Package ‘MALDIrppa’

November 9, 2018

Type Package
Title MALDI Mass Spectrometry Data Robust Pre-Processing and Analysis
Version 1.0.1-2
Date 2018-11-03
Author Javier Palarea-Albaladejo
Maintainer Javier Palarea-Albaladejo <javie.palarea@bioss.ac.uk>
Depends R (>= 3.2.5), MALDIquant, signal, robustbase, wmtsa, lattice
ByteCompile yes
Description Provides methods for quality control and robust pre-processing and analysis of MALDI mass spectrometry data.
License GPL (>= 2)
Repository CRAN
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation no
Date/Publication 2018-11-09 14:20:15 UTC

R topics documented:

MALDIrppa-package ....................................................... 2
addMetadata ................................................................. 2
alignPeaks ................................................................. 3
countPeaks ................................................................. 4
deletePeaks ............................................................... 5
detectOutliers ............................................................ 6
importSpectra ............................................................. 7
peakPatterns .............................................................. 9
plot.scSpectra ............................................................ 10
rawToPeaks ............................................................... 11
rawToSpectra ............................................................ 12
redResolution ............................................................ 13
addMetadata

Description

This function adds metadata to the metaData slot of an AbstractMassObject-class class object.
**alignPeaks**

**Usage**

```
addMetadata(x, metadata, pos)
```

**Arguments**

- `x` List of `AbstractMassObject-class` (MassSpectra or MassPeaks) class objects.
- `metadata` Vector containing the metadata to be included for each element of `x` (same length as `x`).
- `pos` Position of the new metadata within the `metaData` slot list of each element of `x`.

**Value**

List of `AbstractMassObject-class` class objects including the new metadata in their `metaData` slot.

**Examples**

```
# Load example data

data(spectra) # list of MassSpectra class objects
data(type)    # metadata

# Add metadata

info <- paste("Spectrum No.",1:length(spectra)) # Artificial metadata vector

spectra2 <- addMetadata(spectra,info,1)

# Check info in metaData slot

spectra2[[1]]@metaData
```

---

**alignPeaks**

*Compact peak alignment process for MassPeaks objects*

**Description**

This function provides a single command for selecting anchor peaks, peak alignment and binning of `MassPeaks` class objects (*MALDIquant* package). It also deals with alignment-related issues found in high-resolution mass spectrometry data.

**Usage**

```
alignPeaks(x, minFreq = 0.9, tolerance = 0.003, ...)
```
countPeaks

Arguments

- `x`: A list of MassPeaks class objects.
- `minFreq`: Minimum relative frequency of a peak over `x` to be considered as anchor peak for alignment (see referencePeaks).
- `tolerance`: Maximal deviation in peak masses to be considered as identical (see referencePeaks, binPeaks).

Details

See warpMassPeaks and binPeaks in the MALDIquant package for details about the alignment and binning algorithms. Note that alignPeaks applies an additional binning round which helps to correct for misalignment issues found after using the default strict or relaxed bin creation rules in high-resolution mass spectrometry data.

Value

A list of MassPeaks class objects with aligned peaks along a common m/z range.

Examples

```r
# Load example data
data(spectra) # list of MassSpectra class objects

# Some pre-processing

spectra <- screenSpectra(spectra)$fspectra
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)

# Peak alignment

peaks <- alignPeaks(peaks, minFreq = 0.8)
```

countPeaks      Count the number of peaks in MassPeaks objects

Description

This function provides the number of peaks of each element of a list of MassPeaks objects.

Usage

`countPeaks(x)`
**deletePeaks**

**Arguments**

- `x` A list of `MassPeaks` objects.

**Value**

A vector consisting of the number of peaks for each peak profile in `x`.

**Examples**

```r
# Load example data
data(spectra) # list of MassSpectra class objects

# Some pre-processing
spectra <- screenSpectra(spectra)$fspectra
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)

# Count peaks
npeaks <- countPeaks(peaks)
```

**Description**

This function deletes peaks of height (intensity) below a given value in `MassPeaks` objects.

**Usage**

`deletePeaks(x, min = NULL)`

**Arguments**

- `x` A list of `MassPeaks` objects.
- `min` Lower threshold used to discard a peak.

**Details**

This functions takes a list of `MassPeaks` objects and filters out peaks of height (intensity) falling below the given minimum value.

**Value**

A filtered list of `MassPeaks` objects.
detectOutliers

**Examples**

```r
# Load example data
data(spectra) # list of MassSpectra class objects

# Some pre-processing
spectra <- screenSpectra(spectra)$fspectra
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)

# Delete peaks of intensity < 30
peaks <- deletePeaks(peaks, min = 30)
```

**detectOutliers**

*Detection of outlying mass peak profiles*

**Description**

This function identifies outlying cases in a collection of processed mass peak profiles. It can be applied either on peak intensities or binary data (peak presence/absence patterns). It allows to specify a grouping factor in order to execute the procedure at the desired level of aggregation.

**Usage**

`detectOutliers(x, by = NULL, binary = FALSE, ...)`

**Arguments**

- `x`: A list of `MassSpectrum` objects containing processed peaks.
- `by`: If given, a grouping variable (`factor` or `numeric`) subsetting the data.
- `binary`: Logical value. It indicates whether the procedure must be applied on either peak intensities (FALSE, default) or on binary peak presence/absence patterns (TRUE).
- `...`: Optional arguments for the robust outlier detection method.

**Details**

This function marks samples with mass peak profiles that largely deviates from other samples at the given aggregation level. It uses robust methods for the detection of multivariate outliers applied on metric multidimensional scaling (MDS) coordinates (Euclidean distance is used for peak intensities and binary distance for binary profiles; see `dist`). The number of MDS coordinates used is generally set to \( p = \lfloor n/2 \rfloor \), where \( n \) is the number of samples in the target subset. This is an upper cap recommended for the computation of the robust MCD estimator by `covMcd`. However, that rule of thumb can still generate matrix singularity problems with `covMcd` in some cases. When
this occurs detectOutliers further reduces p to use the maximum number of MDS coordinates giving rise to a non-singular covariance matrix (min(p) = 2 in any case). The adaptive multivariate outlier detection algorithm was adapted from the mvoutlier package.

Value

If by = NULL, a logical vector of length equal to the number of elements of x indicating outlying samples by TRUE. Otherwise, a 2-column data.frame is generated which includes such a logical vector along with the grouping variable given in by.

Examples

# Load example data
data(spectra) # list of MassSpectra class objects
data(type) # metadata

# Some pre-processing

sc.results <- screenSpectra(spectra, meta=type)
spectra <- sc.results$spectra # filtered mass spectra
type <- sc.results$meta # filtered metadata
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)
peaks <- alignPeaks(peaks, minFreq = 0.8)

# Find outlying samples at isolate level

## Not run:
out <- detectOutliers(peaks, by = type$(isolate)

## From peak presence/absence patterns

out.binary <- detectOutliers(peaks, by = type$(isolate, binary = TRUE)

## End(Not run)

---

**importSpectra**

Convert mass spectra from text files into MassSpectrum objects

**Description**

This function allows to import collections of mass spectra stored in individual text files into a list of MassSpectrum objects.
importSpectra

Usage

importSpectra(where = getwd())

Arguments

where
Path to the folder where the text files are stored (default: current working directory).

Details

This function works with dat, csv or txt file types containing two columns: the first one referring to common m/z values and the second one to intensities (using single-space separator between both and no column names). It reads all the files in the given folder and creates a list of MassSpectrum objects. For importing data from more specialised file formats we refer the reader to the package MALDIquantForeign.

Value

A list of MassSpectrum objects.

Examples

# Create fake mass spectrometry data

s1 <- cbind(1:20, rlnorm(20))
s2 <- cbind(1:20, rlnorm(20))
s3 <- cbind(1:20, rlnorm(20))

current <- getwd()
dir.create(paste(getwd(),"/example",sep=""))
setwd(paste(getwd(),"/example",sep=""))

write.table(s1, file="s1.csv", row.names = FALSE, col.names = FALSE, sep=" ")
write.table(s2, file="s2.csv", row.names = FALSE, col.names = FALSE, sep=" ")
write.table(s3, file="s3.csv", row.names = FALSE, col.names = FALSE, sep=" ")

# Import files in ~example and arrange into a list of MassSpectrum objects

spectra <- importSpectra()

# Delete example folder and back to original working directory
unlink(getwd(),recursive = TRUE)
setwd(current)
peakPatterns

Display peak presence/absence patterns

Description

This function displays the patterns of peak presence and absence in an intensity matrix as generated from intensityMatrix.

Usage

peakPatterns(x, abs.lab = NA, barplot = TRUE,  
axis.lab = c("m/z", "Index"), bar.col = "red3",  
cell.col = c("white", "dodgerblue"), grid = FALSE,  
grid.col = "black", grid.lty = "dotted", cex.axis = 0.5,  
cex.lab = 0.5, ...)

Arguments

x  A matrix, data.frame or a list of MassPeaks objects.

abs.lab  Unique label used to denote peak absence in x (NA, default).

barplot  Logical value indicating whether a barplot of relative peak frequency across samples is displayed (TRUE, default).

axis.lab  Vector of axis labels in the c("x", "y") format.

bar.col  Colour of the bars in the barplot.

cell.col  Vector of colours for the table cells (format c("col.absence", "col.pesence")).

grid  Logical value indicating whether gridlines are added (FALSE, default).

grid.col  Colour of the gridlines ("black", default).

grid.lty  Style of the gridlines ("dotted", default. See lty in par).

cex.axis  Axis tick labels scaling factor relative to default.

cex.lab  Axis labels scaling factor relative to default.

...  Other arguments.

Details

The peak presence/absence patterns are displayed by rows from the first (top) to the last (bottom) sample in the data set x over the range of common m/z points. Positive peaks are by default represented by coloured cells whereas zero or absent peaks are left blank. A barplot on the top margins shows the relative frequency of a peak at each m/z point across samples.

See Also

See intensityMatrix.
Examples

# Load example data

data(spectra) # list of MassSpectra class objects
data(type) # metadata

# Some pre-processing

sc.results <- screenSpectra(spectra, meta=type)
spectra <- sc.results$fspectra # filtered mass spectra
type <- sc.results$ftmeta # filtered metadata
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)
peaks <- alignPeaks(peaks, minFreq = 0.8)

# Display patterns across all data

peakPatterns(peaks)

# Check results within isolate 280

peakPatterns(peaks[type$Isolate=="280"])

plot.scSpectra  
\textit{Graphical summary of screenSpectra objects}

Description

This is a plot method for scSpectra objects that displays the results from applying screenSpectra to identify potential faulty, low-quality raw mass spectra.

Usage

## S3 method for class 'scSpectra'

plot(x, type = c("index", "hist", "casewise"), breaks = 30, labels = FALSE, col = "green3", ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A scSpectra object as generated by screenSpectra.</td>
</tr>
<tr>
<td>type</td>
<td>Type of graphical display.</td>
</tr>
<tr>
<td>breaks</td>
<td>Number of break points for plotting a histogram when type = &quot;hist&quot; (default = 30).</td>
</tr>
<tr>
<td>labels</td>
<td>Vector of labels for the mass spectra (default = FALSE, no labels).</td>
</tr>
<tr>
<td>col</td>
<td>Colour for the histogram bars when type = &quot;hist&quot;.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments.</td>
</tr>
</tbody>
</table>
Details

For type = "index" (default) the upper and lower fences used to declare a mass spectrum as potentially low-quality are shown along with their A scores (see `screenSpectra` for details). Point labels can be added using the `labels` argument (either a position index when `labels = TRUE` or a given vector of labels; see examples below). For type = "hist" a histogram of the distribution of the A scores is produced along with the tolerance fences. Finally, type = "casewise" displays interactively the marked spectra for visual inspection.

See Also

See `screenSpectra` and `summary.scSpectra`.

Examples

```r
# Load example data

data(spectra) # list of MassSpectra objects
data(type) # metadata

sc.results <- screenSpectra(spectra)
plot(sc.results)
plot(sc.results, labels = TRUE)
plot(sc.results, labels = type$SpectID)
plot(sc.results, type = "hist")
## Not run:
plot(sc.results, type = "casewise", labels = type$SpectID)
## End(Not run)
```

---

`rawToPeaks`  
**Create list of MassPeaks objects**

Description

This is an auxiliary function to create a list of `MassPeaks` objects from raw data.

Usage

`rawToPeaks(mz, I)`

Arguments

- `mz`  
  Vector of m/z values.

- `I`  
  Matrix of peak intensity values.
Details
This function creates a list of `MassPeaks` objects from a vector of common m/z values and a matrix of column vectors of peak intensities for a collection of mass peak profiles. The column names are used to label the elements of the list.

Value
A list of `MassPeaks` objects.

---

rawToSpectra

Create list of `MassSpectrum` objects

Description
This is an auxiliary function to create a list of `MassSpectrum` objects from raw data.

Usage
`rawToSpectra(mz, I)`

Arguments
- `mz` Vector of m/z values.
- `I` Matrix of intensity values.

Details
This function creates a list of `MassSpectrum` objects from a vector of common m/z values and a matrix of column vectors of intensities for a collection of mass spectra. The column names are used to label the elements of the list.

Value
A list of `MassSpectrum` objects.

See Also
See `importSpectra`. 
redResolution

Reduce resolution of MassSpectrum objects

Description

This function allows to obtain a lighter version of a list of MassSpectrum objects by decreasing their m/z resolution.

Usage

redResolution(x, by = 1)

Arguments

x A list of MassSpectrum objects.
by Number of times reduction (by = 1, default).

Details

This function reduces the resolution of mass spectra by eliminating a regular sequence of m/z sampling points in steps given by the argument by. For example, specifying by = 2 means to reduce the length and memory usage of the signal by a half approximately.

Value

A list of MassSpectrum objects.

Examples

# Load example data
data(spectra) # list of MassSpectra class objects

# Reduce resolution by a half
spectra.LowRes <- redResolution(spectra, by = 2)
**Description**

This function implements a quality control check to help in the identification of possibly faulty, low-quality raw mass spectra. It computes an atypicality score and labels suspicious profiles for further inspection and filtering.

**Usage**

```r
screenSpectra(x, meta = NULL, threshold = 1.5, estimator = c("Q", "MAD"),
          method = c("adj.boxplot", "boxplot", "ESD", "Hampel", "RC"),
          nd = 1, lambda = 0.5, ...)
```

**Arguments**

- `x`: A list of `MassSpectrum` objects.
- `meta`: (optional) Matrix or vector containing metadata associated to `x`. Typically a data matrix including spectrum ID, biotype, replicate number, etc. for each element of `x`.
- `threshold`: Multiplicative factor used in computing the upper and lower fences to determine passes and failures. It is related to the actual method used to compute the fences (see `method`). Typically, `threshold = 1.5` (default value) for the boxplot rules, and `threshold = 3` for the others.
- `estimator`: Robust scale estimator used:
  - Q: robust location-free scale estimate (default, see Qn function in robustbase package). More efficient than MAD and adequate for non-symmetric distributions.
  - MAD: median absolute deviation scale estimate. Very robust and preferred for fairly symmetric distributions.
- `method`: Method used to compute upper and lower fences for the identification of atypical mass spectra.
  - boxplot: standard boxplot rule based on the first and third quartiles and the interquartile range.
  - adj.boxplot: extension of boxplot rule for strongly asymmetric data (default).
  - ESD: extreme studentized deviation method. Based on the mean and the standard deviation of the data. Typically used with `threshold = 3` (three-sigma rule).
  - Hampel: robust version of the ESD method based on the median and the median absolute deviation estimate (MAD).
  - RC: as Hampel’s but replacing MAD by Rousseeuw & Croux (1993)’s Qn as scale estimate.
- `nd`: Order for the derivative function of the mass spectra (default = 1).
- `lambda`: Weight given to each component of the atypicality score (values in [0, 1], default = 0.5, see details below).
- `...`: Other arguments.
Details

The procedure computes an atypicality score (A score) based on a weighted function of two components: (1) a robust scale estimator (\( \hat{q} \) or MAD) of the n-order derivative (computed using Savitzky-Golay smoothing filter) of scaled mass spectra and (2) the median intensity of the signals. Given a method to determine tolerance fences, a mass spectrum is labelled as potentially faulty, low-quality according to the magnitude of its A score. The adjacent boxplot method based on the \( \hat{q} \) scale estimator and equal weights to both components (\( \lambda = 0.5 \)) are the default options. The greater \( \lambda \) the higher the weight given to the scale estimator in the A score. The function produces summaries and a list of mass spectra and (if given) associated metadata in which the identified cases were filtered out.

Value

An object of class `scSpectra` with elements:

- `fspectra` List of mass spectra (MassSpectrum class) with potential low-quality cases filtered out.
- `fmeta` Associated filtered metadata (data.frame object).
- `est.table` Results table showing the mass spectra ID, A score and label (pass/failure).
- `...` Other details (see method `summary.scSpectra` for scSpectra objects).

See Also

See methods `summary.scSpectra` and `plot.scSpectra` for scSpectra objects.

Examples

```r
# Load example data
data(spectra) # list of MassSpectrum objects
data(type)    # metadata

# Results using different settings
sc.results <- screenSpectra(spectra)
sr.results <- screenSpectra(spectra, type)
sr.results <- screenSpectra(spectra, type, method = "RC")
sr.results <- screenSpectra(spectra, type, threshold = 3, estimator = "MAD", method = "Hampel")

# Numerical and graphical summary
summary(sc.results)
plot(sc.results)

# Save filtered data for further pre-processing
filtered.spectra <- sc.results$fspectra
filtered.type <- sc.results$fmeta
```
**snrPeaks**

*Extract signal-to-noise ratio thresholds from MassPeaks objects*

**Description**

This function extracts the thresholds used to determine peaks from mass spectra based on signal-to-noise ratio (SNR) (threshold equal to SNR*noise).

**Usage**

`snrPeaks(x)`

**Arguments**

- `x`: A list of `MassPeaks` objects.

**Details**

Given a collection of `MassPeaks` objects as obtained from `detectPeaks`, this function provides the thresholds used in each case to determine peaks from the original mass spectra. The thresholds are calculated as the product of a SNR value set by the user and the estimated noise of the signal (see `detectPeaks`).

**Value**

A list of vectors of SNR-based thresholds, one for each sample.

**Examples**

```r
# Load example data
data(spectra) # list of MassSpectra class objects

# Some pre-processing
spectra <- screenSpectra(spectra)$fspectra
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)

# Extract thresholds for each mass peak profile
SNRs <- snrPeaks(peaks)
```
Example mass spectra profiles

Description

List of mass spectra (MassSpectrum class).

Usage

data(spectra)

Details

Low-resolution version of a MALDI-TOF mass spectrometry data set in the range [2500, 13000] m/z provided for illustration purposes. It consists of 4 technical replicates of 5 biological replicates from 19 bacterial isolates (see type for associated metadata).

Examples

data(spectra)

str(spectra[[1]])
# plot(spectra[[1]])

Numerical summary of screenSpectra objects

Description

This is a summary method for scSpectra objects that generates a numerical summary of the settings and results from applying screenSpectra to identify potential faulty, low-quality raw mass spectra.

Usage

## S3 method for class 'scSpectra'
summary(object, ncases = 10, ...)

Arguments

object A scSpectra object as generated from screenSpectra.
ncases Number of cases shown in the results table.
... Other arguments.
Summary of mass peak profiles

This function generates a numerical summary of a collection of MassPeaks objects.

Usage

summaryPeaks(x, digits = 4)

Arguments

  x          A list of MassPeaks objects.
  digits     Integer indicating the number of decimal places to be used.

Details

  For each MassPeaks on the list this function provides summary statistics of m/z points, peak intensities and SNR thresholds (number, minimum, mean, standard deviation, median, mean absolute deviation, maximum).

Value

  A data.frame containing summary information of a collection of MassPeaks objects.
Examples

# Load example data
data(spectra) # list of MassSpectra class objects
data(type) # metadata

# Some pre-processing
sc.results <- screenSpectra(spectra, meta = type)
spectra <- sc.results$fspectra
type <- sc.results$fmeta

spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)

names(peaks) <- type$SpectID # spectra IDs are lost after removeBaseline()

# Summary of peak profile features (results for positions 10 to 20)
summaryPeaks(peaks[10:20])

summarySpectra

Summary of mass spectra

Description

This function generates a numerical summary of a collection of MassSpectra objects.

Usage

summarySpectra(x, digits = 4)

Arguments

x A list of MassSpectrum objects.
digits Integer indicating the number of decimal places to be used.

Details

For each MassSpectrum on the list this function provides summary statistics of m/z points and signal intensities (number, minimum, mean, standard deviation, median, mean absolute deviation, maximum).
Value

A data.frame containing summary information of a collection of MassSpectrum objects.

Examples

# Load example data
data(spectra)  # list of MassSpectra class objects
data(type)    # metadata

# Summary of spectra features (results for 20 first mass spectra)
summarySpectra(spectra[1:20])

# Some pre-processing
sc.results <- screenSpectra(spectra, meta = type)
spectra <- sc.results$fspectra
type <- sc.results$fmeta

spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
names(spectra) <- type$SpectID  # spectra IDs are lost with removeBaseline()

# Summary of spectra features (results for positions 10 to 20)
summarySpectra(spectra[10:20])

transfIntensity  Transform intensity of MassSpectrum objects

Description

This function applies user-defined transformations on the intensities of MassSpectrum objects.

Usage

transfIntensity(x, fun = NULL, ...)

Arguments

x  A list of MassSpectrum objects.
fun  Name of an user-defined transformation function or any other pre-defined one in R.
...  Other arguments.
Details
This function allows the user to define any sensible function to be applied on signal intensities. For logarithm and square root transformations it is equivalent to `transformIntensity` in the MALDIquant package.

Value
A list of `MassSpectrum` objects with signal intensities transformed according to `fun`.

Examples
# Load example data
data(spectra) # list of MassSpectra class objects

# Scale intensities into [0, 1] by dividing by their maximum value
scale.max <- function(x){x/max(x)} # define scaling function
scaled.spectra <- transfIntensity(spectra, fun = scale.max)

# Compute natural logarithm of intensity values (using the pre-defined sqrt R function)
log.spectra <- transfIntensity(spectra, sqrt)

---

**type**  
*Example mass spectra metadata*

Description
Metadata associated to the `spectra` data set containing information about isolate, biological and technical replicate numbers and mass spectra IDs.

Usage
data(type)

Format
The format is:
Isolate: Factor w/ 14 levels "280","43","45"... 2 2 2 2 2 2 2 2 2 2 ...
BioRep : int 1 1 1 1 2 2 2 2 3 3 ...
TechRep: int 1 2 3 4 1 2 3 4 1 2 ...
SpectID: Factor w/ 315 levels "160408C13","160408C14"... 1 2 3 4 5 6 7 8 9 10 ...
wavSmoothing

Discrete wavelet transformation for MassSpectrum objects

Description
This function performs undecimated wavelet transform (UDWT) on mass spectra in MassSpectrum format.

Usage
wavSmoothing(x, thresh.scale = 2.5, ...)

Arguments
x A list of MassSpectrum objects.
thresh.scale Smoothing factor.
... Other arguments.

Details
This is an adaptation of the wavShrink function (wmtsa package) to be used with MassSpectrum objects. See wavShrink in wmtsa package for options.

Value
A list of MassSpectrum objects with denoised signal intensities.

Examples
# Load example data
data(spectra) # list of MassSpectra class objects
# sqrt transformation and signal smoothing using UDWT
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
writeIntensity  Write intensity matrix in different formats

Description

This function writes an intensity matrix as generated by `intensityMatrix` into a file in the R, csv, NEXUS or FASTA formats. For NEXUS format it allows to specify weights for peaks.

Usage

```r
writeIntensity(x, filename = "intMatrix", format = c("R", "csv", "NEXUS", "FASTA"),
            binary = FALSE, labels = NULL, weights=NULL, ...)  
```

Arguments

- `x`: Intensity matrix as obtained from `intensityMatrix`.
- `filename`: A character string specifying a name for the destination file (filename extension not required).
- `format`: One of R (default .Rdata file), text (comma-separated .csv file), NEXUS (.nex file) or FASTA (.fas file).
- `binary`: Logical value. If TRUE, a binary version (1: peak presence, 0: peak absence) of `x` is saved (default FALSE).
- `labels`: Optional vector of ID labels for the samples.
- `weights`: Optional numeric vector of peak weights (NEXUS format).
- `...`: Additional arguments.

Details

This is a wrapper function to simplify the writing of an intensity matrix in different formats while adding some extra features. It includes the common NEXUS and FASTA formats as an extension of functions in the ape package to handle peak intensity data. It also allows for taxa/sample pre-computed peak weights to be included in the NEXUS file. It checks whether the names meet NEXUS name conventions and gives them adequate format if not. A binary intensity matrix is always internally generated (binary = TRUE) when either the NEXUS or FASTA format is chosen. If any, NA values in `x` are assumed to denote zero intensity/peak absence and are then converted into zeros.

Examples

```r
# Load example data
data(spectra) # list of MassSpectra class objects

# Some pre-processing
spectra <- screenSpectra(spectra)$fspectra
```
writeMetadata

```
spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)
peaks <- alignPeaks(peaks, minFreq = 0.8)

# Intensity matrix
int <- intensityMatrix(peaks)

# default R format
writeIntensity(int, file="int")

# binary NEXUS file
writeIntensity(int, file="int.binary", format="NEXUS", interleaved=FALSE)
```

writeMetadata | Write metadata in different formats

Description

This function is simply a wrapper to write the metadata associated with a collection of mass spectra into a file in either the R or csv format.

Usage

writeMetadata(x, filename = "Metadata", format = c("R", "csv"), ...)

Arguments

- **x**: Metadata in any sensible data format, preferably `matrix` or `data.frame`.
- **filename**: A character string specifying a name for the destination file (filename extension not required).
- **format**: One of R (default `.Rdata` file) or text (comma-separated `.csv` file).
- **...**: Other arguments.

Details

It uses either `save` or `write.table` to store the metadata. Check these functions for adequate data formats.
Examples

# Load example data
data(spectra) # list of MassSpectra class objects
data(type) # metadata

# Some pre-processing
sc.spectra <- screenSpectra(spectra, meta = type)
spectra <- sc.spectra$spectra # filtered spectra
type <- sc.spectra$meta # filtered metadata

spectra <- transformIntensity(spectra, method = "sqrt")
spectra <- wavSmoothing(spectra)
spectra <- removeBaseline(spectra)
peaks <- detectPeaks(spectra)
peaks <- alignPeaks(peaks, minFreq = 0.8)

# Intensity matrix
int <- intensityMatrix(peaks)

# Save resulting data in R format
writeIntensity(int, filename = "MyintMatrix")
writeMetadata(type, filename = "MyMetadata")

# Save resulting data in csv format
writeIntensity(int, filename = "MyintMatrix", format = "csv")
writeMetadata(type, filename = "MyMetadata", format = "csv")
Index

*Topic \textasciitilde kwd1
 alignPeaks, 3
countPeaks, 4
deletePeaks, 5
detectOutliers, 6
importSpectra, 7
plot.scSpectra, 10
rawToPeaks, 11
rawToSpectra, 12
screenSpectra, 14
snrPeaks, 16
summary.scSpectra, 17
summaryPeaks, 18
summarySpectra, 19
wavSmoothing, 22
writeIntensity, 23
writeMetadata, 24

*Topic \textasciitilde kwd2
 alignPeaks, 3
countPeaks, 4
deletePeaks, 5
detectOutliers, 6
importSpectra, 7
plot.scSpectra, 10
rawToPeaks, 11
rawToSpectra, 12
screenSpectra, 14
snrPeaks, 16
summary.scSpectra, 17
summaryPeaks, 18
summarySpectra, 19
wavSmoothing, 22
writeIntensity, 23
writeMetadata, 24

*Topic datasets
 spectra, 17
type, 21

*Topic mass spectrometry, pre-processing, robust

statistics
 MALDIrppa-package, 2
   addMetadata, 2
   alignPeaks, 3
   binPeaks, 4
   countPeaks, 4
   covMcd, 6
   data.frame, 7, 9, 18, 20, 24
   deletePeaks, 5
   detectOutliers, 6
   detectPeaks, 16
   dist, 6
   factor, 6
   importSpectra, 7, 12
   intensityMatrix, 9, 23
   MALDIquant, 2–4
   MALDIrppa (MALDIrppa-package), 2
   MALDIrppa-package, 2
   MassPeaks, 3–5, 9, 11, 12, 16, 18
   MassSpectrum, 6–8, 12–14, 17, 19–22
   matrix, 9, 24
   numeric, 6
   par, 9
   peakPatterns, 9
   plot.scSpectra, 10, 15, 18
   Qn, 14
   rawToPeaks, 11
   rawToSpectra, 12
   redResolution, 13
   referencePeaks, 4
INDEX

save, 24
screenSpectra, 10, 11, 14, 17, 18
snrPeaks, 16
spectra, 17, 21
summary.scSpectra, 11, 15, 17
summaryPeaks, 18
summarySpectra, 19

transfIntensity, 20
transformIntensity, 21
type, 17, 21

warpMassPeaks, 4
wavShrink, 22
wavSmoothing, 22
write.table, 24
writeIntensity, 23
writeMetadata, 24