Package ‘MassSpecWavelet’

March 7, 2024

Type Package
Title Peak Detection for Mass Spectrometry data using wavelet-based algorithms
Version 1.68.0
Encoding UTF-8
Date 2023-04-11

Description Peak Detection in Mass Spectrometry data is one of the important preprocessing steps. The performance of peak detection affects subsequent processes, including protein identification, profile alignment and biomarker identification. Using Continuous Wavelet Transform (CWT), this package provides a reliable algorithm for peak detection that does not require any type of smoothing or previous baseline correction method, providing more consistent results for different spectra. See <doi:10.1093/bioinformatics/btl355} for further details.

URL https://github.com/zeehio/MassSpecWavelet

BugReports http://github.com/zeehio/MassSpecWavelet/issues
License LGPL (>= 2)

Suggests signal, waveslim, BiocStyle, knitr, rmarkdown, RUnit, bench

biocViews ImmunoOncology, MassSpectrometry, Proteomics, PeakDetection

VignetteBuilder knitr

RoxygenNote 7.2.1

Roxygen list(markdown = TRUE)

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

git_branch RELEASE_3_18

git_last_commit 99ece70

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-03-06
Author Pan Du [aut],
   Warren Kibbe [aut],
   Simon Lin [aut],
   Sergio Oller Moreno [aut, cre]
   (https://orcid.org/0000-0002-8994-1549)

Maintainer Sergio Oller Moreno <sergioller@gmail.com>

R topics documented:

- MassSpecWavelet-package
- cwt
- exampleMS
- extendLength
- extendNBase
- findLocalMaxWinSize
- getLocalMaximumCWT
- getRidge
- getRidgeLength
- getRidgeValue
- identifyMajorPeaks
- localMaximum
- mexh
- mzInd2vRange
- mzV2indRange
- peakDetectionCWT
- plotLocalMax
- plotPeak
- plotRidgeList
- prepareWavelets
- smoothDWT
- tuneInPeakInfo

Index

Description

MassSpecWavelet R package is aimed to detect peaks on Mass Spectrometry (MS) data using Continuous Wavelet Transform (CWT).

Author(s)

Pan Du, Simon Lin
References


See Also

Useful links:

- https://github.com/zeehio/MassSpecWavelet

Examples

data(exampleMS)
SNR.Th <- 3
peakInfo <- peakDetectionCWT(exampleMS, SNR.Th = SNR.Th)
majorPeakInfo <- peakInfo$majorPeakInfo
peakIndex <- majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))

---

cwt

Continuous Wavelet Transform (CWT)

Description

CWT(Continuous Wavelet Transform) with Mexican Hat wavelet (by default) to match the peaks in Mass Spectrometry spectrum

Usage

cwt(ms, scales = 1, wavelet = "mexh")

Arguments

ms Mass Spectrometry spectrum (a vector of MS intensities)
scales a vector represents the scales at which to perform CWT. See the Details section. Additionally, a prepared_wavelets object is also accepted (see prepareWavelets()).
wavelet The wavelet base, Mexican Hat by default. User can provide wavelet \( \psi(x) \) as a form of two row matrix. The first row is the \( x \) value, and the second row is \( \psi(x) \) corresponding to \( x \).
Details

The default mother wavelet is a Mexican Hat evaluated in the $[-8, 8]$ range using 1024 points (a step of roughly 1/64):

$$\psi(x) = \frac{2}{\sqrt{3}} \pi^{-0.25} (1 - x^2) \exp(-x^2/2)$$

The $\sigma$ of the mother Mexican Hat is of one $x$ unit.

The scaled wavelet is a downsampled version of the mother wavelet. The scale determines how many samples per $x$ unit are taken. For instance, with the default Mexican Hat wavelet, a scales $= 2$ will evaluate the $[-8, 8]$ range sampling twice per $x$ unit, this is with a step of 0.5. This generates a 33 points long scaled wavelet. Choosing this type of scaling is convenient because the scaled wavelet becomes a wavelet of $\sigma = '{scales}'$ points. Using the default wavelet, a scales smaller than 1 will show sampling issues, while a scales larger than 64 will resample points from the original mother wavelet. If you need to use scales larger than 64, consider providing your own mother wavelet. See the examples.

According to doi:10.1063/1.3505103, if your spectrum has a gaussian peak shape of variance $m^2$ points then the scales range should cover $[1, 1.9m]$. If your spectrum has a Lorentzian peak shape of half-width-half-maximum $L$ points then the scales range should cover $[1, 2.9L]$. They also suggest using a log$_{1.18}$ spacing. Take these values as suggestions for your analysis.

Value

The return is the 2-D CWT coefficient matrix, with column names as the scale. Each column is the CWT coefficients at that scale. If the scales are too big for the given signal, the returned matrix may include less columns than the given scales.

Author(s)

Pan Du, Simon Lin

Examples

data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = "mexh")

## Plot the 2-D CWT coefficients as image (It may take a while!)
xTickInterval <- 1000
image(5000:11000, scales, wCoefs,
    col = terrain.colors(256), axes = FALSE,
    xlab = "m/z index", ylab = "CWT coefficient scale", main = "CWT coefficients"
)
axis(1, at = seq(5000, 11000, by = xTickInterval))
axis(2, at = c(1, seq(10, 64, by = 10)))
box()

## Provide a larger wavelet:
scales <- c(seq(1, 64, 3), seq(72, 128, 8))
psi_xval <- seq(-8, 8, length.out = 4096)
exampleMS <- (2 / sqrt(3) * pi^(-0.25)) * (1 - psi_xval^2) * exp(-psi_xval^2 / 2)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = rbind(psi_xval, psi))
xTickInterval <- 1000
image(5000:11000, scales, wCoefs,
    col = terrain.colors(256), axes = FALSE,
    xlab = "m/z index", ylab = "CWT coefficient scale", main = "CWT coefficients"
)
axis(1, at = seq(5000, 11000, by = xTickInterval))
axis(2, at = c(1, seq(10, 128, by = 10)))
box()

## Custom log1.18 spaced scales:
get_scales <- function(scale_min, scale_max, num_scales) {
    (seq(0, 1, length.out = num_scales)^1.18) * (scale_max - scale_min) + scale_min
}
scales <- get_scales(scale_min = 1, scale_max = 64, num_scales = 32)
print(scales)
# For detecting a gaussian peak of 10 points:
gaussian_peak_sigma <- 10 # points
scales <- get_scales(scale_min = 1, scale_max = 1.9 * gaussian_peak_sigma, num_scales = 32)
print(scales)
# For detecting a lorentzian peak of 10 points:
lorentzian_peak_gamma <- 10 # points
scales <- get_scales(scale_min = 1, scale_max = 2.9 * lorentzian_peak_gamma, num_scales = 32)
print(scales)
extendLength  

Extend the length of a signal or matrix

Description

Extend the length of a signal or matrix by row

Usage

```r
extendLength(
  x,
  addLength = NULL,
  method = c("reflection", "open", "circular"),
  direction = c("right", "left", "both")
)
```

Arguments

- `x`: a vector or matrix with column with each column as a signal
- `addLength`: the length to be extended
- `method`: three methods available, c("reflection", "open", "circular"). By default, it is "reflection".
- `direction`: three options available: c("right", "left", "both")

Value

The extended vector or matrix.

Author(s)

Pan Du

See Also

`extendNBase()`

Examples

```r
a = matrix(rnorm(9), 3)
extendLength(a, 3, direction='right')
```
**extendNBase**

Extend the row number of a matrix as the exponential of base N

**Description**

Extend the data as the exponential of base N by increasing row number.

**Usage**

```r
extendNBase(x, nLevel = 1, base = 2, ...)
```

**Arguments**

- `x`: data matrix
- `nLevel`: the level of DWT decomposition. Basically, it is equivalent to changing the 'base' as base\^nLevel
- `base`: the base, 2 by default
- `...`: other parameters of used by `extendLength()`

**Details**

The method 'open' is padding the the matrix with the last row.

**Value**

Return a extended matrix

**Author(s)**

Pan Du

**See Also**

`extendLength()`

**Examples**

```r
a <- matrix(rnorm(9), 3)
MassSpecWavelet:::extendNBase(a)
```
findLocalMaxWinSize  

Find local maxima and return the size of the window where they are maximum.

Description

Compared to the rest of the package, this is a rather experimental function. If you plan to use it or are interested in it, please open an issue at https://github.com/zeehio/MassSpecWavelet/issues to show your interest.

Usage

findLocalMaxWinSize(x, capWinSize = NA)

Arguments

x  
A numeric vector.

capWinSize  
the maximum window size to report. NA means unlimited.

Value

An integer vector y of the same length as x. y[i] will be the size of the largest window on x containing x[i] where:

• x[i] is a local maximum or a center of a plateau

• x[i] is not at a window border Optionally, if capWinSize is a positive integer, the maximum window size is capped to that value, to increase performance. Use this in case you just want to check if there exists a window of that size. 

getLocalMaximumCWT  
Identify the local maximum of each column in 2-D CWT coefficients matrix

Description

Identify the local maximum of each column in 2-D CWT coefficients matrix by using a slide window. The size of slide window linearly changes from the coarse scale (bigger window size) to detail scale. The scale of CWT increases with the column index.
getLocalMaximumCWT

Usage

```r
getLocalMaximumCWT(
  wCoefs,
  minWinSize = 5,
  amp.Th = 0,
  isAmpThreshRelative = FALSE,
  exclude0scaleAmpThresh = FALSE
)
```

Arguments

- **wCoefs**: 2-D CWT coefficients, each column corresponding to CWT coefficient at one scale. The column name is the scale.
- **minWinSize**: The minimum slide window size used.
- **amp.Th**: The minimum peak amplitude.
- **isAmpThreshRelative**: Whether `amp.Th` is given relative to `max(wCoefs)`.
- **exclude0scaleAmpThresh**: When computing the relative `amp.Th`, if this is set to `TRUE`, the `amp.Th` will exclude the zero-th scale from the `max(wCoefs)`. The zero-th scale corresponds to the original signal, that may have a much larger baseline than the wavelet coefficients and can distort the threshold calculation. The default is `FALSE` to preserve backwards compatibility.

Value

return a matrix with same dimension as CWT coefficient matrix, `wCoefs`. The local maxima are marked as 1, others are 0.

Author(s)

Pan Du

See Also

- localMaximum()

Examples

```r
data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = "mexh")

localMax <- getLocalMaximumCWT(wCoefs)
plotLocalMax(localMax)
```
getRidge

Identify ridges based on the local maximum matrix

Description

Identify ridges by connecting the local maximum of 2-D CWT coefficients from the coarse scale to detail scale. The local maximum matrix is returned from `getLocalMaximumCWT()`.

Usage

```r
getRidge(
  localMax,
  iInit = ncol(localMax),
  step = -1,
  iFinal = 1,
  minWinSize = 5,
  gapTh = 3,
  skip = NULL,
  scaleToWinSize = "doubleodd"
)
```

Arguments

- `localMax`: The local maximum matrix is returned from `getLocalMaximumCWT()` with 1 represents maximum, others are 0.
- `iInit`: The start column to search ridge. By default, it starts from the coarsest scale level.
- `step`: Search step. -1 by default, which means searching from coarse scale to detail scale column by column.
- `iFinal`: The final column index of search ridge.
- `minWinSize`: The minimum slide window size used.
- `gapTh`: The gap allowed during searching for ridge. 3 by default.
- `skip`: The column to be skipped during search.
- `scaleToWinSize`: How scales should be mapped to window sizes. Traditionally, MassSpecWavelet used the "doubleodd" mapping (`winSize <- 2*scale+1`). xcms switched this mapping to "halve" (`winSize <- floor(scale/2)`). Besides "doubleodd" and "halve" this parameter can also be a custom function of the scale.

Value

Return a list of ridge. As some ridges may end at the scale larger than 1, in order to keep the uniqueness of the ridge names, we combined the smallest scale of the ridge and m/z index of the peak at that scale together to name the ridges. For example the ridge name "1\_653" means the peak ridge ends at the CWT scale 1 with m/z index 653 at scale 1.
getRidgeLength

Author(s)
Pan Du, Simon Lin

References

See Also
getLocalMaximumCWT(), identifyMajorPeaks()

Examples
data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = "mexh")

localMax <- getLocalMaximumCWT(wCoefs)
ridgeList <- getRidge(localMax)
plotRidgeList(ridgeList)

getRidgeLength

Estimate the length of the ridge

Description
Estimate the length of the ridge line, which is composed of local maxima at adjacent CWT scales. The ridge line is cut off at the end point, whose amplitude divided by the maximum ridge amplitude is larger than the cutoff amplitude ratio threshold (0.5 by default).

Usage
getRidgeLength(ridgeList, Th = 0.5)

Arguments
ridgeList a list of identified ridges
Th the cutoff amplitude ratio (the amplitude divided by the maximum amplitude of the ridge) threshold of the ridge line end.

Value
a vector of estimated ridge length

Author(s)
Pan Du
identifyMajorPeaks

Examples

stopifnot(getRidgeLength(list(c(5,4,3,2,1), c(5,3,1))) == c(3,2))

getRidgeValue

Get the CWT coefficient values corresponding to the peak ridge

Description

Get the CWT coefficient values corresponding to the peak ridge

Usage

getRidgeValue(ridgeList, wCoefs, skip = 0)

Arguments

ridgeList a list of ridge lines
wCoefs 2-D CWT coefficients
skip the CWT scale level to be skipped, by default the 0 scale level (raw spectrum) is skipped.

Value

A list of ridge values corresponding to the input ridgeList.

Author(s)

Pan Du

identifyMajorPeaks

Identify peaks based on the ridges in 2-D CWT coefficient matrix

Description

Identify the peaks based on the ridge list (returned by getRidge()) in 2-D CWT coefficient matrix and estimated Signal to Noise Ratio (SNR)
**identifyMajorPeaks**

**Usage**

```r
identifyMajorPeaks(
  ms,
  ridgeList,
  wCoefs,
  scales = as.numeric(colnames(wCoefs)),
  SNR.Th = 3,
  peakScaleRange = 5,
  ridgeLength = 32,
  nearbyPeak = FALSE,
  nearbyWinSize = ifelse(nearbyPeak, 150, 100),
  winSize.noise = 500,
  SNR.method = "quantile",
  minNoiseLevel = 0.001,
  excludeBoundariesSize = nearbyWinSize/2
)
```

**Arguments**

- `ms`: the mass spectrometry spectrum
- `ridgeList`: returned by `getRidge()`
- `wCoefs`: 2-D CWT coefficients
- `scales`: scales of CWT, by default it is the colnames of wCoefs
- `SNR.Th`: threshold of SNR
- `peakScaleRange`: the CWT scale range of the peak.
- `ridgeLength`: the maximum ridge scale of the major peaks.
- `nearbyPeak`: determine whether to include the small peaks close to large major peaks
- `nearbyWinSize`: the window size to determine the nearby peaks. Only effective when nearbyPeak is true.
- `winSize.noise`: the local window size to estimate the noise level.
- `SNR.method`: method to estimate noise level. Currently, only 95 percentage quantile is supported.
- `minNoiseLevel`: the minimum noise level used in calculating SNR, i.e., if the estimated noise level is less than "minNoiseLevel", it will use "minNoiseLevel" instead. If the noise level is less than 0.5, it will be treated as the ratio to the maximum amplitude of the spectrum.
- `excludeBoundariesSize`: number of points at each boundary of the ms signal that will be excluded in search for peaks to avoid boundary effects.

**Details**

The determination of the peaks is based on three rules: Rule 1: The maximum ridge scale of the peak should larger than a certain threshold Rule 1.1: Based on the scale of the peak (corresponding to the maximum value of the peak ridge) should be within certain range Rule 2: Based on the peak SNR Rule 3: The peak should not appear at the boundaries of the signal.
**Value**

Return a list with following elements:

- `peakIndex` the m/z indexes of the identified peaks
- `peakCenterIndex` the m/z indexes of peak centers, which correspond to the maximum on the ridge. `peakCenterIndex` includes all the peaks, not just the identified major peaks.
- `peakCenterValue` the CWT coefficients (the maximum on the ridge) corresponding to `peakCenterIndex`
- `peakSNR` the SNR of the peak, which is the ratio of `peakCenterValue` and noise level
- `peakScale` the estimated scale of the peak, which corresponds to the `peakCenterIndex`
- `potentialPeakIndex` the m/z indexes of all potential peaks, which satisfy all requirements of a peak without considering its SNR. Useful, if you want to change to a lower SNR threshold later.
- `allPeakIndex` the m/z indexes of all the peaks, whose order is the same as `peakCenterIndex`, `peakCenterValue`, `peakSNR` and `peakScale`.

All of these return elements have peak names, which are the same as the corresponding peak ridges. See `getRidge()` for details.

**Author(s)**

Pan Du, Simon Lin

**References**


**See Also**

`peakDetectionCWT()`, `tuneInPeakInfo()`

**Examples**

```r
data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS, scales = scales, wavelet = "mexh")

localMax <- getLocalMaximumCWT(wCoefs)
ridgeList <- getRidge(localMax)

SNR.Th <- 3
majorPeakInfo <- identifyMajorPeaks(exampleMS, ridgeList, wCoefs, SNR.Th = SNR.Th)

## Plot the identified peaks
peakIndex <- majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))```
localMaximum

Identify local maximum within a slide window.

Description

The simplest local maximum detection using a sliding window searches for maxima in a window of a given size, and slides that window across the signal, shifting it one position at a time.

Usage

localMaximum(x, winSize = 5)

Arguments

- x: a vector represents a signal profile
- winSize: the slide window size, 5 by default.

Details

The default implementation found here shifts the window by half of its size instead of by one position at a time. This makes the implementation faster, at the expense of not being able to detect peaks that are too close to each other, if they appear in some positions with respect to the windows.

Additionally, this implementation removes all instances of peaks found at a distance less than the window size.

Experimentally, we are exploring other algorithms for local maxima detection. These algorithms can be chosen setting the "MassSpecWavelet.localMaximum.algorithm" option. See the "Finding local maxima" vignette for further details.

Value

Return a vector with the same length of the input x. The position of local maximum is set as 1L, 0L elsewhere.

Author(s)

Pan Du and Sergio Oller

See Also

getLocalMaximumCWT()
Examples

```r
x <- rnorm(200)
lmax <- localMaximum(x, 5)
maxInd <- which(lmax > 0)
plot(x, type = "l")
points(maxInd, x[maxInd], col = "red")
```

---

**mexh**

*The mexican hat function*

**Description**

\[
\psi(x) = \frac{2}{\sqrt{3}} \pi^{-0.25}(1 - x^2) \exp(-x^2/2)
\]

**Usage**

```r
mexh(x)
```

**Arguments**

- **x**
  where to evaluate the mexican hat

**Value**

A vector of the same length as `x` with the corresponding values

**Examples**

```r
x <- seq(-8, 8, length.out = 256)
mexh(x)
```

---

**mzInd2vRange**

*Match m/z index to m/z value with a certain error range*

**Description**

Match m/z index to m/z value with a certain error range

**Usage**

```r
mzInd2vRange(mzInd, error = 0.003)
```
mzV2indRange

Arguments

mzInd a vector of m/z index
error error range

Value

return a vector of sorted m/z values

Author(s)

Pan Du

See Also

mzV2indRange()

Description

Match m/z value to m/z index with a certain error range

Usage

mzV2indRange(mzV, error = 0.003)

Arguments

mzV a vector of m/z value
error error range

Value

return a vector of sorted m/z indexes

Author(s)

Pan Du

See Also

mzInd2vRange()
## peakDetectionCWT

The main function of peak detection by CWT based pattern matching

### Description

This function is a wrapper of `cwt()`, `getLocalMaximumCWT()`, `getRidge()`, `identifyMajorPeaks()`.

### Usage

```r
peakDetectionCWT(
  ms,
  scales = c(1, seq(2, 30, 2), seq(32, 64, 4)),
  SNR.Th = 3,
  nearbyPeak = TRUE,
  peakScaleRange = 5,
  amp.Th = 0.01,
  minNoiseLevel = amp.Th/SNR.Th,
  ridgeLength = 24,
  peakThr = NULL,
  tuneIn = FALSE,
  ...,
  exclude0scaleAmpThresh = FALSE,
  getRidgeParams = list(gapTh = 3, skip = 2)
)
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ms</code></td>
<td>the mass spectrometry spectrum</td>
</tr>
<tr>
<td><code>scales</code></td>
<td>Scales of CWT. See <code>cwt()</code> for details. Additionally, a <code>prepared_wavelets</code> object is also accepted (see <code>prepareWavelets()</code>).</td>
</tr>
<tr>
<td><code>SNR.Th</code></td>
<td>SNR (Signal to Noise Ratio) threshold</td>
</tr>
<tr>
<td><code>nearbyPeak</code></td>
<td>Determine whether to include the nearby small peaks of major peaks. TRUE by default</td>
</tr>
<tr>
<td><code>peakScaleRange</code></td>
<td>the scale range of the peak. larger than 5 by default.</td>
</tr>
<tr>
<td><code>amp.Th</code></td>
<td>the minimum required relative amplitude of the peak (ratio to the maximum of CWT coefficients)</td>
</tr>
<tr>
<td><code>minNoiseLevel</code></td>
<td>the minimum noise level used in computing the SNR</td>
</tr>
<tr>
<td><code>ridgeLength</code></td>
<td>the minimum highest scale of the peak in 2-D CWT coefficient matrix</td>
</tr>
<tr>
<td><code>peakThr</code></td>
<td>Minimal absolute intensity (above the baseline) of peaks to be picked. If this value is provided, then the smoothing function <code>signal::sgolayfilt()</code> will be called to estimate the local intensity. (added based on the suggestion and code of Steffen Neumann)</td>
</tr>
<tr>
<td><code>tuneIn</code></td>
<td></td>
</tr>
<tr>
<td><code>...</code></td>
<td></td>
</tr>
<tr>
<td><code>exclude0scaleAmpThresh</code></td>
<td></td>
</tr>
<tr>
<td><code>getRidgeParams</code></td>
<td>list(gapTh = 3, skip = 2)</td>
</tr>
</tbody>
</table>

### Description

This function is a wrapper of `cwt()`, `getLocalMaximumCWT()`, `getRidge()`, `identifyMajorPeaks()`.
peakDetectionCWT

tuneIn determine whether to tune in the parameter estimation of the detected peaks. If TRUE, peak detection is run again on a segment of the spectrum with more detailed scales. This tuning happens with the default wavelet and settings so it may not be that useful to you if you are using custom wavelets or thresholds.

... other parameters used by identifyMajorPeaks(). Additionally, fl (filter length, with a default value of 1001) and forder (filter order, with a default value of 2) are set and passed to signal::sgolayfilt() when peakThr is given.

exclude0scaleAmpThresh When computing the relative amp.Th, if this is set to TRUE, the amp.Th will exclude the zero-th scale from the max(wCoefs). The zero-th scale corresponds to the original signal, that may have a much larger baseline than the wavelet coefficients and can distort the threshold calculation. The default is FALSE to preserve backwards compatibility.

getRidgeParams A list with parameters for getRidge().

Value

- majorPeakInfo return of identifyMajorPeaks()
- ridgeList return of getRidge()
- localMax return of getLocalMaximumCWT()
- wCoefs 2-D CWT coefficient matrix, see cwt() for details.

Author(s)

Pan Du, Simon Lin

References


See Also

cwt(), getLocalMaximumCWT(), getRidge(), identifyMajorPeaks()

Examples

data(exampleMS)

# Detect peaks with prepared wavelets:
prep_wav <- prepareWavelets(length(exampleMS))
SNR.Th <- 3
peakInfo <- peakDetectionCWT(exampleMS, prep_wav, SNR.Th = SNR.Th, exclude0scaleAmpThresh=TRUE)
peakIndex <- peakInfo$majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))

SNR.Th <- 3
peakInfo <- peakDetectionCWT(exampleMS, SNR.Th = SNR.Th)
majorPeakInfo <- peakInfo$majorPeakInfo
plotLocalMax <- majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))

## In some cases, users may want to add peak filtering based on the absolute peak amplitude
peakInfo <- peakDetectionCWT(exampleMS, SNR.Th = SNR.Th, peakThr = 500)
majorPeakInfo <- peakInfo$majorPeakInfo
peakIndex <- majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))

---

### plotLocalMax

**Plot the local maximum matrix**

**Description**

Plot the local maximum matrix of 2-D CWT coefficients returned by `getLocalMaximumCWT()`

**Usage**

```r
plotLocalMax(
  localMax,
  wCoefs = NULL,
  range = c(1, nrow(localMax)),
  colorMap = "RYB",
  main = NULL,
  cex = 3,
  pch = ".",
  ...
)
```

**Arguments**

- `localMax`: local maximum matrix of 2-D CWT coefficients returned by `getLocalMaximumCWT()`
- `wCoefs`: 2-D CWT coefficients
- `range`: plot range of m/z index
- `colorMap`: the colormap used in plotting the points
- `main`: parameter of `plot()`
- `cex`: parameter of `plot()`
- `pch`: parameter of `plot()`
- `...`: other parameters of `points()`

**Value**

No value is returned; this function is called for its side effects (plot).
Author(s)

Pan Du

See Also

`getLocalMaximumCWT()`

Examples

data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = "mexh")

localMax <- getLocalMaximumCWT(wCoefs)
plotLocalMax(localMax)

Description

Plot the identified peaks over the spectrum. The identified peaks are returned by `peakDetectionCWT()` or `identifyMajorPeaks()`

Usage

```r
plotPeak(
  ms,
  peakIndex = NULL,
  mz = 1:length(ms),
  range = c(min(mz), max(mz)),
  method = c("p", "l"),
  main = NULL,
  log = "",
  ...
)
```

Arguments

- `ms` the MS spectrum
- `peakIndex` m/z indexes of the identified peaks
- `mz` m/z value correspond to m/z index
- `range` the plot range of m/z value
- `method` plot method of the identified peaks. method 'p' plot circles on the peaks; method 'l' add vertical lines over the peaks.
plotRidgeList

main parameter of `plot()`
log parameter of `plot()`
... other parameters of `points()`

Value

No value is returned; this function is called for its side effects (plot).

Author(s)

Pan Du

See Also

`peakDetectionCWT()`, `identifyMajorPeaks()`

Examples

data(exampleMS)
SNR.Th <- 3
peakInfo <- peakDetectionCWT(exampleMS, SNR.Th = SNR.Th)
majorPeakInfo <- peakInfo$majorPeakInfo
peakIndex <- majorPeakInfo$peakIndex
plotPeak(exampleMS, peakIndex, main = paste("Identified peaks with SNR >", SNR.Th))
Arguments

ridgeList returned by `getRidge()`
wCoefs 2-D CWT coefficients
range plot range of m/z index
colorMap colorMap to plot the points of local maximum
main parameter of `plot()`
pch parameter of `plot()`
cex parameter of `plot()`
... other parameters of `points()`

Value

No value is returned; this function is called for its side effects (plot).

Author(s)

Pan Du

See Also

`getRidge()`

Examples

data(exampleMS)
scales <- seq(1, 64, 3)
wCoefs <- cwt(exampleMS[5000:11000], scales = scales, wavelet = "mexh")

localMax <- getLocalMaximumCWT(wCoefs)
ridgeList <- getRidge(localMax)
plotRidgeList(ridgeList)
prepareWavelets

Usage

prepareWavelets(
  mslength,
  scales = c(1, seq(2, 30, 2), seq(32, 64, 4)),
  wavelet = "mexh",
  wavelet_xlimit = 8,
  wavelet_length = 1024L,
  extendLengthScales = TRUE
)

Arguments

- **mslength**: Length of the signal to transform
- **scales**: a vector represents the scales at which to perform CWT. See the Details section. Additionally, a prepared_wavelets object is also accepted (see `prepareWavelets()`).
- **wavelet**: The wavelet base, Mexican Hat by default. User can provide wavelet $\Psi(x)$ as a form of two row matrix. The first row is the $x$ value, and the second row is $\Psi(x)$ corresponding to $x$.
- **wavelet_xlimit**: The mother wavelet will be evaluated between these limits. Ignored if `wavelet` is a matrix.
- **wavelet_length**: The number of points of the mother wavelet. Ignored if `wavelet` is a matrix.
- **extendLengthScales**: A logical value. If the signal is too short, we may need to pad it to convolve it with larger daughter wavelets. Set this to `TRUE` to let scales be used to determine the padding length. It’s set to `FALSE` by default to preserve backwards compatibility.

Value

A prepared_wavelets object.

See Also

cwt

Examples

```r
x <- runif(2000)
scales <- c(1, 2, 4, 8)
prep_wavelets <- prepareWavelets(length(x), scales = scales)
wCoefs <- cwt(x, prep_wavelets)
```
Description

Smooth (denoise) the spectrum by DWT (Discrete Wavelet Transform)

Usage

smoothDWT(
  ms,
  nLevel = 6,
  wf = "la8",
  localNoiseTh = seq(1, 0, by = -0.2),
  localWinSize = 500,
  globalNoiseTh = 0.75,
  smoothMethod = c("soft", "hard"),
  method = c("dwt", "modwt")
)

Arguments

ms        a vector representing the mass spectrum
nLevel    the level of DWT decomposition
wf        the name of wavelet for DWT
localNoiseTh local noise level threshold
localWinSize local window size for estimate local noise threshold
globalNoiseTh global noise level threshold
smoothMethod the method used for denoising. 'hard' means keeping the dwt coefficients higher than the threshold unchanged; "soft" means the dwt coefficients higher than the threshold were subtracted by the threshold.
method    'dwt' or 'modwt' used for decomposition

Value

return the smoothed mass spectrum with the 'detail' component of DWT as an attribute 'detail'.

Author(s)

Pan Du
Description

Based on the identified peak position, more precise estimation of the peak information, i.e., peak position and peak scale, can be got by this function. The basic idea is to cut the segment of spectrum near the identified peaks, and then do similar procedures as `peakDetectionCWT()`, but with more detailed scales around the estimated peak scale.

Usage

tuneInPeakInfo(
  ms,
  majorPeakInfo = NULL,
  peakIndex = NULL,
  peakScale = NULL,
  maxScale = 128,
  ...
)

Arguments

ms    the mass spectrometry spectrum
majorPeakInfo return of `identifyMajorPeaks()`
peakIndex the m/z index of the identified peaks
peakScale the scales of the identified peaks
maxScale the maximum scale allowed for the peak
... other parameters of used by `getLocalMaximumCWT(),getRidge(),identifyMajorPeaks()`

Details

The majorPeakInfo or peakIndex and peakScale must be provided.

Value

peakCenterIndex the updated peak center m/z index
peakScale the updated peak scale
peakValue the corresponding peak value

Author(s)

Pan Du
tuneInPeakInfo

References


See Also

peakDetectionCWT()

Examples

data(exampleMS)
SNR.Th <- 3
peakInfo <- peakDetectionCWT(exampleMS, SNR.Th = SNR.Th)
majorPeakInfo <- peakInfo$majorPeakInfo
betterPeakInfo <- tuneInPeakInfo(exampleMS, majorPeakInfo)
plot(500:length(exampleMS), exampleMS[500:length(exampleMS)], type = "l", log = "x")
abline(v = betterPeakInfo$peakCenterIndex, col = "red")
Index

* datasets
  exampleMS, 5
* hplot
  plotLocalMax, 20
  plotPeak, 21
  plotRidgeList, 22
* internal
  extendLength, 6
  extendNBase, 7
* methods
  cwt, 3
  getLocalMaximumCWT, 8
  getRidge, 10
  getRidgeLength, 11
  getRidgeValue, 12
  identifyMajorPeaks, 12
  identifyMajorPeaks(), 11, 18, 19, 21, 22, 26
  localMaximum, 15
  localMaximum(), 9
  MassSpecWavelet
    (MassSpecWavelet-package), 2
    MassSpecWavelet-package, 2
    mexh, 16
    mzInd2vRange, 16
    mzInd2vRange(), 17
    mzV2indRange, 17
    mzV2indRange(), 17
    peakDetectionCWT, 18
    peakDetectionCWT(), 14, 21, 22, 26, 27
    plot(), 20, 22, 23
    plotLocalMax, 20
    plotPeak, 21
    plotRidgeList, 22
    points(), 20, 22, 23
    prepareWavelets, 23
    prepareWavelets(), 3, 18, 24
    signal::sgolayfilt(), 18, 19
    smoothDWT, 25
    tuneInPeakInfo, 26
    tuneInPeakInfo(), 14
  cwt(), 18, 19
  exampleMS, 5
  extendLength, 6
  extendLength(), 7
  extendNBase, 7
  extendNBase(), 6
  findLocalMaxWinSize, 8
  getLocalMaximumCWT, 8
  getLocalMaximumCWT(), 10, 11, 15, 18–21, 26
  getRidge, 10
  getRidge(), 12–14, 18, 19, 22, 23, 26
  getRidgeLength, 11
  getRidgeValue, 12
  identifyMajorPeaks, 12
  identifyMajorPeaks(), 11, 18, 19, 21, 22, 26
  localMaximum, 15
  localMaximum(), 9
  MassSpecWavelet
    (MassSpecWavelet-package), 2
    MassSpecWavelet-package, 2
    mexh, 16
    mzInd2vRange, 16
    mzInd2vRange(), 17
    mzV2indRange, 17
    mzV2indRange(), 17
    peakDetectionCWT, 18
    peakDetectionCWT(), 14, 21, 22, 26, 27
    plot(), 20, 22, 23
    plotLocalMax, 20
    plotPeak, 21
    plotRidgeList, 22
    points(), 20, 22, 23
    prepareWavelets, 23
    prepareWavelets(), 3, 18, 24
    signal::sgolayfilt(), 18, 19
    smoothDWT, 25
    tuneInPeakInfo, 26
    tuneInPeakInfo(), 14