`	the number of the data set object to select in pkMat.
`ids`	a vector of rMSI obj ID's.

Value

a vector containing the rows of peak matrix that correspond to the selected rMSI object ID's.

`getrMSIdataInfo` *getrMSIdataInfo.*

Description

Obtains all storing information of an rMSI object and returns it as a list.

Usage

```
getrMSIdataInfo(img)
```

Arguments

`img`	an rMSI data object.

Value

a list containing all storing information.

`getrMSIdataInfoMultipleDataSets` *getrMSIdataInfoMultipleDataSets.*

Description

Obtains all storing information of various rMSI objects and returns them unified in a list. The supplied rMSI objects must have the same number of mass channels and the same data type. Otherwise an error will be raised. The returned list contains an extra file "datasets" to indecate at which dataset belongs each ramdisk.

Usage

```
getrMSIdataInfoMultipleDataSets(imgs_list)
```

Arguments

`imgs_list`	a list of rMSI objects.

Value

a list containing all storing information unified.

hello *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

import_imzMLpeakList *import_imzMLpeakList.*

Description

import a processed imzML file to a rMSIproc peak list object.

Usage

```
import_imzMLpeakList(
  imzML_File,
  ibd_File = paste(sub("\\.[^.]*$", "", imzML_File), ".ibd", sep = "")
)
```

Arguments

imzML_File	full path to .imzML file.
ibd_File	path to the binary file (default the same as imzML file but with .ibd extension)

Value

a list containing: the peakList, the rMSI formated position matrix and the pixel size.

InternalReferenceSpectrum *InternalReferenceSpectrum.*

Description

Calculates the dataset reference spectrum to use in label free alignment. The reference spectrum is the spectrum in the dataset with the best correlation to average spectrum and with high TIC.

Usage

```
InternalReferenceSpectrum(img, reference = img$mean)
```

Arguments

`img`	MSI image to calculate internal reference spectrum.
`reference`	the spectrum to use as reference for correlations.

Value

a list with the intensity vector corresponding to the reference spectrum, the score and the pixel ID selected as reference.

InternalReferenceSpectrumMultipleDatasets *InternalReferenceSpectrumMultipleDatasets.*

Description

Calculates the dataset reference spectrum to use in label free alignment. The reference spectrum is the spectrum in the dataset with the best correlation to average spectrum and with high TIC.

Usage

```
InternalReferenceSpectrumMultipleDatasets(img_list)
```

Arguments

`img_list`	a list of various rMSI objects.

Value

a list with the intensity vector corresponding to the reference spectrum, the score, the image index which contains the refernce spectrum and the pixel ID selected as reference.

isotopeAnnotation *isotopeAnnotation*

Description

Finds and evaluate isotope candidates for each ion mass.

Usage

```
isotopeAnnotation(
  PeakMtx,
  isoNumber = 2,
  tolerance = 30,
  scoreThreshold = 0.8,
  toleranceUnits = "ppm",
  imageVector = NULL
)
```

Arguments

`PeakMtx`	List. An rMSIprocPeakMatrix. Must contain at least the following categories: • PeakMtx$intensity. A matrix containg the intensities of the peaks for each pixel (rows = pixels, cols = peaks). • PeakMtx$mass. A vector containg the masses of each peak. Must be in the same order with the columns of the intensity marix. • PeakMtx$numPixels. Number of pixels (rows in your matrix).
`isoNumber`	Integer. Number of isotopes to be found.
`tolerance`	Integer. Mass tolerance for the candidates in scans or ppms.
`scoreThreshold`	Numeric. Score value to consider a ion mass a good isotope candidate. Only the ions that have this number or greater will undergo the following isotope searching stages.
`toleranceUnits`	String. Must be 'ppm' or 'scan'. If ToleranceUnits is 'scan' then ImageVector must be the mass channels vector of the rMSI image (rMSIObj$mass).
`imageVector`	Numeric Vector. The mass channels vector of the imaging dataset containing all the scans.

Value

A list containing the results of the test and other kind of information.

`loadDataCube` *loadDataCube.*

Description

Loads an rMSI data cube (aka. ramdisk file) to an R matrix using C++ backend. This method is just for testing convinience and should not be used because it copies a matrix indexed by rows to a matrix indexed by columns so it performs very slowly. Instead of using this method use rMSI::loadImgCunckFromCube.

Usage

```
loadDataCube(img, cubeSel)
```

Arguments

`img`	and rMSI object image.
`cubeSel`	the cube to load R index.

Value

an R matrix with spectra in rows.

`LoadPeakMatrix` *LoadPeakMatrix.*

Description

Loads a binned peaks matrix from HDD.

Usage

```
LoadPeakMatrix(data_path)
```

Arguments

`data_path`	full path to zip file where data is stored.

Value

an R List containing intensity, SNR and area matrices, mass axis vector and if available the normalizations data.frame.

`LoadPeakMatrixC` *LoadPeakMatrix.*

Description

Loads a binned peaks matrix from HDD.

Usage

```
LoadPeakMatrixC(path)
```

Arguments

`path`	full path to directory from where data must be loaded.

Value

an R List containing intensity, SNR and area matrices, mass axis vector and if available the normalizations data.frame.

`MergePeakMatrices` *MergePeakMatrices.*

Description

Merges a list containing various peak matrices in a single peak matrix. The rMSIproc binning method is used to calculate the new masses.

Usage

```
MergePeakMatrices(PeakMatrixList, binningTolerance = 100, binningFilter = 0.01)
```

Arguments

`PeakMatrixList`	A list of various peak matrix objexts produced using rMSIproc.
`binningTolerance`	the tolerance used to merge peaks to the same bin specified in ppm. It is recomanded to use the half of the peak width.
`binningFilter`	the peaks bins non detected in at least the BinFitler*TotalNumberOfPixels spectra will be deleted.

Value

a intensity matrix where each row corresponds to an spectrum.

`MergerMSIDataSets` *MergerMSIDataSets.*

Description

Merges various rMSI objects in order to process all of them in the same run. For example, this is usefull to align data form diferent experiments together. In case that images don't share the same mass axis, all of them will be resampled and stored in a new ramdisk. If pixel_id is provided, a new ramdisk for each image will be created to filter out non specified pixel ID's. Discarding pixels of a dataset may result interesting to avoid artifacts in alignment and peak binning if some regions are not well-correlated to the rest of tissue.

Usage

```
MergerMSIDataSets(img_list, ramdisk_path, pixel_id = NULL)
```

Arguments

`img_list`	a list of images to be merged.
`ramdisk_path`	a path where resampled data ramdisk will be stored.
`pixel_id`	a list containing a vector of ID's to retain for each img. If some img have to use all ID's 0 may be supplied. Th ID's must be sorted in ascending order.

Value

a list with the merged images.

`NoiseEstimationFFTCosWin` *NoiseEstimationFFTCosWin.*

Description

Estimate the noise of a spectrum using a FFT filter and a cosinus window in frequency domain.

Usage

```
NoiseEstimationFFTCosWin(x, filWinSize = 40L)
```

Arguments

`x`	an Rcpp::NumericVector containing the spectrum intensities.
`filWinSize`	an integer specified the cosinus win size in samples.

Value

an Rcpp::NumericVector containing the estimated noise.

`NoiseEstimationFFTCosWinMat`
NoiseEstimationFFTCosWinMat.

Description

Estimate the noise of some spectra using a FFT filter and a cosinus window in frequency domain.

Usage

```
NoiseEstimationFFTCosWinMat(x, filWinSize = 40L)
```

Arguments

`x`	an Rcpp::NumericMatrix containing the spectra intensities. Each spectrum in a row.
`filWinSize`	an integer specified the cosinus win size in samples.

Value

an Rcpp::NumericMatrix containing the estimated noise in a matrix where each spectrum is a row.

`NoiseEstimationFFTExpWin`
NoiseEstimationFFTExpWin.

Description

Estimate the noise of a spectrum using a FFT filter and a decay exponential window in frequency domain.

Usage

```
NoiseEstimationFFTExpWin(x, filWinSize = 40L)
```

Arguments

`x`	an Rcpp::NumericVector containing the spectrum intensities.
`filWinSize`	an integer specified the cosinus win size in samples.

Value

an Rcpp::NumericVector containing the estimated noise.

NoiseEstimationFFTExpWinMat *NoiseEstimationFFTExpWinMat.*

Description

Estimate the noise of some spectra using a FFT filter and a decay exponential window in frequency domain.

Usage

```
NoiseEstimationFFTExpWinMat(x, filWinSize = 40L)
```

Arguments

`x`	an Rcpp::NumericMatrix containing the spectra intensities. Each spectrum in a row.
`filWinSize`	an integer specified the cosinus win size in samples.

Value

a

peakAnnotation *peakAnnotation*

Description

Searches for isotopic ions in the peak matrix and evaluates them using morphology, intensity and mass error criteria. This algorithm only searches carbon-based isotopic ions. Using the infromation generated by the isotopes test, the algorithm then proceeds to search adduct pairs between the monoisotopic ions.

Usage

```
peakAnnotation(
  PeakMtx,
  iso.number = 2,
  iso.tolerance = 30,
  iso.scoreThreshold = 0.75,
  iso.toleranceUnits = "ppm",
  iso.imageVector = NULL,
  add.tolerance = 50,
 add.adducts = data.frame(mass = c(38.963706, 22.98976, 1.007825), name = c("K", "Na",
    "H"))
)
```

Arguments

`PeakMtx` List. An rMSIprocPeakMatrix. Must contain at least the following categories:

- PeakMtx$intensity. A matrix containg the intensities of the peaks for each pixel (rows = pixels, cols = ions).
- PeakMtx$mass. A vector containg the masses of each peak. Must be in the same order with the columns of the intensity marix.
- PeakMtx$numPixels. Number of pixels (rows in your matrix).

`iso.number` Integer. Number of isotopes to be found.

`iso.tolerance` Integer. Mass tolerance for the isotope candidates in scans or ppms.

`iso.scoreThreshold` Numeric. Score value to consider a ion mass a good isotope candidate. Only the ions that have this number or greater will undergo the following isotope searching stages.

`iso.toleranceUnits` String. Must be 'ppm' or 'scan'. If ToleranceUnits is 'scan' then ImageVector must be the mass channels vector of the rMSI image (rMSIObj$mass).

`iso.imageVector` Numeric Vector. The mass channels vector of the imaging dataset containing all the scans.

`add.tolerance` Integer. Mass error tolerance in ppm.

`add.adductDataFrame` Data frame with two columns.

- $name. Column containing the names as strings of the elements or molecules that form adducts
- $mass. Masses of the adduct forming elements.

Value

A list of lists. All the infomarion related with the isotopes can be found in $isotopes sub-list, and the same for adducts in the $adducts sub-list.

- $isotopes: List of lists. Contains the followin sub-lists:
  - $Mn: List of matrices. Contains all the M+n ions evaluated and the results matrix. The name of the lists referes to the peak selected as monoisotopic.
  - $isotopicPeaks: Vector. Indices of the mass vector beloning to isotopic peaks.
  - $monoisotopicPeaks: Vector. Indices of the mass vector beloning to monoisotopic peaks.
- $adducts: List of matrices. Contains all the adduct pairs found during the test. The names refer to the neutral masses of the hypothetic ions.

PeakList2PeakMatrix *PeakList2PeakMatrix.*

Description

Convert's an R peak list into a peak matrix.

Usage

```
PeakList2PeakMatrix(
  PeakList,
  PosMatrix,
  BinTolerance = 5,
  BinToleranceUsingPPM = T,
  BinFilter = 0.1,
  imgUUID = rMSI:::uuid_timebased(),
  imgName = "rMSIproc_PeakList2PeakMatrix_peakMat"
)
```

Arguments

`PeakList`	R peak list. Missing values will be set to zero.
`PosMatrix`	the pos matrix as following the same format as in rMSI object.
`BinTolerance`	tolerance used to merge peaks to the same bin. It is recomanded to use the half of peak width in ppm units.
`BinToleranceUsingPPM`	if True the peak binning tolerance is specified in ppm, if false the tolerance is set using scans.
`BinFilter`	the peaks bins non detected in at least the BinFitler*TotalNumberOfPixels spectra will be deleted.
`imgUUID`	a unique ID for the peak matrix, if not provided it will be automatically generated.
`imgName`	the name of the original MSI dataset.

Value

peak matrix.

Examples

```

  #Import an imzML using centroid mode (after peak picking).
  pkLst <- rMSIproc::import_imzMLpeakList("~/path/to/processed/data/file.imzML")

  #Run the peak-binning to get the peak matrix.
 myPkMat <- rMSIproc::PeakList2PeakMatrix(pkLst$peakList, pkLst$pos, BinTolerance = 25, BinToleranceUsingPPM = T)
```

```
#Save the peak matrix.
rMSIproc::StorePeakMatrix("~/path/to/file.zip", myPkMat)
```

plot.rMSIprocPeakMatrix *Generic plot method for rMSIproc peak matrix.*

Description

Generic plot method for rMSIproc peak matrix.

Usage

```
## S3 method for class 'rMSIprocPeakMatrix'
plot(x, values, method = "mz", use_ggplot = FALSE)
```

Arguments

x	rMSIproc peak matrix object.
values	the values used by the plot. The behaviour of this parameter is controlled by the 'method' argument.
method	a method used by the plot. Available options are: "mz", "values" and "clusters".
use_ggplot	a boolean specifing if a ggplot2 backed must be used for plotting.

Details

This generic plot method allows to create different graphics from an rMSIproc peak matrix. The plot type is controlled by the 'method' argument with the following option:

- **mz**: produce an ion map using the 'values' argument as the target m/z channel to display (this is the default behaviour).
- **values**: produce an image using the values given in the 'values' argument as pixel intensities. This method is useful to display the results of user's calculation directly over the image.
- **clusters**: the 'values' argument contains a vector of integers indicating at which cluster belong each pixel in the image. This method produce an image automatically coloured according the clusters vector. The colours codes are returned to be reused in further graphics.

Examples

```
#For the following example we will load an rMSIproc peak matrix in the pks variable:
pks <- rMSIproc::LoadPeakMatrix("/path/to/my/peak/matrix.zip")

#Plot the m/z 848.7 distribution:
plot(pks, 848.7)
```

```
#Plot the TIC value of each pixel using the 'values' method:
plot(pks, values = pks$normalizations$TIC, method = "values")

#Cluster the peak matrix using kmeans and display each cluster on the image:
clus <- kmeans(pks$intensity/pks$normalizations$TIC, centers = 5)
plot(pks, clus$cluster, method = "clusters")
```

`plotClusterImage` *plotClusterImage.*

Description

Plot a segmentation image with the user-given clusters.

Usage

```
plotClusterImage(
  peakMatrix,
  clusters,
  nrow = 2,
  ncol = 2,
  byrow = T,
  margin = 20,
  img_names = peakMatrix$names,
  labels = img_names,
  rotations = rep(0, length(img_names)),
  mirror_x = rep(F, length(img_names)),
  mirror_y = rep(F, length(img_names)),
  pixel_size_um = 100
)
```

Arguments

`peakMatrix`	the peak matrix in an rMSIproc object.
`clusters`	a vector with integer number according the cluster of each pixel.
`nrow`	number of rows of the plotted matrix layout (only used if byrow == F)
`ncol`	number of cols of the plotted matrix layout (only used if byrow == T)
`byrow`	a bool specifing if images must be arranged in rows.
`margin`	the separation between plotted images.
`img_names`	a character vector with img names in the desierd plotting order.
`labels`	an alternative character vector with the labels to be displayed for each image.
`rotations`	a vector with the rotation of each images specified in degrees.
`mirror_x`	a bool vector specifing if each image must be flipped or not in X direction prior to rotation.

`mirror_y`	a bool vector specifing if each image must be flipped or not in Y direction prior to rotation.
`pixel_size_um`	the pixel resolution in um.

Value

a vector with the used color for each cluster sorted according clustering numering in assending order.

`plotClusterImageG` *plotClusterImageG.*

Description

This function is just for convenience since calling plotValuesImageG with a factor produces the same results. This function is equivalent to rMSIproc::plotClusterImage() but using the ggplot2.

Usage

```
plotClusterImageG(
  peakMatrix,
  clusters,
  plot_rows = 2,
  plot_cols = 2,
  plot_byrow = T,
  plot_rotations = rep(0, length(peakMatrix$names)),
  plot_mirror_X = rep(F, length(peakMatrix$names)),
  plot_mirror_Y = rep(F, length(peakMatrix$names)),
  plot_margin = 40,
  plot_labels = peakMatrix$names,
  title_label = "",
  fixed_aspect_ratio = F
)
```

Arguments

`peakMatrix`	an rMSIproc peak matrix.
`clusters`	a vector with integer number according the cluster of each pixel.
`plot_rows`	number of rows to arrange multiple images in the plotting area.
`plot_cols`	number of columns to arrange multiple images in the plotting area.
`plot_byrow`	a boolean idicating if the plotted images must be sorted by rows.
`plot_rotations`	a vector with the rotation in degree to apply to each image.
`plot_mirror_X`	a vector of booleans idicatinc if each image must be flipped horizontally.
`plot_mirror_Y`	a vector of booleans idicatinc if each image must be flipped vertically.
`plot_margin`	a numeric value that determines the separation between images.

`plot_labels`	text labels to be used for each image.
`title_label`	Text label for the plot main title.
`fixed_aspect_ratio`	set this flag to true to fix the aspect ratio of the ion images.

Value

a ggplot2 object.

`plotMassDriftComparedG` *plotMassDriftCompared.*

Description

Creates a plot in a grid to compare the spectral alignment of an ion before and after the rMSIproc processing. For a given mass this function will display in a single figure: - The ion map. - The alignment before the processing (RAW). - The alignment after the processing (Aligned).

Usage

```
plotMassDriftComparedG(
  peakMatrix_RAW,
  peakMatrix_ALNG,
  peaklist_RAW,
  peaklist_ALNG,
  mass,
  error_range_ppm = 400,
  min_SNR = 10,
  mass_offset_RAW = 0,
  title_label = "",
  normalization_RAW = NA,
  normalization_ALNG = NA,
  ion_map_plot_cols = 2,
  ion_map_plot_rows = 2,
  ion_map_plot_byrow = T,
  ion_map_plot_rotations = rep(0, length(peakMatrix_ALNG$names)),
  ion_map_plot_labels = peakMatrix_ALNG$names,
  ion_map_plot_margin = 40,
  ion_map_plot_mirror_X = rep(F, length(peakMatrix_ALNG$names)),
  ion_map_plot_mirror_Y = rep(F, length(peakMatrix_ALNG$names)),
  ion_map_fixed_aspect_ratio = F
)
```

Arguments

`peakMatrix_RAW`	an rMSIproc peak matrix obtained without the alginment routine (RAW).
`peakMatrix_ALNG`	an rMSIproc peak matrix obtained with the alginment routine (Aligned).
`peaklist_RAW`	a peak list generated with rMSIproc without the alginment routine (RAW).
`peaklist_ALNG`	a peak list generated with rMSIproc with the alginment routine (Aligned).
`mass`	the ion mass to be plot.
`error_range_ppm`	a mass range to be plot specified in ppm.
`min_SNR`	peaks with a signal to noise ratio below this parameter will be discarded.
`mass_offset_RAW`	a mass offset applied to the RAW data to manually compensate possible mass shifts.
`title_label`	the main title of the plot.
`normalization_RAW`	a vector containing the normalization value for each pixel in the RAW data or NA if no normalization should be applied.
`normalization_ALNG`	a vector containing the normalization value for each pixel in the aligned data or NA if no normalization should be applied.
`ion_map_plot_cols`	number of columns to arrange multiple images in the plotting area.
`ion_map_plot_rows`	number of rows to arrange multiple images in the plotting area.
`ion_map_plot_byrow`	a boolean idicating if the plotted images must be sorted by rows.
`ion_map_plot_rotations`	a vector with the rotation in degree to apply to each image.
`ion_map_plot_labels`	text labels to be used for each image.
`ion_map_plot_margin`	a numeric value that determines the separation between images.
`ion_map_plot_mirror_X`	a vector of booleans idicatinc if each image must be flipped horizontally.
`ion_map_plot_mirror_Y`	a vector of booleans idicatinc if each image must be flipped vertically.
`ion_map_fixed_aspect_ratio`	set this flag to true to fix the aspect ratio of the ion images.

Value

a ggplot2 object.

plotMassDriftG *plotMassDriftG.*

Description

Plot the mass shift observed at a target mass.

Usage

```
plotMassDriftG(
  peakMatrix,
  peakList,
  target_mass,
  error_range_ppm,
  min_SNR = 10,
  mass_offset = 0,
  title_label = "",
  normalization = NA,
  visible_legend = T,
  legend_title = "",
  N = 1
)
```

Arguments

`peakMatrix`	an rMSIproc peak matrix.
`peakList`	an rMSIproc peak list (no binning here!).
`target_mass`	the target mass to represent.
`error_range_ppm`	an error range in ppm to be represented around the target mass.
`min_SNR`	the minimum signal to noise ratio to be represented.
`mass_offset`	a mass offset to shift the plot mass axis relative to the target mass.
`title_label`	a string to be used as plot title. Chemical forumlas will be parsed to produce better results.
`normalization`	a vector containing the normalization value for each pixel or NA if no normalization should be applied.
`visible_legend`	a boolean specfing if a legend detailing the included MS images must be displayed.
`legend_title`	the title for the legend.
`N`	numbe of dots to display for each pixel, only the N highest SNR will be displayed.

Value

a ggplot2 object.

plotMassDriftImageComparedG

plotMassDriftImageComparedG.

Description

plots the image map displaying the mass shift at each raster position of two datasets. this allows to compare the results of the alignment routine (raw vs. aliged).

Usage

```
plotMassDriftImageComparedG(
  peakMatrix_RAW,
  peakMatrix_ALNG,
  peaklist_RAW,
  peaklist_ALNG,
  mass,
  error_range_ppm = 400,
  mass_offset_RAW = 0,
  title_label = "",
  ion_map_plot_cols = 2,
  ion_map_plot_rows = 2,
  ion_map_plot_byrow = T,
  ion_map_plot_rotations = rep(0, length(peakMatrix_ALNG$names)),
  ion_map_plot_labels = peakMatrix_ALNG$names,
  ion_map_plot_margin = 40,
  ion_map_plot_mirror_X = rep(F, length(peakMatrix_ALNG$names)),
  ion_map_plot_mirror_Y = rep(F, length(peakMatrix_ALNG$names)),
  ion_map_fixed_aspect_ratio = F
)
```

Arguments

`peakMatrix_RAW`	an rMSIproc peak matrix obtained without the alginment routine (RAW).
`peakMatrix_ALNG`	an rMSIproc peak matrix obtained with the alginment routine (Aligned).
`peaklist_RAW`	a peak list generated with rMSIproc without the alginment routine (RAW).
`peaklist_ALNG`	a peak list generated with rMSIproc with the alginment routine (Aligned).
`mass`	the ion mass to be plot.
`error_range_ppm`	a mass range to be plot specified in ppm.
`mass_offset_RAW`	a mass offset applied to the RAW data to manually compensate possible mass shifts.
`title_label`	the main title of the plot.

`ion_map_plot_cols`
: number of columns to arrange multiple images in the plotting area.

`ion_map_plot_rows`
: number of rows to arrange multiple images in the plotting area.

`ion_map_plot_byrow`
: a boolean idicating if the plotted images must be sorted by rows.

`ion_map_plot_rotations`
: a vector with the rotation in degree to apply to each image.

`ion_map_plot_labels`
: text labels to be used for each image.

`ion_map_plot_margin`
: a numeric value that determines the separation between images.

`ion_map_plot_mirror_X`
: a vector of booleans idicatinc if each image must be flipped horizontally.

`ion_map_plot_mirror_Y`
: a vector of booleans idicatinc if each image must be flipped vertically.

`ion_map_fixed_aspect_ratio`
: set this flag to true to fix the aspect ratio of the ion images.

Value

a ggplot2 object.

---

plotMassDriftImageG *plotMassDriftImageG.*

Description

plots an image map displaying the mass shift at each raster position.

Usage

```
plotMassDriftImageG(
  peakMatrix,
  peakList,
  target_mass,
  error_range_ppm,
  plot_rows = 2,
  plot_cols = 2,
  plot_byrow = T,
  plot_rotations = rep(0, length(peakMatrix$names)),
  plot_mirror_X = rep(F, length(peakMatrix$names)),
  plot_mirror_Y = rep(F, length(peakMatrix$names)),
  plot_margin = 40,
  plot_labels = peakMatrix$names,
  title_label = "",
  fixed_aspect_ratio = F
)
```

Arguments

`peakMatrix`	an rMSIproc peak matrix.
`peakList`	an rMSIproc peak list (no binning here!).
`target_mass`	the target mass to represent.
`error_range_ppm`	an error range in ppm to be represented around the target mass.
`plot_rows`	number of rows to arrange multiple images in the plotting area.
`plot_cols`	number of columns to arrange multiple images in the plotting area.
`plot_byrow`	a boolean idicating if the plotted images must be sorted by rows.
`plot_rotations`	a vector with the rotation in degree to apply to each image.
`plot_mirror_X`	a vector of booleans idicatinc if each image must be flipped horizontally.
`plot_mirror_Y`	a vector of booleans idicatinc if each image must be flipped vertically.
`plot_margin`	a numeric value that determines the separation between images.
`plot_labels`	text labels to be used for each image.
`title_label`	Text label for the plot main title.
`fixed_aspect_ratio`	set this flag to true to fix the aspect ratio of the ion images.

Value

a ggplot2 object.

`plotPeakImage` *plotPeakImage.*

Description

plot the ion image map of a given mass or column of a rMSIproc peak matrix object. If the peak matrix contains data from various datasets the images will be layout horizontally or vertically. At leas mz or column must be specified.

Usage

```
plotPeakImage(
  peakMatrix,
  mz = NULL,
  column = NULL,
  matrix = "intensity",
  normalization = NULL,
  nrow = 2,
  ncol = 2,
  byrow = T,
  margin = 20,
  img_names = peakMatrix$names,
  labels = img_names,
  rotations = rep(0, length(img_names)),
  mirror_x = rep(F, length(img_names)),
  mirror_y = rep(F, length(img_names)),
  pixel_size_um = 100,
  light = 8
)
```

Arguments

`peakMatrix`	the peak matrix in an rMSIproc object.
`mz`	the peak mass to plot, the nearest peak mass will be ploted.
`column`	the column of the peak matrix to plot.
`matrix`	the name of the peak matrix to plot.
`normalization`	the name of normalization to use.
`nrow`	number of rows of the plotted matrix layout (only used if byrow == F)
`ncol`	number of cols of the plotted matrix layout (only used if byrow == T)
`byrow`	a bool specifing if images must be arranged in rows.
`margin`	the separation between plotted images.
`img_names`	a character vector with img names in the desierd plotting order.
`labels`	an alternative character vector with the labels to be displayed for each image.
`rotations`	a vector with the rotation of each images specified in degrees.
`mirror_x`	a bool vector specifing if each image must be flipped or not in X direction prior to rotation.
`mirror_y`	a bool vector specifing if each image must be flipped or not in Y direction prior to rotation.
`pixel_size_um`	the pixel resolution in um.
`light`	the lighting of the plotted image.

`plotPeakImageG` *plotPeakImageG.*

Description

plots and ion map image using the rMSIproc peak matrix. This function is equivalent to the rMSIproc::plotPeakImage but used ggplot to produce more publication-ready results.

Usage

```
plotPeakImageG(
  peakMatrix,
  mass,
  normalization = NA,
  plot_rows = 2,
  plot_cols = 2,
  plot_byrow = T,
  plot_rotations = rep(0, length(peakMatrix$names)),
  plot_mirror_X = rep(F, length(peakMatrix$names)),
  plot_mirror_Y = rep(F, length(peakMatrix$names)),
  plot_margin = 40,
  plot_labels = peakMatrix$names,
  title_label = "",
  fixed_aspect_ratio = F,
  display_colorbar = T
)
```

Arguments

`peakMatrix`	an rMSIproc peak matrix.
`mass`	the ion mass to be plot.
`normalization`	a vector containing the normalization value for each pixel or NA if no normalization should be applied.
`plot_rows`	number of rows of the plotted matrix layout (only used if byrow == F).
`plot_cols`	number of cols of the plotted matrix layout (only used if byrow == T).
`plot_byrow`	a bool specifing if images must be arranged in rows.
`plot_rotations`	a vector with the rotation of each images specified in degrees.
`plot_mirror_X`	a bool vector specifing if each image must be flipped or not in X direction prior to rotation.
`plot_mirror_Y`	a bool vector specifing if each image must be flipped or not in Y direction prior to rotation.
`plot_margin`	the separation between plotted images.
`plot_labels`	an alternative character vector with the labels to be displayed for each image.
`title_label`	the main title of the plot
`fixed_aspect_ratio`	set this flag to true to fix the aspect ratio of the ion images.
`display_colorbar`	set if the colour bar must be displayed.

Value

a ggplot2 object.

plotValuesImage *plotValuesImage*

Description

Plot arbitrary values in a MS images raster positions.

Usage

```
plotValuesImage(
  peakMatrix,
  values,
  nrow = 2,
  ncol = 2,
  byrow = T,
  margin = 20,
  img_names = peakMatrix$names,
  labels = img_names,
  rotations = rep(0, length(img_names)),
  mirror_x = rep(F, length(img_names)),
  mirror_y = rep(F, length(img_names)),
  pixel_size_um = 100,
  scale_title = "",
  light = 8
)
```

Arguments

`peakMatrix`	the peak matrix in an rMSIproc object.
`values`	a vector with the values to be plotted in each pixel.
`nrow`	number of rows of the plotted matrix layout (only used if byrow == F)
`ncol`	number of cols of the plotted matrix layout (only used if byrow == T)
`byrow`	a bool specifing if images must be arranged in rows.
`margin`	the separation between plotted images.
`img_names`	a character vector with img names in the desierd plotting order.
`labels`	an alternative character vector with the labels to be displayed for each image.
`rotations`	a vector with the rotation of each images specified in degrees.
`mirror_x`	a bool vector specifing if each image must be flipped or not in X direction prior to rotation.
`mirror_y`	a bool vector specifing if each image must be flipped or not in Y direction prior to rotation.
`pixel_size_um`	the pixel resolution in um.
`scale_title`	the label for the color scale.
`light`	the lighting of the plotted image.

`plotValuesImageG` *plotValuesImageG.*

Description

Plot a raster image of arbitrary pixel values. This cna be use to plot PCA results for example. This function is equivalent to the rMSIproc::plotValuesImage() but using ggplot2. If the pixel_values is a "factor" object then discrete colors are used. This is useful to plot clustering results.

Usage

```
plotValuesImageG(
  peakMatrix,
  pixel_values,
  scale_label = "",
  title_label = "",
  plot_rows = 2,
  plot_cols = 2,
  plot_byrow = T,
  plot_rotations = rep(0, length(peakMatrix$names)),
  plot_mirror_X = rep(F, length(peakMatrix$names)),
  plot_mirror_Y = rep(F, length(peakMatrix$names)),
  plot_margin = 40,
  plot_labels = peakMatrix$names,
  gradient_scale_colours = rev(rainbow(n = 100, start = 0, end = 0.6)),
  gradient_scale_limits = c(min(pixel_values), max(pixel_values)),
  fixed_aspect_ratio = F,
  display_colorbar = T
)
```

Arguments

`peakMatrix`	an rMSIproc peak matrix.
`pixel_values`	a vector with the pixel values, factor object can be used for a discrete image.
`scale_label`	Text label for the plot scale bar.
`title_label`	Text label for the plot main title.
`plot_rows`	number of rows to arrange multiple images in the plotting area.
`plot_cols`	number of columns to arrange multiple images in the plotting area.
`plot_byrow`	a boolean idicating if the plotted images must be sorted by rows.
`plot_rotations`	a vector with the rotation in degree to apply to each image.
`plot_mirror_X`	a vector of booleans idicatinc if each image must be flipped horizontally.
`plot_mirror_Y`	a vector of booleans idicatinc if each image must be flipped vertically.
`plot_margin`	a numeric value that determines the separation between images.
`plot_labels`	text labels to be used for each image.

`gradient_scale_colours`	alternative color scale for the image.
`gradient_scale_limits`	alternative limits for the pixel values.
`fixed_aspect_ratio`	set this flag to true to fix the aspect ratio of the ion images.
`display_colorbar`	set if the colour bar must be displayed.

Value

a ggplot2 object.

---

print.rMSIprocPeakMatrix *Displays a summary of a peak matrix.*

Description

Displays a summary of a peak matrix.

Usage

```
## S3 method for class 'rMSIprocPeakMatrix'
print(x)
```

Arguments

`x`	rMSIproc peak matrix object.

---

printrMSIdataInfo *printrMSIdataInfo.*

Description

Prints HDD storing information of rMSI data object using the C++ backend.

Usage

```
printrMSIdataInfo(img)
```

Arguments

`img`	and rMSI object image.

`ProcessImage` *ProcessImage.*

Description

Perform all image pre-processing using a multi-threading implementation. If aligment is used then the hdd files are overwirted with aligned data. A recomeneded value of aligment ietarations is 3.

Usage

```
ProcessImage(
  img,
  EnableSmoothing = T,
  SmoothingKernelSize = 5,
  EnableAlignment = T,
  AlignmentIterations = 3,
  AlignmentMaxShiftppm = 200,
  AlignmentBilinear = F,
  AlignmentRefLow = 0,
  AlignmentRefMid = 0.5,
  AlignmentRefHigh = 1,
  AlignmentOversampling = 2,
  EnableCalibration = T,
  CalibrationPeakWin = 20,
  EnablePeakPicking = T,
  SNR = 5,
  peakWindow = 10,
  peakUpSampling = 10,
  UseBinning = T,
  BinTolerance = 5,
  BinFilter = 0.05,
  BinToleranceUsingPPM = F,
  EnableSpectraNormalization = T,
  EnableTICNorm = T,
  EnableRMSNorm = T,
  EnableMAXNorm = T,
  EnableTICAcqNorm = T,
  NumOfThreads = min(parallel::detectCores()/2, 6),
  CalSpan = 0.75,
  ExportPeakList = F
)
```

Arguments

`img`	an rMSI data object to process or a list of rMSI objects if various datasets must merged for processing.

`EnableSmoothing`	a boolean declaring if smoothing alignment must be performed.
`SmoothingKernelSize`	size of smoothing kernel. NULL value may be specified if EnableSmoothing is FALSE.
`EnableAlignment`	a boolean declaring if label-free alignment must be performed.
`AlignmentIterations`	number of iterations of label-free alignment. The rMSI ramdisk will be overwritted with aligned data. NULL value may be specified if EnableAlignment is FALSE.
`AlignmentMaxShiftppm`	the maximum shift that alignment can apply in ppm. NULL value may be specified if EnableAlignment is FALSE.
`AlignmentBilinear`	if TRUE the biliniar algiment mode will be used insetad of linear.
`AlignmentRefLow`	the relative location of the spectrum where the bottom part correlation is calculated.
`AlignmentRefMid`	the relative location of the spectrum where the central part correlation is calculated (only for bilinear mode).
`AlignmentRefHigh`	the relative location of the spectrum where the top part correlation is calculated.
`AlignmentOversampling`	the oversampling used in spectrum scale/shift to provide better accuracy.
`EnableCalibration`	a boolean declaring if mass calibration must be performed.
`CalibrationPeakWin`	the windows size used for peak detection in calibration window.
`EnablePeakPicking`	a boolean declaring if peak-pickin (and binning) must be performed.
`SNR`	minimal singal to noise ratio of peaks to retain. NULL value may be specified if EnablePeakPicking is FALSE.
`peakWindow`	windows size used for peak detection. Generally should be similar to peak with number of data points. NULL value may be specified if EnablePeakPicking is FALSE.
`peakUpSampling`	upsampling factor used in peak interpolation fo exact mass prediction. NULL value may be specified if EnablePeakPicking is FALSE.
`UseBinning`	if true binned matrices are returned instead of peak lists.
`BinTolerance`	the tolerance used to merge peaks to the same bin. It is recomanded to use the half of peak width in ppm units. NULL value may be specified if EnablePeakPicking is FALSE.
`BinFilter`	the peaks bins non detected in at least the BinFitler*TotalNumberOfPixels spectra will be deleted. NULL value may be specified if EnablePeakPicking is FALSE.

`BinToleranceUsingPPM`	if True the peak binning tolerance is specified in ppm, if false the tolerance is set using scans.
`EnableSpectraNormalization`	if normalization must be applied.
`EnableTICNorm`	if TIC normalization must be performed on spectra.
`EnableRMSNorm`	if RMS normalization must be performed on spectra.
`EnableMAXNorm`	if MAX normalization must be performed on spectra.
`EnableTICAcqNorm`	if TICAcq normalization must be performed on spectra.
`NumOfThreads`	the number number of threads used to process the data.
`CalSpan`	the used span for the loess fittin used in mass calibration.
`ExportPeakList`	a boolean detailing wheter the un-binned peak list must be exported or not.

Value

a named list containing: - The process image reference (procImg). - The results peak-picking (peakMat). This can be returned in two forms: From1 (if binning is used) - a list containing three matrices (intensity, SNR and area) and a vector with a common mass axis. Form2 (if NO binning is applied) - a list of detected peaks for each pixel. - The applied mass shifts in first alignment iteration (alignShifts).

`ProcessWizard` *ProcessWizard.*

Description

Imports and process MSI data using a friendly GUI. Various images can be loaded and processed with a single execution. Data can be in XMASS, tar (rMSI) or imzML format. Processed data will be saved in a user specified directory. The applied processing consists in: - Label-free aligment (various iterations can be performed, zero iterations means no alignment). - Peak-picking. - Peak-binning. - Mass calibration with internal reference compounds. Processed data includes: - a .tar file with the processed data. - a rMSIproc formated matrices with binned peaks. - a plain text file with used processing parameters.

Usage

```
ProcessWizard(
  deleteRamdisk = T,
  overwriteRamdisk = F,
  calibrationSpan = 0.75,
  store_binsize_txt_file = F
)
```

Arguments

`deleteRamdisk`	if the used ramdisks for MS images must be deleted for each image processing (will be deleted after saving it to .tar file).
`overwriteRamdisk`	if the current ramdisk must be overwrited.
`calibrationSpan`	the used span in the loess fitting for mass calibration.
`store_binsize_txt_file`	if the binSize used in each binning column must be soterd in a text file.

`rcpp_hello` *Hello, Rcpp!*

Description

Returns an R `list` containing the character vector `c("foo","bar")` and the numeric vector `c(0,1)`.

Usage

```
rcpp_hello()
```

Examples

```
rcpp_hello()
```

`SaveProcessingParams` *SaveProcessingParams.*

Description

Save all parameters in a list of processing params generated using ImportWizardGui() function. Parameters will be saved in a plain text file.

Usage

```
SaveProcessingParams(
  procParams,
  filepath,
  xmlRoiFilesInclude = NULL,
  xmlRoiFilesExclude = NULL,
  RoiNormalization = NULL
)
```

Arguments

`procParams`	a list of parameters.
`filepath`	a full path where params will be stored
`xmlRoiFilesInclude`	a vector with the used ROI XML files for ID inclusion, NULL if no ROI was used.
`xmlRoiFilesExclude`	a vector with the used ROI XML files for ID exclusion, NULL if no ROI was used.
`RoiNormalization`	a string with the name of normalization used to export the data summary.

`Smoothing` *Smoothing.*

Description

Smoothing.

Usage

```
Smoothing(x, method = "SavitzkyGolay", ...)
```

Arguments

`x`	the intensities of a a mass spectrum to smooth.
`method`	the method to use for smoothing, available option are: "SavitzkyGolay".
`...`	specific parameters to each smoothing method Smooths a vector of data using the specified smoothing method.

Value

the smoothed data.

`Smoothing_SavitzkyGolay` *Smoothing_SavitzkyGolay.*

Description

Computes the Savitzky-Golay smoothing of a vector x using a filter size of sgSize.

Usage

```
Smoothing_SavitzkyGolay(x, sgSize = 5L)
```

Arguments

`x`	the data vector to smooth.
`sgSize`	valid values are: 5, 7, 9, 11, 13, 15.

Value

the smoothed data vector.

`StorePeakMatrix` *StorePeakMatrix.*

Description

Stores a binned peaks matrix to HDD. Data is stored zip compressed, so it is recomeneded to specify the name with .zip extension.

Usage

```
StorePeakMatrix(data_path, data)
```

Arguments

`data_path`	full path including filename where data must be stored.
`data`	a List containing intensity, SNR and area matrices, the mass axis vector and a data.frame containing in each variable a normalization vector.

`StorePeakMatrixC` *StorePeakMatrix.*

Description

Stores a binned peaks matrix to HDD.

Usage

```
StorePeakMatrixC(path, mat)
```

Arguments

`path`	full path to directory where data must be stored.
`mat`	an R List containing intensity, SNR and area matrices the mass axis vector and an R data.frame containing a normalization on each column.

`summary.rMSIprocPeakMatrix` *Displays a summary of a peak matrix.*

Description

Displays a summary of a peak matrix.

Usage

```
## S3 method for class 'rMSIprocPeakMatrix'
summary(x)
```

Arguments

`x`	rMSIproc peak matrix object.

`TestAreaWindow` *TestAreaWindow.*

Description

Method to test the implementation of Area window in R session.

Usage

```
TestAreaWindow(mass, WinSize = 20L, UpSampling = 10L)
```

Arguments

`mass`	a NumericVector containing the mass axis of the spectrum.
`WinSize`	The windows used to detect peaks and caculate noise.
`UpSampling`	the oversampling used for acurate mass detection and area integration.

Value

a NumericVector containing the Area Window.

`TestHanningWindow` *TestHanningWindow.*

Description

Method to test the implementation of Hanning window in R session.

Usage

```
TestHanningWindow(mass, WinSize = 20L, UpSampling = 10L)
```

Arguments

`mass`	a NumericVector containing the mass axis of the spectrum.
`WinSize`	The windows used to detect peaks and caculate noise.
`UpSampling`	the oversampling used for acurate mass detection and area integration.

Value

a NumericVector containing the Hanning Window.

TestPeakInterpolation_C *TestPeakInterpolation_C.*

Description

TestPeakInterpolation_C.

Usage

```
TestPeakInterpolation_C(
  mass,
  intensity,
  peakIndex,
  WinSize = 20L,
  UpSampling = 10L,
  useHanning = FALSE,
  Iterations = 1L
)
```

Arguments

`mass`	a NumericVector containing the mass axis of the spectrum.
`intensity`	a NumericVector where peaks must be detected.
`peakIndex`	the location of the peak to interpolate in the spectrum.
`WinSize`	The windows used to detect peaks and caculate noise.
`UpSampling`	the oversampling used for acurate mass detection and area integration.
`useHanning`	if hanning windowing must be used befor interpolation.
`Iterations`	number of iterations to perform. This is just for testing interpolation efficiency

Value

a NumerixVector with the FFT interpolated peak shape.

theme_black *theme_black. custom ggplot theme to display ion map images with a black background*

Description

theme_black. custom ggplot theme to display ion map images with a black background

Usage

```
theme_black(base_size = 12, base_family = "")
```

Arguments

`base_size`

`base_family`

`[.rMSIprocPeakMatrix` *Subsetting operator for rMSIproc peak matrices.*

Description

Subsetting operator for rMSIproc peak matrices.

Usage

```
## S3 method for class 'rMSIprocPeakMatrix'
x[pixels = 1:sum(x$numPixels), columns = 1:length(x$mass)]
```

Arguments

`x`	rMSIproc peak matrix object.
`pixels`	the selected rows to retain in the peak matrix. This argument can be an integer, a boolean or a character.
`columns`	an integer vector with the peak matrix columns to retain.

Value

rMSIproc peak matrix object.

Examples

```
#For the following example we will load an rMSIproc peak matrix in the pks variable:
pks <- rMSIproc::LoadPeakMatrix("/path/to/my/peak/matrix.zip")

#Subsetting a peak marix by pixel ID's:
pks1_100 <- pks[1:100, ]

#Subsetting a peak matrix using a boolean expression:
clus <- kmeans(pks$intensity/pks$normalizations$TIC, centers = 5) #perform a kmeans clustering with the whole peak
clus2SubImg <- pks[clus$cluster==2, ] #Creat a subdataset with only pixels belonging to cluster 2

#Subsetting a peak matrix using image names:
pks_brain1 <- pks["Brain_img1", ]

#Subsetting to a specific column range:
pks_massrange <- pks[, 10:50]
```

```
#Subsetting by columns and rows:
pks_brain1 <- pks["Brain_img1", 10:50]
```

Index