Package 'MassSpecWavelet'

November 12, 2026

Type Package

Title Peak Detection for Mass Spectrometry data using wavelet-based algorithms

Version 1.78.2

Encoding UTF-8

Date 2026-07-05

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 7.3.2

Roxygen list(markdown = TRUE)

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

git_branch RELEASE_3_23

git_last_commit 37682e0

git_last_commit_date 2026-07-06

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Pan Du [aut],
Warren Kibbe [aut],
Simon Lin [aut],
Sergio Oller Moreno [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8994-1549>)

Maintainer Sergio Oller Moreno <sergioller@gmail.com>

Contents

MassSpecWavelet-package 2
cwt 3
exampleMS 5
extendLength 6
extendNBase 7
findLocalMaxWinSize 8
getLocalMaximumCWT 8
getRidge 10
getRidgeLength 11
getRidgeValue 12
identifyMajorPeaks 12
localMaximum 16
mexh 17
mzInd2vRange 17
mzV2indRange 18
peakDetectionCWT 19
plotLocalMax 21
plotPeak 22
plotRidgeList 23
prepareWavelets 24
smoothDWT 26
tuneInPeakInfo 27

Index **29**

`MassSpecWavelet-package`
Peak detection of mass spectrum by Wavelet transform based methods

---

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

Du, P., Kibbe, W.A. and Lin, S.M. (2006) Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching, Bioinformatics, 22, 2059-2065.

See Also

Useful links:

- https://github.com/zeehio/MassSpecWavelet
- Report bugs at http://github.com/zeehio/MassSpecWavelet/issues

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 σ 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)
```

```
psi <- (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)
```

---

exampleMS *An example mass spectrum*

Description

An example mass spectrum from CAMDA 2006. All-in-1 Protein Standard II (Ciphergen Cat. # C100-0007) were measured on Ciphergen NP20 chips. There are 7 polypeptides in the sample with m/z values of 7034, 12230, 16951, 29023, 46671, 66433, 147300.

Format

A numeric vector represents the mass spectrum with equal sample intervals.

Source

CAMDA, CAMDA 2006 Competition Data Set. 2006, http://camda.duke.edu.

extendLength *Extend the length of a signal or matrix*

Description

Extend the length of a signal or matrix by row

Usage

```
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

```
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

```
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

```
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. @export @examples x <- c(1, 2, 3, 2, 1) findLocalMaxWinSize(x)

`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.

Usage

```
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

```
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

```
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.

Author(s)

Pan Du, Simon Lin

References

Du, P., Kibbe, W.A. and Lin, S.M. (2006) Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching, Bioinformatics, 22, 2059-2065.

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

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). The criteria for peak identification is described in the Details section.

Usage

```
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 as obtained by cwt().
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, used to estimate the signal of the SNR. See Details. If a single value is given then all scales larger than the value will be considered. If two values are given only the scales between those values will be considered.
ridgeLength	the maximum ridge scale of the major peaks.
nearbyPeak	determine whether to include the small peaks close to large major peaks. See Details.
nearbyWinSize	the window size to determine the nearby peaks. Only effective when nearbyPeak=TRUE.
winSize.noise	the local window size to estimate the noise level.
SNR.method	method to estimate the noise level. See Details.
minNoiseLevel	the minimum noise level used in calculating SNR. This value should be zero or positive. If the number is smaller than one, it is assumed to be a fraction of the largest wavelet coefficient in the data. Otherwise it is assumed to be the actual noise level. If you want to fix the actual noise level to a value smaller than one, you should name the value as fixed as in minNoiseLevel = c("fixed"= 0.5). See details.
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 ridge list may return peaks than have to be filtered out. This function filters the peaks according to the following rules. All rules must pass for a peak to be identified as such.

- The maximum scale of the peak ridge should be larger than `ridgeLength`. If `nearbyPeak=TRUE`, all peaks at less than `nearbyWinSize` points from a peak that fullfills this rule are also considered.
- The SNR of the peak must be larger than `SNR.Th`.
- The peak should not appear at the first or last `excludeBoundariesSize` points.

To debug and diagnose why good peaks get filtered, you may want to set `ridgeLength=0`, `SNR.Th=0` and/or `excludeBoundariesSize=0` to disable each of the filtering criteria.

SNR estimation:

The SNR is defined as $SNR = \frac{signal}{noise}$. Both signal and noise values need to be estimated for each peak.

The "signal" is estimated as the maximum wavelet coefficient obtained in the corresponding peak ridge, considering all the scales within `peakScaleRange`.

The "noise" is estimated differently depending on the `SNR.method`. All methods use a window of data points of size `2 * winSize.noise + 1` centered at the peak to make the noise estimation. Here is how the noise is estimated depending on the `SNR.method` value:

- `"quantile"`: The "noise" is the 95% quantile of the absolute value of the wavelet coefficients at scale 1 in the window.
- `"sd"`: The "noise" is the standard deviation of the absolute value of the wavelet coefficients at scale 1 in the window.
- `"mad"`: The "noise" is the `mad()` with `center=0` of the absolute value of the wavelet coefficients at scale 1 in the window.
- `"data.mean"`: The "noise" is the mean value of the ms spectrum in the window.
- `"data.mean.quant"`: The "noise" is the mean value of the ms spectrum in the window, but only considering values below the 95% quantile in the window.

If the obtained noise estimation is below the minimum noise level, that minimum is used as the noise estimation instead. Check `minNoiseLevel` for further details on how the minimum noise level is defined.

Using the estimated "signal" and "noise", we compute the `peakSNR` value for each peak.

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.

peakValue the CWT coefficients (the maximum on the ridge) corresponding to peakCenterIndex

peakSNR the SNR of the peak, which is the ratio of peakValue 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` `peakScale` and `peakRidgeLengthScale`.

peakRidgeLengthScale The largest scale value found for each ridge.

peakNoise The estimated noise on each peak, used to compute the SNR.

selInd Three logical vectors, one for each rule, determining which peak fullfills which rules.

`peakRidgeLengthScale`, `peakNoise` and `selInd` are meant for debugging and there is no guarantee they will appear in future versions. Please open an issue if you depend on them for any calculation if you find them useful.

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

Du, P., Kibbe, W.A. and Lin, S.M. (2006) Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching, Bioinformatics, 22, 2059-2065.

See Also

`peakDetectionCWT()`, `tuneInPeakInfo()`

Examples

```
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` else where.

Author(s)

Pan Du and Sergio Oller

See Also

`getLocalMaximumCWT()`

Examples

```
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

```
mexh(x)
```

Arguments

x	where to evaluate the mexican hat

Value

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

Examples

```
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

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

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()

---

mzV2indRange *Match m/z value to m/z index with a certain error range*

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

```
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

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

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

Du, P., Kibbe, W.A. and Lin, S.M. (2006) Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching, Bioinformatics, 22, 2059-2065.

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
```

```
peakIndex <- 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

```
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)
```

---

plotPeak *Plot the identified peaks over the spectrum*

Description

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

Usage

```
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.

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))
```

---

plotRidgeList *Plot the ridge list*

Description

Plot the ridge list returned by getRidge()

Usage

```
plotRidgeList(
  ridgeList,
  wCoefs = NULL,
  range = NULL,
  colorMap = "RYB",
  main = NULL,
  pch = ".",
  cex = 3,
  ...
)
```

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 *Prepare daughter wavelets for faster CWT*

Description

Prepare daughter wavelets for faster CWT

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

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

smoothDWT *smooth (denoise) the spectrum by DWT (Discrete Wavelet Transform)*

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

tuneInPeakInfo *Tune in the peak information: peak position and peak scale*

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

References

Du, P., Kibbe, W.A. and Lin, S.M. (2006) Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching, Bioinformatics, 22, 2059-2065.

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, 21
plotPeak, 22
plotRidgeList, 23
∗ **internal**
extendLength, 6
extendNBase, 7
∗ **methods**
cwt, 3
getLocalMaximumCWT, 8
getRidge, 10
getRidgeLength, 11
getRidgeValue, 12
identifyMajorPeaks, 12
localMaximum, 16
mzInd2vRange, 17
mzV2indRange, 18
peakDetectionCWT, 19
smoothDWT, 26
tuneInPeakInfo, 27
∗ **package**
MassSpecWavelet-package, 2

cwt, 3
cwt(), *13*, *19, 20*

exampleMS, 5
extendLength, 6
extendLength(), *7*
extendNBase, 7
extendNBase(), *6*

findLocalMaxWinSize, 8

getLocalMaximumCWT, 8
getLocalMaximumCWT(), *10, 11*, *16*, *19–22*, *27*
getRidge, 10
getRidge(), *12, 13*, *15*, *19, 20*, *23, 24*, *27*
getRidgeLength, 11
getRidgeValue, 12

identifyMajorPeaks, 12
identifyMajorPeaks(), *11*, *19, 20*, *22, 23*, *27*

localMaximum, 16
localMaximum(), *9*

mad(), *14*
MassSpecWavelet *(*MassSpecWavelet-package*)*, 2
MassSpecWavelet-package, 2
mexh, 17
mzInd2vRange, 17
mzInd2vRange(), *18*
mzV2indRange, 18
mzV2indRange(), *18*

peakDetectionCWT, 19
peakDetectionCWT(), *15*, *22, 23*, *27, 28*
plot(), *21*, *23, 24*
plotLocalMax, 21
plotPeak, 22
plotRidgeList, 23
points(), *21*, *23, 24*
prepareWavelets, 24
prepareWavelets(), *3*, *19*, *25*

signal::sgolayfilt(), *19, 20*
smoothDWT, 26

tuneInPeakInfo, 27
tuneInPeakInfo(), *15*