Package 'speaq2'

November 15, 2016

Version 0.1.0

Date 2016-11-10

Title Wavelet Based Tools for Feature-Wise Analysis and Quantification of Nuclear Magnetic Resonance (NMR) Spectra

Author Charlie Beirnaert, Kris Laukens

Maintainer Charlie Beirnaert <charlie.beirnaert@uantwerpen.be>

Description
Tools for feature based analysis of Nuclear Magnetic Resonance (NMR) spectra. Wavelets are used to quantify the peaks which are grouped into features with hierarchical clustering. The results can easily be incorporated in large-scale studies that include LC-MS analysis (as this is also peak/feature based).

Depends R (>= 2.15),

Imports MassSpecWavelet, mQTL, parallel, doSNOW, data.table, foreach, stats, cluster, utils, graphics, grDevices

Suggests datasets, speaq, knitr, rmarkdown, grid, gridExtra, gridBase, ggplot2

LazyData true

VignetteBuilder knitr

License GPL-3

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-15 08:45:34

R topics documented:

`AddPlottingStuff` *Add plotting variables*

Description

This functions adds a few variables which make plotting features easier (and more informative). Since for example every peaks keeps it original ppm value, if you want to plot the groups this function adds the group ppm value. Also sample labels can be added.

Usage

```
AddPlottingStuff(Y.peaks, X.ppm = NULL, groupLabels = NULL)
```

Arguments

`Y.peaks`	data frame obtained from either of the 'getWaveletPeaks', 'PeakAligner' or 'PeakFilling' function.
`X.ppm`	The vector with the ppm values (numeric vector).
`groupLabels`	The groupLabels (numeric or factor).

Value

Returns a data frame with added plotting variables (groupPPM for aligned features and labels for plotting).

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# This function works on a data frame resulting from the 'getWaveletPeaks' function
# DetectedPeaks <- getWaveletPeaks(X.ppm= PPM.vector, Y=Y.spec,  baselineThresh = 10,nCPU  = 4)
Aligned.peaks = PeakAligner = function (Y.peaks = DetectedPeaks)

## End(Not run)
```

BuildFeatureMatrix *Build a Feature matrix from the with speaq 2.0 processed data*

Description

This function converts the aligned peak data (so at least pake detection and alignment/grouping has to be completed) to a matrix with features (aligned peaks) in the columns and the value of that peak for every sample in the rows.

Usage

```
BuildFeatureMatrix(Y.data, var = "peakValue", impute = "zero",
  delete.below.threshold = FALSE, baselineThresh = 500, snrThres = 3,
  thresholds.pass = "any-to-pass")
```

Arguments

Y.data	The dataset after (at least) peak detection and alignment with speaq 2.0.
var	The variable to be used in the Featurematrix. This can be any of 'peakIndex', 'peakPPM', 'peakValue' (default), 'peakSNR', 'peakScale', or 'Sample'.
impute	What to impute when a certain peak is missing for a certain sample and feature combo. Options are 'zero' (or 'zeros'), any other statement will produce NA's.
delete.below.threshold	Whether to ignore peaks for which the 'var' variable has a value below 'baselineThresh' (default = FALSE).
baselineThresh	The threshold for the 'var' variable peaks have to surpass to be included in the feature matrix.
snrThres	The threshold for the signal-to-noise ratio of a peak.
thresholds.pass	This variable lets users deside whether a peak has to pass all the thresholds (both snrThres and baselineThresh), or just one. (If the peak does not need to surpass any thresholds set 'delete.below.threshold' to FALSE).

Value

a matrix, data.matrix, with samples for rows and features for columns. The values in the matrix are thoes of the 'var' variable.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# 'DetectedPeaks_aligned' is the peak data after wavelet based peak detection and alignment
Featurematrix <- BuildFeatureMatrix(Y.data = DetectedPeaks_aligned, var = 'peakValue')

## End(Not run)
```

`BuildRawDataMatrix` *Build a raw data matrix (spectra) from spectra of unequal length*

Description

This function can be used to build a data matrix from ill aligned spectra or of spectra of unequal length. the result is a matrix whereby the first column matches (approximately) with a sinle left ppm value and the last column matches (approxiamtely) with a single right ppm value. Crucial is that the sample rates of the machine are the same this should be always the case otherwise comparing intensities becomes meaningless. Note that, as standard in NMR spectra, the highest ppm value is on the left

Usage

```
BuildRawDataMatrix(spectrum.list, ppm.list = NULL, ppm.edges.matrix = NULL)
```

Arguments

`spectrum.list`	A list of the spectra (y-values). Since by definition some of these differ in length this has to be in list form with a single spectrum per list item.
`ppm.list`	The list of corresponding ppm values (x-values) with the highest ppm-value at the begining (left) as is the convention for NMR spectra. (This our ppm.edges.matrix has to be provided).
`ppm.edges.matrix`	The list with the starting and ending ppm values (highest ppm-value on the left/in the beginning). This or ppm.list has to be provided.

Value

SpectraAndPPM A list with 2 elements, the DataMatrix and the ppmMatrix.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
# this is an example for 3 meaningless spectra
lengths_of_spectra <- c(100,150,120)
measurement_distance <- 0.01
starting_ppm_values <- c(8.7, 9.0, 9.0)
spectra <- list()
ppm_values <- list()
for (k in 1:3) {
    spectra[[k]] <- runif(lengths_of_spectra[k], min = 0, max = 10)

    # note the minus sign in the 'by' statement
    ppm_values[[k]] <- seq(from = starting_ppm_values[k], by = -measurement_distance,
                           length.out = lengths_of_spectra[k])
}
new.Data <- BuildRawDataMatrix(spectrum.list = spectra, ppm.list = ppm_values)
spectraMatrix <- new.Data$DataMatrix
ppmMatrix <- new.Data$ppmMatrix
```

---

drawSpecPPM *Plot NMR spectra from a spectra data matrix*

Description

This function plots NMR spectra (so with the largest ppm values on the left) with a number of plotting options

Usage

```
drawSpecPPM(Y.spec, X.ppm, LeftIndex = -1, RightIndex = -1,
  groupFactor = NULL, useLog = FALSE, maxHeight = -1, minHeight = -1,
  nAxisPos = 4, xlab = NULL, ylab = NULL, title = NULL, ticks = NULL,
  ROI = NULL, ROI.ppm = NULL, roiWidth = 100, roiWidth.ppm = NULL,
  legend.extra.x = 2, legend.extra.y = 2, legendpos = NULL,
  colourstyle = "ggplot", manual.colours = NULL, lwd = 1,
  noLegend = FALSE)
```

Arguments

Y.spec	(required) The raw spectra in matrix format (1 sample per row) or numeric vector (in case of 1 spectrum)
X.ppm	(required) The vector with the ppm values
LeftIndex	The starting index of the ppm values for plotting. default = -1 indicates the first ppm (the largest) value is the start of the plot
RightIndex	The stopping index for plotting. default = -1 indicates the last ppm value (the smallest) is the end of the plot
groupFactor	The groupFactors. If provided different colors will be used for each group.

useLog	If set to 'TRUE' the spectra will be log10 transformed (default = FALSE).
maxHeight	The maximal height of the plot (default = -1, this indicates no maximal value).
minHeight	The minimal height of the plot (default = -1, this indicates no minimal value).
nAxisPos	The number of equally spaced tickmarks.
xlab	The label on the x axis.
ylab	The label on the y axis.
title	The title of the plot.
ticks	Position tick manually by providing ppm values.
ROI	If provided (with an index value, not a ppm value) only this region of interest will be plotted. (supply no ROI or ROI.ppm values, for the full spectrum, or specify only 1, either ROI or ROI.ppm).
ROI.ppm	If provided (a ppm value, not an index value) only this region of interest will be plotted. (supply no ROI or ROI.ppm values, for the full spectrum, or specify only 1, either ROI or ROI.ppm).
roiWidth	The width of the ROI (region of interest) plot in index points/measurement points. The plot will span from ROI/ROI.ppm - roiWidth to ROI/ROI.ppm + roiWidth. (only supply roiWidth or roiWidth.ppm if needed).
roiWidth.ppm	The width of the ROI (region of interest) plot in ppm. The plot will span from ROI/ROI.ppm - roiWidth.ppm to ROI/ROI.ppm + roiWidth.ppm. (only supply roiWidth or roiWidth.ppm if needed).
legend.extra.x	Increase (or decrease) the horizontal speace in the legend, this is useful for exporting larger figures.
legend.extra.y	Increase (or decrease) the vertical speace in the legend, this is useful for exporting larger figures.
legendpos	The position of the legend (standard R legend positioning, default = 'topleft').
colourstyle	The colours used in the plot, either standard R or ggplot colours (default).
manual.colours	Provide specific colours to be used in the plot.
lwd	The linewidth.
noLegend	If set to TRUE no legend will be plotted (default = FALSE).

Value

an R plot

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
data(Winedata)
Spectra = Winedata$spectra
ppm.wine = Winedata$ppm
wine.color = Winedata$wine.color
```

```
drawSpecPPM(Y.spec=Spectra, X.ppm=ppm.wine, groupFactor = wine.color,
title = 'Raw wine data spectra')
```

`getWaveletPeaks` *Convert raw NMR spectra to peak data by using wavelets*

Description

This function converts phase corrected NMR spectra to peak data by using wavelet based peak detection (with the MassSpecWavelet package)

Usage

```
getWaveletPeaks(Y.spec, X.ppm, sample.labels = NULL, window.width = "small",
  window.split = 4, scales = seq(1, 16, 1), baselineThresh = 1000,
  SNR.Th = -1, nCPU = -1, include_nearbyPeaks = TRUE)
```

Arguments

`Y.spec`	The spectra in matrix format.
`X.ppm`	The x/ppm values of the spectra (in single vector or matrix format).
`sample.labels`	The sample labels (numeric), if not supplied these will simply be the sample numbers.
`window.width`	The width of the detection window for the wavelets. Because of the Fourier transform lengths of 512 (window.width = 'small') of 1024 (window.width = 'large') are preferable.
`window.split`	A positive, even and whole number indicating in how many parts the sliding window is split up. With every iteration the window slides one part further.
`scales`	The scales to be used in the wavelet based peak detection, see peakDetectionCWT.
`baselineThresh`	Peaks with a peakValue lower than this threshold will be removed (default = 1000).
`SNR.Th`	The Signal-to-noise threshold, see peakDetectionCWT.
`nCPU`	The amount of cpu's to be used for peak detection. If set to '-1' all available cores minus 1 will be used.
`include_nearbyPeaks`	If set to TRUE small peaks in the tails of larger ones will be included in the peak data, see peakDetectionCWT.

Value

The peaks detected with the wavelets.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# This example takes ca. 50 seconds (with nCPU = 4) on a 2.5GHz machine
data(Winedata)
Y.spec = as.matrix(Winedata$spectra)
PPM.vector = as.numeric(Winedata$ppm)
DetectedPeaks <- getWaveletPeaks(X.ppm= PPM.vector, Y=Y.spec,  baselineThresh = 10,nCPU  = 4)

## End(Not run)
```

---

hclust.grouping *Realignment with hierarchical clustering (used in the PeakAligner function)*

Description

Internal function in the PeakAligner function for generating the hierarchical clustering tree and cutting it.

Usage

```
hclust.grouping(current.peaks, min.samp.grp = 1, max.dupli.prop = 0.25,
  maxClust = 10, linkage = "average")
```

Arguments

current.peaks	A number of neighbouring peaks to be aligned.
min.samp.grp	The minimal amount of samples needed te form a group.
max.dupli.prop	The maximal duplication proportion allowed for a group to be considered a sigle group.
maxClust	The maximum number of clusters (depth of the tree).
linkage	The linkage to be used in the hierarchical clustering. See the 'method' argument in hclust.

Value

Returns a data frame with realigned peaks (aligned on groupIndex).

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

PeakAligner *Realignment with hierarchical clustering*

Description

This functions aligns the data frame obtained after wavelet based peak detection with the 'getWaveletPeaks' function

Usage

```
PeakAligner(Y.peaks, grouping.window.width = 100, verbose = FALSE,
  min.samp.grp = 1, max.dupli.prop = 0.25, maxClust = 10,
  Jaccard.regroup.threshold = 0.25, linkage = "average")
```

Arguments

`Y.peaks`	data frame obtained from the 'getWaveletPeaks' function.
`grouping.window.width`	The width of the sliding window (in measurement points). Measurments are taken for when this sliding window is taken too small, but best set this too a value that a normal peak is comfortably in a window. Note if large shifts occur in your dataset (like in the wine dataset) it is best to set this parameter larger.
`verbose`	If set to TRUE the window selection process is documented in real time (default = FALSE).
`min.samp.grp`	The minimal amount of samples needed te form a group, see hclust.grouping.
`max.dupli.prop`	The maximal duplication proportion allowed for a group to be considered a sigle group, see hclust.grouping.
`maxClust`	The maximum number of clusters (depth of the tree), see hclust.grouping.
`Jaccard.regroup.threshold`	If 2 neighbouring groups have a jaccard index smaller than this 'Jaccard.regroup.threshold' (indicating that they are quite complementary as they have little peaks samples in common), then they are merged and regrouped. This situation can occur if a group is accidentally cut in half by the window approach.
`linkage`	The linkage to be used in the hierarchical clustering. See the 'method' argument in hclust.

Value

Returns a data frame with realigned peaks (aligned on groupIndex).

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# This function works on a data frame resulting from the 'getWaveletPeaks' function
# DetectedPeaks <- getWaveletPeaks(X.ppm= PPM.vector, Y=Y.spec,  baselineThresh = 10,nCPU  = 4)
Aligned.peaks = PeakAligner = function (Y.peaks = DetectedPeaks)

## End(Not run)
```

PeakFilling *Peak filling of any missed peaks*

Description

This functions detects which samples (after alignment) are missing from every aligned peak group and reanalyses the raw data to verify whether this peak is actually non-existent for this sample

Usage

```
PeakFilling(Y.aligned, Y.spec, max.index.shift = 10, window.width = "small",
  nCPU = -1)
```

Arguments

Y.aligned	Aligned peaks (after the 'PeakAligner' function).
Y.spec	The raw NMR spectra in matrix format.
max.index.shift	Maximal shift in index between a filled peak and the group it belongs to.
window.width	The width of the detection window for the wavelets. Because of the Fourier transform lengths of 512 (window.width = 'small') of 1024 (window.width = 'large') are preferable.
nCPU	The amount of cpu's to be used for peak detection. If set to '-1' all available cores minus 1 will be used.

Value

Returns a data frame with aligned peaks and possibly extra peaks obtained from the raw data (these peaks have SNR = NA).

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# This function works on a data frame resulting from the 'PeakAligner' function
# DetectedPeaks <- getWaveletPeaks(X.ppm= PPM.vector, Y=Y.spec,  baselineThresh = 10,nCPU  = 4)
# Aligned.peaks = PeakAligner = function (Y.peaks = DetectedPeaks)
Filled.Peaks = SpecPeak.filling(Y.aligned = Aligned.peaks, Y.spec = Y)

## End(Not run)
```

`regroupR` *Regroup faulty aligned peaks*

Description

If there are groups misaligned by the peakAligner function, they will be realigned by also using the peak signal to noise ratio as well as the ppm values.

Usage

```
regroupR(aligned.peaks, list.to.regroup, min.samp.grp = 1,
  max.dupli.prop = 0.1, maxClust = 10)
```

Arguments

`aligned.peaks`	The aligned peaks data.
`list.to.regroup`	The peak indices of groups to regroup (the groups, indicated by their peakIndex, in 1 list item will be merged and regrouped).
`min.samp.grp`	The minimal amount of samples needed te form a group.
`max.dupli.prop`	The maximal duplication proportion allowed for a group to be considered a sigle group.
`maxClust`	The maximum number of clusters (depth of the tree).

Value

Returns a data frame with regrouped peaks.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

`relevant.features.p` *identify features (columns in the datamatrix) which are significantly associated with the outcome vector*

Description

This function produces a p-value for every column in the datamatrix, corresponding to the null hypothesis that outcome/responsevector is independent of that feature.

Usage

```
relevant.features.p(datamatrix, responsevector, p.adj = "bonferroni")
```

Arguments

`datamatrix`	the data matrix with a column for each feature.
`responsevector`	the vector of outcomes/responses (e.g. class labels). the length of this vector should match the amount of rows in datamatrix.
`p.adj`	the adjustment method for the p-values. Any of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr' or 'none' are accepted.

Value

data with the features and their (adjusted) p-values, one for every column in the datamatrix .

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
nSamples <- 10
nFeatures <- 20
data.matrix <- matrix( stats::runif(n=nFeatures*nSamples, min=0,max=100),
ncol = nFeatures, nrow = nSamples)

response <- c( rep(0,nSamples/2), rep(1,nSamples/2) )
p_values <- relevant.features.p(datamatrix = data.matrix, responsevector =
response, p.adj = 'none')
p_values_adjusted <- relevant.features.p( datamatrix = data.matrix,
responsevector = response, p.adj = 'bonferroni')
```

SCANT *SCAle, Normalize and Transform a data matrix*

Description

This function allows the column-wise or row-wise scaling, normalization and tranformation operations on a data matrix.

Usage

```
SCANT(data.matrix, type = "unit", what = "columns")
```

Arguments

data.matrix	the data matrix to be scaled, normalized or transformed.
type	the operations to be performed, this can be multiple and are performed sequantially. Any of 'unit', 'pareto', 'log10', 'log2', 'center', 'range', 'vast' or 'prob.Q' are accepted.
what	to specify on which to perform the operations (row or column).

Value

The scaled, normalized and/or transformed matrix.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
Samples <- 10
Features <- 20
data.matrix <- matrix(runif(n=Features*Samples, min=0,max=100),
ncol = Features, nrow = Samples)

changed_matrix = SCANT(data.matrix, type=c('pareto', 'center'), what = 'columns')
```

SilhouetR *SilhouetR*

Description

Calculate the Silhouette value for

Usage

```
SilhouetR(DataMatrix, GroupIndices, distance = "euclid", stand = TRUE)
```

Arguments

DataMatrix	a matrix with the raw data, 1 variable per column.
GroupIndices	The vector with the roup indices (length must be equal to the amount of rwos in DataMatrix).
distance	The distance metric to be used, see daisy.
stand	whether to standardize the data before calculating the dissimilarities. See daisy.

Value

Returns the silhouette values.

Author(s)

Charlie Beirnaert, <charlie.beirnaert@uantwerpen.be>

Examples

```
## Not run:
# This function works on aligned peak data
library(ggplot2)
Silhouette.values = SilhouetR(DataMatrix = Aligned.peaks$peakPPM,
Aligned.peaks$peakIndex, distance = 'euclid', stand = TRUE)
ggplot(SilhouetteValues, aes(SilhouetteValues)) +geom_freqpoly(binwidth = 0.03) +theme_bw()

## End(Not run)
```

`Winedata` *Wine dataset*

Description

1H-NMR data of 40 wines, different origins and colors are included.

Usage

```
data(Winedata)
```

Format

A list with the spectra, ppm values, color and origin as list entries.

Source

University of Copenhagen, Dept. of Food Science, Quality & Technology

References

Larsen et al. (2006) An exploratory chemometric study of 1H-NMR spectra of table wines. J.Chemom. 20 (2006) 198-208 (Wiley Online Library)

Examples

```
data(Winedata)
Spectra <- Winedata$spectra
ppm.wine <- Winedata$ppm
wine.color <- Winedata$wine.color
wine.origin <- Winedata$origin
```

Index