Package 'speaq'

July 23, 2025

Version 2.7.0

Date 2022-05-20

Title Tools for Nuclear Magnetic Resonance (NMR) Spectra Alignment, Peak Based Processing, Quantitative Analysis and Visualizations

Author Charlie Beirnaert, Trung Nghia Vu, Pieter Meysman, Kris Laukens and Dirk Valkenborg

Maintainer Charlie Beirnaert <charlie_beirnaert@icloud.com>

Description Makes Nuclear Magnetic Resonance spectroscopy (NMR spectroscopy) data analysis as easy as possible by only requiring a small set of functions to perform an entire analysis. 'speaq' offers the possibility of raw spectra alignment and quantitation but also an analysis based on features whereby the spectra are converted to peaks which are then grouped and turned into features. These features can be processed with any number of statistical tools either included in 'speaq' or available elsewhere on CRAN. More details can be found in Vu et al. (2011) <doi:10.1186/1471-2105-12-405> and Beirnaert et al. (2018) <doi:10.1371/journal.pcbi.1006018>.

Depends R (>= 3.1.0),

Imports MassSpecWavelet, cluster, parallel, doSNOW, data.table, foreach, stats, Rfast, utils, graphics, grDevices, ggplot2, gridExtra, reshape2, rvest, xml2, missForest, impute

Suggests datasets, knitr, rmarkdown, grid, gridBase

LazyData true

VignetteBuilder knitr

License Apache License 2.0

RoxygenNote 7.2.0

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-23 09:10:02 UTC

Contents

`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

```
subset <- GetWinedata.subset()
subset.spectra = as.matrix(subset$Spectra)
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm,
                              nCPU = 1) # nCPU set to 2 for the vignette build

test.peaks.plot = AddPlottingStuff(test.peaks, subset.ppm, subset$Color)
#head(test.peaks.plot)
```

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

Description

This function converts the grouped peak data to a matrix. The matrix has features (peaks groups) in the columns and the value of the peak for every sample in the rows.

Usage

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

Arguments

Y.data	The dataset after (at least) peak detection and grouping with speaq 2.0. The dataset after peak filling is recommended.
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", the default), "median" (imputation with feature median), "randomForest" (imputation with missForest function from package missForest) or kNN followed by a number indicating the amount of neighbours to use e.g. "kNN5" or "kNN10" (as per the method of Troyanskaya, 2001) or lasty "User_value" (this will allow the use of any value specified with the imputation_val argument e.g. the median of the raw spectra). Any other statement will produce NA's.
imputation_val	If the "User_value" imputation option is chosen this value will be used to impute the missing values.
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 that 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 decide 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 those of the 'var' variable.

Author(s)

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

References

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman, Missing value estimation methods for DNA microarrays BIOINFORMATICS Vol. 17 no. 6, 2001 Pages 520-525

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)
```

```
test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm,
                              nCPU = 1) # nCPU set to 2 for the vignette build

test.grouped <- PeakGrouper(Y.peaks = test.peaks)

test.Features <- BuildFeatureMatrix(test.grouped)
```

`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 single left ppm value and the last column matches (approximately) 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 beginning (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
```

BWR *BW ratio calculation*

Description

Compute the BW ratios from data groups

Usage

```
BWR(X, groupLabel)
```

Arguments

X	The spectral dataset in the matrix format in which each row contains a single sample.
groupLabel	Group label of samples in the dataset.

Value

Return BW ratio

Author(s)

Trung Nghia Vu

See Also

createNullSampling

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
maxShift = 50;
Y <- dohCluster(X,
                peakList = peakList,
                refInd = refInd,
                maxShift  = maxShift,
                acceptLostPeak = TRUE, verbose=FALSE);
# find the BW-statistic
BW = BWR(Y, groupLabel);
```

`createNullSampling` *Building a null hypothesis data*

Description

Create a null sampling data (N times) and write them to a file

Usage

```
createNullSampling(X, groupLabel, N = 100, verbose = TRUE)
```

Arguments

`X`	The spectral dataset in the matrix format in which each row contains a single sample
`groupLabel`	Group label of samples in the dataset
`N`	The number of iteration for creating null sample distribution
`verbose`	A boolean value to allow print out process information

Value

A matrix with N rows containing the null distribution.

Author(s)

Trung Nghia Vu

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
maxShift = 50;
Y <- dohCluster(X,
                peakList = peakList,
                refInd = refInd,
                maxShift  = maxShift,
                acceptLostPeak = TRUE, verbose=FALSE);
# find the BW-statistic
BW = BWR(Y, groupLabel);
H0 = createNullSampling(Y, groupLabel, N = 100,verbose=FALSE)
```

`detectSpecPeaks` *Peak detection for spectra*

Description

Divide the whole spectra into smaller segments and detect peaks by using MassSpecWavelet package. Note that, the peak lists could be found by using other methods, this function is just a choice.

Usage

```
detectSpecPeaks(
  X,
  nDivRange = 128,
  scales = seq(1, 16, 2),
  baselineThresh = 50000,
  SNR.Th = -1,
  verbose = TRUE
)
```

Arguments

`X`	The spectral dataset in matrix format in which each row contains a single sample
`nDivRange`	The size of a single small segment after division of spectra

scales	The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function.
baselineThresh	It will remove all peaks under an intensity set by baselineThresh.
SNR.Th	The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function. If you set -1, the function will itself re-compute this value.
verbose	A boolean value to allow print out process information.

Value

The peak lists of the spectra

Author(s)

Trung Nghia Vu

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
```

---

dohCluster *CluPA function for multiple spectra.*

Description

Use CluPA for alignment for multiple spectra.

Usage

```
dohCluster(
  X,
  peakList,
  refInd = 0,
  maxShift = 100,
  acceptLostPeak = TRUE,
  verbose = TRUE
)
```

Arguments

X	The spectral dataset in the matrix format in which each row contains a single sample
peakList	The peak lists of the spectra
refInd	The index of the reference spectrum.
maxShift	The maximum number of the points for a shift step.
acceptLostPeak	This is an option for users, TRUE is the default value. If the users believe that all the peaks in the peak list are true positive, change it to FALSE.
verbose	A boolean value to allow print out process information.

Value

The aligned spectra.

Author(s)

Trung Nghia Vu

See Also

dohClusterCustommedSegments

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
maxShift = 50;
Y <- dohCluster(X,
                peakList = peakList,
                refInd = refInd,
                maxShift  = maxShift,
                acceptLostPeak = TRUE, verbose=FALSE);
```

dohClusterCustommedSegments *Use CluPA for alignment with additional information*

Description

This function integrates some additional information from user such as references for each specific segment, segment ignorance, maximum step size.. to align spectra using CluPA.

Usage

```
dohClusterCustommedSegments(
  X,
  peakList,
  refInd,
  segmentInfoMat,
  minSegSize = 128,
  maxShift = 100,
  acceptLostPeak = TRUE,
  verbose = TRUE
)
```

Arguments

X	The spectral dataset in the matrix format in which each row contains a single sample
peakList	The peak lists of the spectra
refInd	The index of the reference spectrum.
segmentInfoMat	The matrix containing the additional information for segments from the users. This parameter must be a matrix.
minSegSize	The minimum size of the segments which could be considered for alignment.
maxShift	The maximum number of the points for a shift step.
acceptLostPeak	This is an option for users, TRUE is the default value. If the users believe that all the peaks in the peak list are true positive, change it to FALSE.
verbose	A boolean value to allow print out process information.

Details

Each row of the segmentInfoMat matrix includes 5 values. For example, it could be imported from a CSV file consisting of following content: # begin,end,forAlign,ref,maxShift 100,200,0,0,0 450,680,1,0,50 # Each column could be explained as the following: * begin: the starting point of the segment. * end: the end point of the segment. * forAlign: the segment is aligned (1) or not (0). * ref: the index of the reference spectrum. If 0, the algorithm will select the reference found by the reference finding step. * maxShift: the maximum number of points of a shift to left/right. It is worth to note that only segments with forAlign=1 (column 3) will be taken into account for spectral alignment.

Value

The aligned spectral segments.

Author(s)

Trung Nghia Vu

See Also

dohCluster

Examples

```
cat("\n Please see more examples in the vignettes file.")
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
segmentInfoMat=matrix(data=c(100,200,0,0,0,
                             50,680,1,0,50),nrow=2,ncol=5,byrow=TRUE
)
colnames(segmentInfoMat)=c("begin","end","forAlign","ref","maxShift")
segmentInfoMat
maxShift = 50;
Yc <- dohClusterCustommedSegments(X,
                                  peakList = peakList,
                                  refInd = refInd,
                                  maxShift  = maxShift,
                                  acceptLostPeak = TRUE,
                                  segmentInfoMat = segmentInfoMat,
                                  minSegSize = 128,
                                  verbose=FALSE)
```

---

doShift *Segment shift*

Description

Move a spectral segment of a sample shiftStep points to right or left

Usage

```
doShift(specSeg, shiftStep)
```

Arguments

specSeg	The segment which needs to be shifted
shiftStep	The shift step for moving. If it is a negative (positive) value, the segment is moved to left (right).

Value

The new segment after shifting.

Author(s)

Trung Nghia Vu

See Also

hClustAlign, findShiftStepFFT

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
maxShift=50;
refSpec=X[1,];
tarSpec=X[2,];
adj=findShiftStepFFT(refSpec, tarSpec,maxShift=maxShift);
newTarSpec=doShift(tarSpec,adj$stepAdj);
```

---

drawBW *BW and percentile ratios plot*

Description

This function is used to plot BW and percentile ratios

Usage

```
drawBW(
  BW,
  perc,
  X,
  startP = -1,
  endP = -1,
```

```
  groupLabel = NULL,
  highBound = -1,
  lowBound = -1,
  nAxisPos = 4,
  offside = 0
)
```

Arguments

BW	An array of the BW ratios.
perc	An array of the percentile ratios.
X	The spectral dataset in matrix format in which each row contains a single sample.
startP	The starting point of the segment. If it is -1, the starting point is from beginning of the spectra.
endP	The ending point of the segment. If it is -1, the ending point is the last point of the spectra.
groupLabel	The default value is NULL, it means that a single spectrum has a distinct color. Otherwise, the spectra is colored by their label.
highBound	Default value is -1, that means the plot covers also the highest intensity peaks in the figure. If the users want to limit the upper height of the figure, set this parameter by the limited value.
lowBound	Default value is -1, that means the plot covers also the lowest intensity peaks in the figure. If the users want to limit the under height of the figure, set this parameter by the limited value.
nAxisPos	The number of ticks that will be displayed in the horizontal axis.
offside	The offside of values in x-axis for display.

Value

Return a plot containing both the BW and the spectra.

Author(s)

Trung Nghia Vu

See Also

drawSpec

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
```

```
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
maxShift = 50;
Y <- dohCluster(X,
                peakList = peakList,
                refInd = refInd,
                maxShift  = maxShift,
                acceptLostPeak = TRUE, verbose=FALSE);
# find the BW-statistic
BW = BWR(Y, groupLabel);
N = 100;
alpha = 0.05;
# create sampled H0 and export to file
H0 = createNullSampling(Y, groupLabel, N = N,verbose=FALSE)
#compute percentile of alpha
perc = double(ncol(Y));
alpha_corr = alpha/sum(returnLocalMaxima(Y[2,])$pkMax>50000);
for (i in seq_along(perc)) {
    perc[i] = quantile(H0[,i],1-alpha_corr, type = 3);
}
drawBW(BW, perc,Y, groupLabel = groupLabel)
```

---

drawSpec *Spectral plot*

Description

This function allows to draw a segment or the whole spectra with limited high/low bounds of intensity.

Usage

```
drawSpec(
  X,
  startP = -1,
  endP = -1,
  groupLabel = NULL,
  useLog = -1,
  highBound = -1,
  lowBound = -1,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
```

```
  nAxisPos = 4,
  offside = 0
)
```

Arguments

X	The spectral dataset in matrix format in which each row contains a single sample.
startP	The starting point of the segment. If it is -1, the starting point is from beginning of the spectra.
endP	The ending point of the segment. If it is -1, the ending point is the last point of the spectra.
groupLabel	The default value is NULL, it means that a single spectrum has a distinct color. Otherwise, the spectra is colored by their label.
useLog	The default value is -1, that means do not use a log transformation. If users want to transform the intensities to logarit values before plotting, set it to 1.
highBound	Default value is -1, that means the plot covers also the highest intensity peaks in the figure. If the users want to limit the upper height of the figure, set this parameter by the limited value.
lowBound	Default value is -1, that means the plot covers also the lowest intensity peaks in the figure. If the users want to limit the under height of the figure, set this parameter by the limited value.
xlab	The default value is NULL, if so, "index" is displayed at the horizontal axis.
ylab	The default value is NULL, if so, "intensity" is displayed at the vertical axis.
main	The default value is NULL, if so, the title shows the values of startP and endP.
nAxisPos	The number of ticks that you want to display in horizontal axis.
offside	The offside of values in x-axis for display.

Value

Return a plot of the spectra.

Author(s)

Trung Nghia Vu

See Also

drawBW

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
drawSpec(X)
```

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 space in the legend, this is useful for exporting larger figures.
legend.extra.y	Increase (or decrease) the vertical space 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')
```

---

findRef *Reference finding*

Description

This function is to heuristically detect a reference spectrum.

Usage

```
findRef(peakList)
```

Arguments

peakList	The peak lists of the spectra.

Value

list of 2: refInd (The index of the reference spectrum found by the algorithm) and orderSpec (A sorted array of the spectra by their goodness values)

Author(s)

Trung Nghia Vu

References

Vu TN, Valkenborg D, Smets K, Verwaest KA, Dommisse R, Lemi\‘ere F, Verschoren A, Goethals B, Laukens K. (2011) An integrated workflow for robust alignment and simplified quantitative analysis of NMR spectrometry data. BMC Bioinformatics. 2011 Oct 20;12:405.

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
```

```
                            SNR.Th = -1,
                            verbose=FALSE
);
cat("\n Find the spectrum reference...")
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
cat("\n Order of spectrum for reference  \n");
for (i in seq_along(resFindRef$orderSpec))
    cat(paste(i, ":",resFindRef$orderSpec[i],sep=""), " ");
cat("\n The reference is: ", refInd);
```

`findSegPeakList` *Selecting the peaks in a segment*

Description

This function is to find out which peaks belonging to a segment which ranges from startP to endP

Usage

```
findSegPeakList(peakList, startP, endP)
```

Arguments

`peakList`	The peak lists of the spectra.
`startP`	The starting point of the segment.
`endP`	The ending point of the segment.

Value

The list of indices of the peaks in the segment.

Author(s)

Trung Nghia Vu

See Also

`dohClusterCustommedSegments`

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
                            );
cat("\n ", peakList[[1]])
segmentpeakList= findSegPeakList(peakList[[1]],400,600);
cat("\n ", segmentpeakList)
```

`findShiftStepFFT` *Finding the shift-step by using Fast Fourier Transform cross- correlation*

Description

This function uses Fast Fourier Transform cross-correlation to find out the shift step between two spectra.

Usage

```
findShiftStepFFT(refSpec, tarSpec, maxShift = 0, scale = NULL)
```

Arguments

`refSpec`	The reference spectrum.
`tarSpec`	The target spectrum which needs to be aligned.
`maxShift`	The maximum number of points for a shift step. If this value is zero, the algorithm will check on the whole length of the spectra.
`scale`	Boolean value (TRUE/FALSE) for scaling data before Fast Fourier Transform cross-correlation step. If scale=NULL but mean/median of absolute data is too small (<1), the scaling will be applied. This might happen for very low abundant spectra like chromatograms. For normal NMR spectra, the scaling is usually not applied.

Value

list of 2: corValue (The best correlation value) and stepAdj (The shift step found by the algorithm)

Author(s)

Trung Nghia Vu

See Also

hClustAlign

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
maxShift=50;
refSpec=X[1,];
tarSpec=X[2,];
adj=findShiftStepFFT(refSpec, tarSpec,maxShift=maxShift);
```

---

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,
  raw_peakheight = FALSE,
  duplicate_detection_multiplier = 1
)
```

Arguments

Y.spec	The spectra in matrix format (rows = samples, columns = measurement points).
X.ppm	The x/ppm values of the spectra (in single vector or matrix format).
sample.labels	The sample labels (optional), 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.

`raw_peakheight` (default = FALSE) Whether to use the raw peak height of a peak instead of the optimal CWT coefficient (which is a measure for AUC).

`duplicate_detection_multiplier` (default 1) In case users want to process other spectra besides NMR, this parameter will increase the limit for two peaks to be considered a duplicate detection. When dealing with more distorted spectra this parameter can be increased (recommended to not increase above 10).

Value

The peaks detected with the wavelets.

Author(s)

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

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm ,
                              nCPU = 1) # nCPU set to 2 for the vignette build
```

GetWinedata.subset *Get subset of Winedata for code examples*

Description

This functions extracts a small part of the Winedata to be used in code testing and code examples

Usage

```
GetWinedata.subset()
```

Value

list of 2: spectra, ppm values, color and origin.

Author(s)

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

Examples

```
subset <- GetWinedata.subset()
subset.spectra = subset$Spectra
subset.ppm = subset$PPM.vector
```

hclust.grouping *Grouping with hierarchical clustering (used in the PeakGrouper function)*

Description

Internal function in the PeakGrouper 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 grouped.
min.samp.grp	The minimal amount of samples needed to form a group.
max.dupli.prop	The maximal duplication proportion allowed for a group to be considered a single 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 grouped peaks.

Author(s)

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

See Also

PeakGrouper

---

hClustAlign *CluPA function for two spectra.*

Description

This function implements the idea of the CluPA algorithm to align the target spectrum against the reference spectrum.

Usage

```
hClustAlign(
  refSpec,
  tarSpec,
  peakList,
  peakLabel,
  startP,
  endP,
  distanceMethod = "average",
  maxShift = 0,
  acceptLostPeak = FALSE
)
```

Arguments

refSpec	The reference spectrum.
tarSpec	The target spectrum.
peakList	List of peaks of the both reference and target spectra
peakLabel	The list of the labels of the peaks
startP	The starting point of the segment.
endP	The ending point of the segment.
distanceMethod	The distance method for the hierarchical clustering algorithm.
maxShift	The maximum number of points for a shift step.
acceptLostPeak	This is an option for users, TRUE is the default value. If the users believe that all the peaks in the peak list are true positive, change it to FALSE.

Value

list of 2: tarSpec (The target spectrum after alignment) and peakList (The peak list after alignment)

Author(s)

Trung Nghia Vu

References

Vu TN, Valkenborg D, Smets K, Verwaest KA, Dommisse R, Lemi\‘ere F, Verschoren A, Goethals B, Laukens K. (2011) An integrated workflow for robust alignment and simplified quantitative analysis of NMR spectrometry data. BMC Bioinformatics. 2011 Oct 20;12:405.

See Also

dohCluster

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
peakList <- detectSpecPeaks(X,
                            nDivRange = c(128),
                            scales = seq(1, 16, 2),
                            baselineThresh = 50000,
                            SNR.Th = -1,
                            verbose=FALSE
);
resFindRef<- findRef(peakList);
refInd <- resFindRef$refInd;
tarInd=1;
refSpec=X[refInd,];
tarSpec=X[tarInd,];
mergedPeakList=c(peakList[[refInd]],peakList[[tarInd]]);
```

```
mergedPeakLabel=double(length(mergedPeakList));
for (i in seq_along(peakList[[refInd]]) ) mergedPeakLabel[i]=1;
startP=1;
endP=length(tarSpec);
res=hClustAlign(refSpec,tarSpec,mergedPeakList,mergedPeakLabel,startP,endP,
                maxShift=50,acceptLostPeak=TRUE)
```

HMDBsearchR *Submit 1H NMR peaks to HMDB for compound search*

Description

This function allows to search HMDB from within R by simply submitting the peaks you want to search for. The function will open a webpage with the query results or provide a link to the HMDB page with the results.

Usage

```
HMDBsearchR(peakVector, ppmTol = 0.02, returnURL = FALSE)
```

Arguments

peakVector	A vector with ppm values of peaks
ppmTol	The ppm tolerance for the HMDB search (default = 0.02).
returnURL	Return the URL instead of opening a webpage.

Value

Opens a webpage or returns a URL with the HMDB results

Author(s)

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

Examples

```
## Not run:
HMDBsearchR(peakVector = c(3.2, 3.38), ppmTol = 0.2, returnURL = TRUE)

## End(Not run)
```

makeSimulatedData *Create a simulated NMR spectral data*

Description

Generate an NMR spectral data for testing.

Usage

```
makeSimulatedData()
```

Details

We generate a NMR spectral data sets that contains two group A and group B. One at around 300 has a single tip and the other at around 600 has double tips that intentionally contains biological variation. First, a single spectrum is created based on statistic information (mean, standard deviation of intensity) achieved from real NMR spectra. Then, we randomly shift the spectrum to maximum 50 data points and add some biological and technical variations to each point intensity to the spectrum to create a new spectrum. The collection of spectra from each group is the final dataset.

Value

a list with 2 elements: data (The simulated NMR spectral data matrix) and label (Group label of each spectrum)

Author(s)

Trung Nghia Vu

Examples

```
res <- makeSimulatedData();
X <- res$data;
groupLabel <- res$label;
```

PeakFilling *Peak filling of any missed peaks*

Description

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

Usage

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

Arguments

Y.grouped	Peaks groups (output from the 'PeakGrouper' 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.
FilMethod	A more robust method for peak filling has been implemented. This is now the default. The former method can be used by specifying FilMethod == "old" however this will be deprecated.

Value

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

Author(s)

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

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm,
                              nCPU = 1) # nCPU set to 1 for the vignette build

test.grouped <- PeakGrouper(Y.peaks = test.peaks)

test.filled <- PeakFilling(Y.grouped = test.grouped,
```

```
                    Y.spec = subset.spectra,
                    nCPU = 1) # nCPU set to 1 for the vignette build
```

PeakGrouper *Peak grouping with hierarchical clustering*

Description

This functions groups the peaks obtained after wavelet based peak detection (with the 'getWavelet-Peaks' function).

Usage

```
PeakGrouper(
  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). Measurements 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 to form a group, see hclust.grouping.
`max.dupli.prop`	The maximal duplication proportion allowed for a group to be considered a single 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 grouped peaks. Peaks in a group are indicated with an identical peakIndex

Author(s)

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

See Also

hclust.grouping

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm ,
                              nCPU = 1) # nCPU set to 2 for the vignette build

test.grouped <- PeakGrouper(Y.peaks = test.peaks)
```

---

regroupR *Regroup faulty grouped peaks*

Description

If there are peaks wrongly grouped by the peakGrouper function, they will be regrouped by using the ppm values together with the peak signal to noise ratio.

Usage

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

Arguments

grouped.peaks	The grouped 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 to form a group.
max.dupli.prop	The maximal duplication proportion allowed for a group to be considered a single 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.*

Description

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

Usage

```
relevant.features.p(
  datamatrix,
  response,
  p.adj = "BH",
  POI = 1,
  responsevector = NULL
)
```

Arguments

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

POI	Only if 'response' is a matrix! The p values of interest. This is a number indicating which column of the 'response' matrix you are interested in. POI can range from 1 (default) to the number of columns in 'response'.
responsevector	(deprecated), please use the the more general 'response' variable instead.

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)

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

---

returnLocalMaxima *Local maximum detection*

Description

Find and return local maximum of a single spectrum.

Usage

```
returnLocalMaxima(spectrum)
```

Arguments

spectrum	A spectral sample in the vector format.

Value

list of 2: locMax (Locations of the found local maximum peaks) and pkMax (Intensities of the found local maximum peaks)

Author(s)

Trung Nghia Vu

Examples

```
res=makeSimulatedData();
X=res$data;
groupLabel=res$label;
returnLocalMaxima(X[2,])
```

`ROIplot` *Plot NMR spectra, together with raw and grouped peaks*

Description

This function plots NMR spectra, peak plots and grouped peak plots all in figure for easy comparison.

Usage

```
ROIplot(
  Y.spec,
  X.ppm,
  ungrouped.peaks,
  grouped.peaks,
  ROI = NULL,
  ROI.ppm = NULL,
  roiWidth = 100,
  roiWidth.ppm = NULL,
  groupLabels = NULL,
  output = NULL
)
```

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
`ungrouped.peaks`	(required) The data resulting from peak detection with wavelets
`grouped.peaks`	(required) The data after grouping (with PeakGrouper)
`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).
groupLabels	The vector with group labels (as factors)
output	Whether to return a plot (default), or the individual ggplot objects (output = "ggObjects")

Value

a plot

Author(s)

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

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm,
                              nCPU = 1) # nCPU set to 2 for the vignette build

test.grouped <- PeakGrouper(Y.peaks = test.peaks)

ROI.ppm <- 4.9
roiWidth.ppm <- 0.15

plots <- ROIplot(Y.spec = subset.spectra,
                 X.ppm =subset.ppm,
                 ungrouped.peaks = test.peaks,
                 grouped.peaks = test.grouped ,
                 ROI.ppm = ROI.ppm,
                 roiWidth.ppm = roiWidth.ppm ,
                 output = "ggObjects"
                 )
```

SCANT *SCAle, Normalize and Transform a data matrix*

Description

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

Usage

```
SCANT(data.matrix, type = "unit", feature_orientation = "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 sequentially. Any of 'unit', 'pareto', 'log10', 'log2', 'center', 'range', 'vast', or 'max' are accepted.
feature_orientation	default = "columns". This corresponds to the default feature matrix with samples as rows and features as columns. The other option is "rows": samples as columns and different features as different rows.

Value

The scaled, normalized and/or transformed matrix.

Author(s)

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

References

van den Berg RA, Hoefsloot HCJ, Westerhuis JA, et al. Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics 2006; 7:142.

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'), feature_orientation = 'columns')
```

SilhouetR *SilhouetR*

Description

This function calculate Silhouette values. The function is generic, as such silhouette values can be calculated between samples of different classes or it can be used to calculate silhouette values between different groups of peaks. This is the way in which it is used for the speaq package (see the example).

Usage

```
SilhouetR(DataMatrix, GroupIndices, distance = "euclidean")
```

Arguments

DataMatrix	a matrix with the raw data, 1 variable per column.
GroupIndices	The vector with the group indices (length must be equal to the amount of rows in DataMatrix).
distance	The distance metric to be used, "euclidean" or "manhattan".

Value

Returns the silhouette values. Note if a group contains only 1 no Silhouette value can be calculated (will give NA)

Author(s)

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

Examples

```
subset <- GetWinedata.subset()
# to reduce the example time we only select spectra 1 & 2
subset.spectra = as.matrix(subset$Spectra)[1:2,]
subset.ppm = as.numeric(subset$PPM)

test.peaks <- getWaveletPeaks(Y.spec=subset.spectra,
                              X.ppm=subset.ppm,
                              nCPU = 1) # nCPU set to 2 for the vignette build

test.grouped <- PeakGrouper(Y.peaks = test.peaks)

Silhouette.values = SilhouetR(DataMatrix = test.grouped$peakPPM,
                              test.grouped$peakIndex,
                              distance = "euclidean")

hist(Silhouette.values$SilhouetteValues)
```

`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. Available at 'models.life.ku.dk/datasets'

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