Package: Rnmr1D (via r-universe)

March 26, 2025

Type Package

Title Perform the Complete Processing of a Set of Proton Nuclear Magnetic Resonance Spectra

Version 1.6.0

Date 2023-06-24

Copyright Institut national de recherche pour l'agriculture, l'alimentation et l'environnement (INRAE)

URL `https://github.com/INRA/Rnmr1D`

Encoding UTF-8

Maintainer Daniel Jacob <`daniel.jacob@inrae.fr`>

Description Perform the complete processing of a set of proton nuclear magnetic resonance spectra from the free induction decay (raw data) and based on a processing sequence (macro-command file). An additional file specifies all the spectra to be considered by associating their sample code as well as the levels of experimental factors to which they belong. More detail can be found in Jacob et al. (2017) <doi:10.1007/s11306-017-1178-y>.

Depends R (>= 3.1.0)

License GPL (>= 2)

Imports Rcpp (>= 0.12.7), base64enc (>= 0.1), MASS(>= 7.3), Matrix, methods, scales, doParallel (>= 1.0.11), foreach (>= 1.4.4), igraph (>= 1.2.1), impute (>= 1.54.0), MassSpecWavelet (>= 1.46.0), ptw (>= 1.9), signal (>= 0.7), XML (>= 3.98), ggplot2 (>= 3.0.0), plotly (>= 4.8.0), plyr (>= 1.8.4), minqa(>= 1.2.4)

LinkingTo Rcpp

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev libssl-dev

Repository https://inra.r-universe.dev

RemoteUrl https://github.com/inra/rnmr1d

RemoteRef HEAD

RemoteSha 0fbe1338d7d273d87039ff3b91b2dfc85a3e6a5c

Contents

`calibSpectrum` *calibSpectrum*

Description

`calibSpectrum` belongs to the low-level functions group - it processes only one raw spectrum at time.

Usage

```
calibSpectrum(spec, zoneref, ppmref)
```

Arguments

`spec`	a spectrum object returned by the readSpectrum function
`zoneref`	the ppm range containing the TSP/DSS signal
`ppmref`	the ppm reference value

Value

spec object

checkMacroCmdFile *checkMacroCmdFile*

Description

checkMacroCmdFile Check if the macro-commands included in the input file (commandfile) are compliant with the allowed commands.

Usage

```
checkMacroCmdFile(commandfile)
```

Arguments

commandfile	The macro-commands file - the allowed commands are : 'align', 'warp', 'clupa', 'gbaseline', 'baseline', 'qnmrbline', 'airpls', 'binning', 'calibration', 'normalisation', 'denoising', 'bucket', 'zero'.

Value

return 1 if the macro-commands included in the input file are compliant, 0 if not.

See Also

the NMRProcFlow online documentation https://nmrprocflow.org/ and especially the Macro-command Reference Guide (https://nmrprocflow.org/themes/pdf/Macrocommand.pdf)

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
CMDFILE <- file.path(data_dir, "NP_macro_cmd.txt")
ret <- checkMacroCmdFile(CMDFILE)
```

cleanPeaks *cleanPeaks*

Description

cleanPeaks cleans the peaks under a specified threshold and also remove redundant peaks having the same position

Usage

```
cleanPeaks(spec, peaks, ratioPN, keeprows = FALSE)
```

Arguments

`spec`	a 'spec' object
`peaks`	a data.frame of the input peaks
`ratioPN`	Threshold of the Peak/Noise ratio below which the peaks will be rejected
`keeprows`	indicates if row names must be preserved.

Value

a data.frame of the remaining peaks

`computeBL` *computeBL*

Description

`computeBL` computes baseline based on the model.

Usage

```
computeBL(spec, model)
```

Arguments

`spec`	a 'spec' object
`model`	a model object

Value

a vector of the baseline estimated during the deconvolution process

`deconvParams` *deconvParams*

Description

Initialize the deconvolution parameter list

Usage

```
deconvParams
```

Format

An object of class `list` of length 45.

Value

- `flist` : Filter type list : 'smooth1', 'smooth2' and'smooth3' for Savitzky-Golay filter, 'daub8' and 'symlet8' for filter based on wavelet
- `criterion` : Criterion type for the optimizations : 0 => R2, 1 => 1/Std(residues) - default value = 0
- `reltol` : Criterion tolerance for the optimization - default value = 0.0001
- `facN` : Noise factor applied while the peak finding step - default value = NULL
- `ratioPN` : Peak/Noise Ratio applied while the peak selection step - default value = 1
- `obl` : Optimization of a baseline (BL) for each massif. 0 means no BL, an integer greater than 0 indicates the polynomial order of the BL default value = 0
- `distPeaks` : PeakFinder : min distance between 2 peaks (as multiple of sigma_min which is typically equal to 0.0005 ppm) - default value = 2
- `optim` : Indicates if optimisation is applied - default value = 1
- `oppm` : Indicates if ppm optimisation is applied - default value = 1
- `osigma` : Indicates if sigma optimisation is applied - default value = 1
- `d2meth` : PeakFinder : Indicates if minima method to the second derivation is applied
- `spcv` : PeakFinder : Maximal CV on Spectrum - default value = 0.005
- `d2cv` : PeakFinder : Maximum CV on the derived spectrum - default value = 0.05
- `d1filt` : Apply Filter (1) on the 1st derivate or not (0) - default value = 0
- `d2filt` : Apply Filter (1) on the 2nd derivate or not (0) - default value = 0
- `sigma_min` : Optimization of Sigmas : Fixe the minimum limit of sigmas - default value = 0.0005
- `sigma_max` : Optimization of Sigmas : Fixe the maximum limit of sigmas - default value = 0.005
- `verbose` : Indicates if we want information messages - default value = 1
- `exclude_zones` : Exclude ppm zones for the criterion evaluation - default value = NULL

`detectCores` *detectCores*

Description

`detectCores` is simply a shortcut for parallel::detectCores().

Usage

```
detectCores(...)
```

Arguments

`...`	See `parallel::detectCores`

doProcCmd *doProcCmd*

Description

doProcCmd it process the Macro-commands string array specified at input.

Usage

```
doProcCmd(specObj, cmdstr, ncpu = 1, debug = FALSE)
```

Arguments

specObj	a complex list return by doProcessing function. See the manual page of the doProcessing function for more details on its structure.
cmdstr	the Macro-commands string array; See the Macro-command Reference Guide (https://nmrprocflow.org/themes/pdf/Macrocommand.pdf) to have more details about macro-commands.
ncpu	The number of cores [default: 1]
debug	a boolean to specify if we want the function to be more verbose.

Value

specMat : a 'specMat' object - See the manual page of the doProcessing function for more details on its structure

Examples

```
    data_dir <- system.file("extra", package = "Rnmr1D")
    CMDFILE <- file.path(data_dir, "NP_macro_cmd.txt")
    SAMPLEFILE <- file.path(data_dir, "Samples.txt")
    out <- Rnmr1D::doProcessing(data_dir, cmdfile=CMDFILE,
                                samplefile=SAMPLEFILE, ncpu=2)
# Apply an intelligent bucketing (AIBIN)
    specMat.new <- Rnmr1D::doProcCmd(out,
             c("bucket aibin 10.2 10.5 0.3 3 0",
                 "9.5 4.9",
                 "4.8 0.5",
                 "EOL"
              ),ncpu=2, debug=TRUE)
    out$specMat <- specMat.new
```

doProcessing *doProcessing*

Description

`doProcessing` is the main function of this package. Indeed, this function performs the complete processing of a set of 1D NMR spectra from the FID (raw data) and based on a processing sequence (macro-command file). An additional file specifies all the spectra to be considered by associating their sample code as well as the levels of experimental factors to which they belong. In this way it is possible to select only a subset of spectra instead of the whole set.

Usage

```
doProcessing(
  path,
  cmdfile,
  samplefile = NULL,
  bucketfile = NULL,
  phcfile = NULL,
  ncpu = 1
)
```

Arguments

`path`	The full path of either the raw spectra directory on the disk
`cmdfile`	The full path name of the Macro-commands file for processing (text format)
`samplefile`	The full path name of the Sample file (tabular format)
`bucketfile`	The full path name of the file of bucket's zones (tabular format)
`phcfile`	The full path name of the phasing file for samples if required (tabular format)
`ncpu`	The number of cores [default: 1]

Value

`doProcessing` returns a list containing the following components:

- `samples` : the samples matrix with the correspondence of the raw spectra, as well as the levels of the experimental factors if specified in the input.
- `factors` : the factors matrix with the corresponding factor names. At minimum, the list contains the Samplecode label corresponding to the samples without their group level.
- `rawids` : list of the full directories of the raw spectra (i.e. where the FID files are accessible)
- `infos` : list of the acquisition and processing parameters for each (raw) spectra.
- `specMat` : objects list regarding the spectra data.
  - `int` : the matrix of the spectra data (`nspec` rows X `size` columns)
  - `nspec` : the number of spectra

- `size` : the size (i.e number of points) of each spectra
- `ppm_min`, `ppm_max` : the minimum and the maximum ppm values of spectra
- `ppm` : the vector of the ppm values (`size` values)
- `dppm` : the ppm increment between each point
- `buckets_zones` : the matrix of the buckets zones including two columns (min and max)
- `namesASintMax` : boolean - If TRUE, generate all output matrix with bucket names based on ppm values of the maximum of the average intensity of all spectra within the ppm range of each bucket. If FALSE (default), then bucket names will be based on the ppm range center of each bucket.

See Also

the NMRProcFlow online documentation `https://nmrprocflow.org/` and especially the Macro-command Reference Guide (`https://nmrprocflow.org/themes/pdf/Macrocommand.pdf`)

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
cmdfile <- file.path(data_dir, "NP_macro_cmd.txt")
samplefile <- file.path(data_dir, "Samples.txt")
out <- Rnmr1D::doProcessing(data_dir, cmdfile=cmdfile,
                            samplefile=samplefile, ncpu=2)
```

`estimateBL` *estimateBL*

Description

`estimateBL` estimates of the baseline of the spectrum in the corresponding ppm range (based on the C_Estime_LB routine)

Usage

```
estimateBL(spec, ppmrange, WS = 50, NEIGH = 35)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	the ppm range in which the baseline will be estimated
`WS`	Size of the window (in number of points) from which a rolling average will be established
`NEIGH`	The minimum window size (in number of points) in which the signal compared to its mean can be considered as belonging to the baseline.

Value

a vector of the estimated baseline

`filterByThreshold` *filterByThreshold*

Description

`filterByThreshold` applies a filtering based on wavelet by specifying a threshold value

Usage

```
filterByThreshold(s, wavelet, type = 0)
```

Arguments

`s`	the spectral signal as a numerical vector
`wavelet`	the name of the wavelet: haar, daub2, daub4, daub8, symlet2, symlet4, symlet8
`type`	the type of the threshold : 0 for Soft threshold, 1 for Hard threshold

Value

a vector of the same dimension as the entry one

`filterByWT` *filterByWT*

Description

`filterByWT` applies a filtering based on wavelet using the universal threshold

Usage

```
filterByWT(s, wavelet, threshold = 0.5)
```

Arguments

`s`	the spectral signal as a numerical vector
`wavelet`	the name of the wavelet: haar, daub2, daub4, daub8, symlet2, symlet4, symlet8
`threshold`	the threshold value - default value is 0.5

Value

a vector of the same dimension as the entry one

filterSavGol *filterSavGol*

Description

`filterSavGol` applies a Savitzky-Golay filter on a spectral signal.

Usage

```
filterSavGol(s, m, nl, nr)
```

Arguments

`s`	the spectral signal as a numerical vector
`m`	the degree of the polynomial filter (integer)
`nl`	width of the sliding window on the left (integer)
`nr`	width of the sliding window on the rigth (integer)

Value

a vector of the same dimension as the entry one

generateMetadata *generateMetadata*

Description

`generateMetadata` Generate the metadata from the list of raw spectra namely the samples, the experimental factors and the list of selected raw spectra. Depending on whether the sample matrix is supplied as input or not,

Usage

```
generateMetadata(RAWDIR, procParams, samples = NULL)
```

Arguments

`RAWDIR`	The full path of either the raw spectra directory on the disk
`procParams`	the list of processing parameters. First initialize this list with the `Spec1r.Procpar.default` list, then modify parameters depending of your spectra set.
`samples`	the samples matrix with the correspondence of the raw spectra

Value

generateMetadata returns a list containing the following components:

- samples : the samples matrix with the correspondence of the raw spectra, as well as the levels of the experimental factors if specified in the input.
- factors : the factors matrix with the corresponding factor names. At minimum, the list contains the Samplecode label corresponding to the samples without their group level.
- rawids : list of the full directories of the raw spectra (i.e. where the FID files are accessible)

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
samplefile <- file.path(data_dir, "Samples.txt")
samples <- read.table(samplefile, sep="\t", header=TRUE,stringsAsFactors=FALSE)
metadata <- generateMetadata(data_dir, procParams=Spec1rProcpar, samples)
```

---

getBucketsDataset *getBucketsDataset*

Description

Generates the matrix including the integrations of the areas defined by the buckets (columns) on each spectrum (rows)

Usage

```
getBucketsDataset(specObj, norm_meth = "none", zoneref = NA)
```

Arguments

specObj	a complex list return by doProcessing function. See the manual page of the doProcessing function for more details on its structure.
norm_meth	Normalization method. The possible values are : 'none', 'CSN' or 'PDN'. See below.
zoneref	Specify the ppm zone of the internal reference (i.e. ERETIC) if applicable. default is NA.

Details

Before bucket data export in order to make all spectra comparable with each other, the variations of the overall concentrations of samples have to be taken into account. We propose two normalization methods. In NMR metabolomics, the total intensity normalization (called the Constant Sum Normalization) is often used so that all spectra correspond to the same overall concentration. It simply consists in normalizing the total intensity of each individual spectrum to a same value. An other method called Probabilistic Quotient Normalization (Dieterle et al. 2006) assumes that biologically interesting concentration changes influence only parts of the NMR spectrum, while dilution effects will affect all metabolites signals. Probabilistic Quotient Normalization (PQN) starts by the calculation of a reference spectrum based on the median spectrum. Next, for each variable of interest the quotient of a given test spectrum and reference spectrum is calculated and the median of all quotients is estimated. Finally, all variables of the test spectrum are divided by the median quotient. An internal reference can be used to normalize the data. For example, an electronic reference (ERETIC, see Akoka et al. 1999, or ERETIC2 generated with TopSpin software) can be used for this purpose. The integral value of each bucket will be divided by the integral value of the ppm range given as reference.

Value

the data matrix

References

Akoka S1, Barantin L, Trierweiler M. (1999) Concentration Measurement by Proton NMR Using the ERETIC Method, Anal. Chem 71(13):2554-7. doi: 10.1021/ac981422i.

Dieterle F., Ross A., Schlotterbeck G. and Senn H. (2006). Probabilistic Quotient Normalization as Robust Method to Account for Dilution of Complex Biological Mixtures. Application in 1H NMR Metabonomics. Analytical Chemistry, 78:4281-4290.doi: 10.1021/ac051632c

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
cmdfile <- file.path(data_dir, "NP_macro_cmd.txt")
samplefile <- file.path(data_dir, "Samples.txt")
out <- Rnmr1D::doProcessing(data_dir, cmdfile=cmdfile,
                              samplefile=samplefile, ncpu=2)
outMat <- getBucketsDataset(out, norm_meth='CSN')
```

`getBucketsTable` *getBucketsTable*

Description

Generates the buckets table

Usage

```
getBucketsTable(specObj)
```

Arguments

`specObj`	a complex list return by `doProcessing` function. See the manual page of the `doProcessing` function for more details on its structure.

Value

the buckets table

getClusters *getClusters*

Description

From the data matrix generated from the integration of all bucket zones (columns) for each spectrum (rows), we can take advantage of the concentration variability of each compound in a series of samples by performing a clustering based on significant correlations that link these buckets together into clusters. Bucket Clustering based on either a lower threshold applied on correlations or a cutting value applied on a hierarchical tree of the variables (buckets) generated by an Hierarchical Clustering Analysis (HCA).

Usage

```
getClusters(data, method = "hca", ...)
```

Arguments

`data`	the matrix including the integrations of the areas defined by the buckets (columns) on each spectrum (rows)
`method`	Clustering method of the buckets. Either 'corr' for 'correlation' or 'hca' for 'hierarchical clustering analysis'.
`...`	Depending on the chosen method: • `corr` : cval, dC, ncpu • `hca` : vcutusr

Details

At the bucketing step (see above), we have chosen the intelligent bucketing, it means that each bucket exact matches with one resonance peak. Thanks to this, the buckets now have a strong chemical meaning, since the resonance peaks are the fingerprints of chemical compounds. However, to assign a chemical compound, several resonance peaks are generally required in 1D 1 H-NMR metabolic profiling. To generate relevant clusters (i.e. clusters possibly matching to chemical compounds), two approaches have been implemented:

- Bucket Clustering based on a lower threshold applied on correlations
  - In this approach an appropriate correlation threshold is applied on the correlation matrix before its cluster decomposition. Moreover, an improvement can be done by searching for a trade-off on a tolerance interval of the correlation threshold : from a fixed threshold of the correlation (cval), the clustering is calculated for the three values (cval-dC, cval, cval+dC), where dC is the tolerance interval of the correlation threshold. From these three sets of clusters, we establish a merger according to the following rules: 1) if a large cluster is broken, we keep the two resulting clusters. 2) If a small cluster disappears, the initial cluster is conserved. Generally, an interval of the correlation threshold included between 0.002 and 0.01 gives good trade-off.

- Bucket Clustering based on a hierarchical tree of the variables (buckets) generated by an Hierarchical Clustering Analysis (HCA)
  - In this approach a Hierachical Classification Analysis (HCA, `hclust`) is applied on the data after calculating a matrix distance ("euclidian" by default). Then, a cut is applied on the tree (`cutree`) resulting from `hclust`, into several groups by specifying the cut height(s). For finding best cut value, the cut height is chosen i) by testing several values equally spaced in a given range of the cut height, then, 2) by keeping the one that gives the more cluster and by including most bucket variables. Otherwise, a cut value has to be specified by the user (vcutusr)

Value

`getClusters` returns a list containing the following components:

- `vstats` Statistics that served to find the best value of the criterion (matrix)
- `clusters` List of the ppm value corresponding to each cluster. the length of the list equal to number of clusters
- `clustertab` the associations matrix that gives for each cluster (column 2) the corresponding buckets (column 1)
- `params` List of parameters related to the chosen method for which the clustering was performed.
- `vcrit` Value of the (best/user) criterion, i.e correlation threshold for 'corr' method or the cut value for the 'hca' method.
- `indxopt` Index value within the vstats matrix corresponding to the criterion value (vcrit)

References

Jacob D., Deborde C. and Moing A. (2013) An efficient spectra processing method for metabolite identification from 1H-NMR metabolomics data. Analytical and Bioanalytical Chemistry 405(15) 5049-5061 doi: 10.1007/s00216-013-6852-y

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
cmdfile <- file.path(data_dir, "NP_macro_cmd.txt")
samplefile <- file.path(data_dir, "Samples.txt")
out <- Rnmr1D::doProcessing(data_dir, cmdfile=cmdfile,
                              samplefile=samplefile, ncpu=2)
outMat <- getBucketsDataset(out, norm_meth='CSN')
clustcorr <- getClusters(outMat, method='corr', cval=0, dC=0.003, ncpu=2)
clusthca <- getClusters(outMat, method='hca', vcutusr=0)
```

`getDeconvParams` *getDeconvParams*

Description

`getDeconvParams` merges some specific parameters values with the full deconvolution list and return the resulting list. With no parameter as input, it returns the default parameter list.

Usage

```
getDeconvParams(params = NULL)
```

Arguments

`params`	a list defining some specific parameters for deconvolution

Value

the resulting list of deconvolution parameters.

`getMergedDataset` *getMergedDataset*

Description

merged variables for each cluster (based on their average)

Usage

```
getMergedDataset(data, clustObj, onlycluster = FALSE)
```

Arguments

`data`	the matrix including the integrations of the areas defined by the buckets (columns) on each spectrum (rows)
`clustObj`	a list generated by the `getClusters` function
`onlycluster`	boolean - specifies if the merged data matrix at output must only contain the merged clusters (TRUE) or if it must also contain the buckets that are not include within a cluster (FALSE)

getSlices *getSlices*

Description

Slice the spectrum in order to define ranges for Local Deconvolution (LSDeconv) and return only those include the provided ppmranges

Usage

```
getSlices(spec, ppmranges, flatwidth = 0.004, snrfactor = 4, maxrange = 0.3)
```

Arguments

`spec`	a 'spec' object
`ppmranges`	ppm ranges as a matrix in order to apply the deconvolution, each row specifying a zone
`flatwidth`	specifies the minimum width of a zone in which the spectrum intensities are close to zero to consider this one as a cutting zone (default 0.003 ppm)
`snrfactor`	specifies factor applied on the Std Dev. of the noise used as threshold for first derivate intensities (default=4)
`maxrange`	specifies the maximum width of a cutting zone (default 0.3 ppm)

Value

a list of ppm range

getSnrDataset *getSnrDataset*

Description

Generates the Signal-Noise-Ratio dataset

Usage

```
getSnrDataset(specObj, zone_noise = c(10.2, 10.5), ratio = TRUE)
```

Arguments

`specObj`	a complex list return by `doProcessing` function. See the manual page of the `doProcessing` function for more details on its structure.
`zone_noise`	Specify a ppm range of noisy zone default is c(10.2,10.5)
`ratio`	boolean; TRUE for output Signal-Noise Ratio, or FALSE to output maximum value of each bucket and in addition, the estimate noise as a separate column

Details

whatever the bucketing approach used, the Signal-to-Noise ratio is a good quality indicator. Thus, it is possible to check buckets based on their Signal-to-Noise ratio.

Value

the Signal-Noise-Ratio matrix

`getSpectraData` *getSnrDataset*

Description

Generates the spectral data matrix. The first column indicates the value of ppm, then the following columns correspond to spectral data, one column per spectrum.

Usage

```
getSpectraData(specObj)
```

Arguments

`specObj`	a complex list return by `doProcessing` function.

Value

the spectral data matrix

`ggplotClusters` *ggplotClusters*

Description

Plots the boxplot of all clusters allowing to have an insight on the clusters distribution. Plot based on ggplot2

Usage

```
ggplotClusters(data, clustObj)
```

Arguments

`data`	the matrix including the integrations of the areas defined by the buckets (columns) on each spectrum (rows)
`clustObj`	a list generated by the `getClusters` function

`ggplotCriterion` *ggplotCriterion*

Description

Plots the curves that show the number of clusters, the number of clustered buckets and the size of biggest cluster versus the criterion, namely the correlation threshold for the 'corr' method, the cutting value for the 'hca' method.

Usage

```
ggplotCriterion(clustObj, reverse = FALSE)
```

Arguments

`clustObj`	a list generated by the `getClusters` function
`reverse`	indicates if the x axis need to be reversed

`ggplotLoadings` *plotLoadings*

Description

Plots the two components defined by pc1, pc2 of the matrix of variable loadings coming from a multivariable analysis, typically a Principal Component Analysis (PCA). It can also plot the ellipses corresponding to each cluster defined by the associations matrix if not null. (in fact it is the main interest of this function).

Usage

```
ggplotLoadings(
  data,
  pc1 = 1,
  pc2 = 2,
  EV = NULL,
  associations = NULL,
  main = "Loadings",
  onlylabels = FALSE,
  highlabels = FALSE,
  gcontour = "ellipse"
)
```

Arguments

`data`	the matrix of variable loadings coming from a multivariable analysis, typically a Principal Component Analysis (PCA)
`pc1`	the fist component of the matrix of variable loadings to be plotted.
`pc2`	the second component of the matrix of variable loadings to be plotted.
`EV`	Eigenvalues vector
`associations`	the associations matrix that gives for each cluster (column 2) the corresponding buckets (column 1). See `getClusters`
`main`	Change the default plot title on the rigth corner
`onlylabels`	if TRUE, put only the association names without drawing the cluster contours. Implies that association matrix is provided.
`highlabels`	if TRUE, put the the association names in blue, and others in grey. Implies that association matrix is provided and fONLYLABELS equal to TRUE.
`gcontour`	type of contour; possible values are : 'ellipse', 'polygon', 'ellipse2', 'none'

`ggplotPlotly` *ggplotPlotly*

Description

Translate 'ggplot2' graphs to an interactive plotly version

Usage

```
ggplotPlotly(g, width = NULL, height = NULL, textposition = "right")
```

Arguments

`g`	The ggplot2 graph object to be translated into an interactive plotly version
`width`	Width of the plot in pixels (optional, defaults to automatic sizing).
`height`	Height of the plot in pixels (optional, defaults to automatic sizing)
`textposition`	Position of the labels on the graphs relative to the points. Possible values are : 'right', 'left', 'top' or 'buttom'

ggplotScores *ggplotScores*

Description

Plots the two components defined by pc1, pc2 of the matrix of scores coming from a multivariable analysis, typically a Principal Component Analysis (PCA).

Usage

```
ggplotScores(
  data,
  pc1 = 1,
  pc2 = 2,
  groups = NULL,
  EV = NULL,
  main = "Scores",
  glabels = FALSE,
  psize = 3,
  gcontour = "ellipse",
  params = list(cellipse = 0.95),
  colors = NULL
)
```

Arguments

`data`	the matrix of scores coming from a multivariable analysis, typically a Principal Component Analysis (PCA)
`pc1`	the fist component of the matrix of variable loadings to be plotted.
`pc2`	the second component of the matrix of variable loadings to be plotted.
`groups`	the vector defining the factorial groups (same dimension as data rows)
`EV`	Eigenvalues vector
`main`	the plot main title
`glabels`	boolean indicating if labels have to be plotted
`psize`	point size
`gcontour`	type of contour; possible values are : 'ellipse', 'polygon', 'ellipse2', 'none'
`params`	parameters depending on the contour type

GSDeconv *GSDeconv*

Description

Global Spectra Deconvolution: `GSDeconv` belongs to the low-level functions group for deconvolution.

Usage

```
GSDeconv(
  spec,
  ppmrange,
  params = NULL,
  filter = "symlet8",
  scset = c(2, 3, 12),
  verbose = 1
)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`params`	a list of specific parameters for deconvolution
`filter`	a filter type for filtering the noise and smoothing the signal
`scset`	a set of scmin values
`verbose`	level of debug information

Value

a model object

Lorentz *Lorentz*

Description

`Lorentz` belongs to the low-level functions group for deconvolution.

Usage

```
Lorentz(ppm, amp, x0, sigma, asym)
```

Arguments

`ppm`	a vector of ppm values
`amp`	amplitude of the lorentzian
`x0`	central value of the lorentzian
`sigma`	half-width of the lorentzian
`asym`	asymetric parameter

Value

a vector of the lorentzian values (same size as ppm)

`LSDeconv` *LSDeconv*

Description

Local Spectra Deconvolution: `LSDeconv` belongs to the low-level functions group for deconvolution.

Usage

```
LSDeconv(
  spec,
  ppmrange,
  params = NULL,
  filterset = c("daub8"),
  oblset = 0,
  verbose = 1
)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`params`	a list of specific parameters for deconvolution including or not (i.e equal to NULL) the matrix defining peaks, one peak by row, with columns defined as : pos, ppm, amp, sigma, eta
`filterset`	a set of filter type for filtering the noise and smoothing the signal (only if the matrix defining peaks not defined in order to find peaks)
`oblset`	a set of baseline order for fitting
`verbose`	level of debug information

Value

a model object

MultiLSDeconv *MultiLSDeconv*

Description

Multiple Local Spectra Deconvolution: `MultiLSDeconv` belongs to the low-level functions group for deconvolution.

Usage

```
MultiLSDeconv(
  spec,
  ppmranges = NULL,
  params = NULL,
  filterset = c(7, 9),
  oblset = 0,
  ncpu = 4,
  verbose = 0
)
```

Arguments

`spec`	a 'spec' object
`ppmranges`	ppm ranges as a matrix in order to apply the deconvolution, each row specifying a zone
`params`	a list of specific parameters for deconvolution
`filterset`	a set of filter type for filtering the noise and smoothing the signal (only if the matrix defining peaks not defined in order to find peaks)
`oblset`	a set of baseline order for fitting
`ncpu`	number of CPU for parallel computing
`verbose`	level of debug information

Value

a model object

`optimOneVoigt` *optimOneVoigt*

Description

`optimOneVoigt` belongs to the low-level functions group for deconvolution.

Usage

```
optimOneVoigt(X, Y, par)
```

Arguments

`X`	a vector of ppm values
`Y`	a vector of intensities
`par`	a vector of the 3 pseudo-voigt parameters namely: Amplitude, central ppm value, 2 ppm widths at mid-height for mixed lorentizian and gaussian

Value

a vector of the pseudo-voigt parameters (same size as par)

`peakFiltering` *peakFiltering*

Description

`peakFiltering` belongs to the low-level functions group for deconvolution.

Usage

```
peakFiltering(spec, peaks, ratioPN)
```

Arguments

`spec`	a 'spec' object
`peaks`	the matrix defining peaks, one peak by row, with columns defined as : pos, ppm, amp, sigma, eta
`ratioPN`	the ratio Peaks/Noise for filtering

Value

a dataframe

peakFinder *peakFinder*

Description

`peakFinder` belongs to the low-level functions group for deconvolution.

Usage

```
peakFinder(spec, ppmrange, params = NULL, filter = "none", verbose = 1)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`params`	a list of specific parameters for deconvolution
`filter`	a filter type for filtering the noise and smoothing the signal
`verbose`	level of debug information

Value

a list

peakOptimize *peakOptimize*

Description

`peakOptimize` belongs to the low-level functions group for deconvolution.

Usage

```
peakOptimize(spec, ppmrange, params, verbose = 1)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`params`	a list of specific parameters for deconvolution, including the matrix defining peaks, one peak by row, with columns defined as : pos, ppm, amp, sigma, eta
`verbose`	level of debug information

Value

a list

plotClusters *plotClusters*

Description

Plots the boxplot of all clusters allowing to have an insight on the clusters distribution

Usage

```
plotClusters(
  data,
  clustObj,
  horiz = TRUE,
  main = "Boxplot by clusters (log10 transformed)"
)
```

Arguments

`data`	the matrix including the integrations of the areas defined by the buckets (columns) on each spectrum (rows)
`clustObj`	a list generated by the `getClusters` function
`horiz`	Boolean - Indicates if the plot is horizontal (TRUE) or vertical (FALSE)
`main`	Main title of the plot

plotCriterion *plotCriterion*

Description

Plots the curves that show the number of clusters, the number of clustered buckets and the size of biggest cluster versus the criterion, namely the correlation threshold for the 'corr' method, the cutting value for the 'hca' method.

Usage

```
plotCriterion(clustObj, reverse = FALSE)
```

Arguments

`clustObj`	a list generated by the `getClusters` function
`reverse`	Boolean - indicates if x-axis must be reversed (TRUE) or nor (FALSE)

plotLoadings *plotLoadings*

Description

Plots the two components defined by pc1, pc2 of the matrix of variable loadings coming from a multivariable analysis, typically a Principal Component Analysis (PCA). It can also plot the ellipses corresponding to each cluster defined by the associations matrix if not null. (in fact it is the main interest of this function).

Usage

```
plotLoadings(
  data,
  pc1,
  pc2,
  associations = NULL,
  main = "Loadings",
  xlimu = c(min(data[, pc1]), max(data[, pc1])),
  ylimu = c(min(data[, pc2]), max(data[, pc2])),
  cexlabel = 1,
  pch = 20,
  ellipse = TRUE,
  level = 0.8
)
```

Arguments

`data`	the matrix of variable loadings coming from a multivariable analysis, typically a Principal Component Analysis (PCA)
`pc1`	the fist component of the matrix of variable loadings to be plotted.
`pc2`	the second component of the matrix of variable loadings to be plotted.
`associations`	the associations matrix that gives for each cluster (column 2) the corresponding buckets (column 1)
`main`	Change the default plot title on the rigth corner
`xlimu`	gives the limit to be plotted of the first component
`ylimu`	gives the limit to be plotted of the second component
`cexlabel`	number indicating the amount by which plotting text and symbols should be scaled relative to the default.
`pch`	Plotting Symbols
`ellipse`	boolean - specifies if ellipses are plot or not for each cluster
`level`	confidence level for plotting the ellipses

`plotModel` *plotModel*

Description

`plotModel` plots the model along with the resulting voigt functions from deconvolution

Usage

```
plotModel(
  spec,
  model,
  exclude_zones = NULL,
  labels = c("ppm", "id"),
  groups = NULL,
  tags = FALSE,
  xlab = "",
  ylab = "",
  title = "",
  grp_colors = NULL
)
```

Arguments

`spec`	a 'spec' object (see `readSpectrum`, `Spec1rDoProc`)
`model`	a 'model' object (see `specDeconv`, `peakOptimize`, `GSDeconv`, `LSDeconv`)
`exclude_zones`	a list of vector defining the excluded zones for lorentzian plots
`labels`	choose as legend labels either 'ppm' or 'id'
`tags`	boolean allowing you to put identification tags at the top of each peak
`title`	title of the graphic
`grp_colors`	specifies the colors for the first groups and/or peaks

`plotScores` *plotScores*

Description

Plots the two components defined by pc1, pc2 of the matrix of scores coming from a multivariable analysis, typically a Principal Component Analysis (PCA).

Usage

```
plotScores(
  data,
  pc1,
  pc2,
  samples,
  factor = NULL,
  cexlabel = 1.2,
  level = 0.95,
  xlim = NULL,
  ylim = NULL,
  col = NULL
)
```

Arguments

`data`	the matrix of scores coming from a multivariable analysis, typically a Principal Component Analysis (PCA)
`pc1`	the fist component of the matrix of variable loadings to be plotted.
`pc2`	the second component of the matrix of variable loadings to be plotted. as well as the levels of the experimental factors if specified in the input. See `doProcessing` or `generateMetadata`
`samples`	the samples matrix with the correspondence of the raw spectra,
`factor`	if not null, the name of one of the columns defining the factorial groups in the samples matrix at input
`cexlabel`	number indicating the amount by which plotting text and symbols should be scaled relative to the default.
`level`	confidence level for plotting the corresponding ellipse
`xlim`	gives the limit to be plotted of the first component
`ylim`	gives the limit to be plotted of the second component
`col`	colors vector for ellipses - automatically defined by default

---

`plotSpec` *plotSpec*

Description

`plotSpec` plots all signals defined by a matrix.

Usage

```
plotSpec(
  ppmrange,
  x,
  y,
  ynames = c("Origin", "Filtered", "Model"),
 ycolors = c("grey", "blue", "red", "green", "orange", "magenta", "cyan", "darkgreen",
    "darkorange"),
  ysel = NULL,
  xlab = "",
  ylab = "",
  title = ""
)
```

Arguments

`ppmrange`	a ppm range defining the window of the plotting
`x`	a vector defining the x-axis (abscissa)
`y`	a vector or a matrix defining the y-axes (ordinates), each signal as a column
`ynames`	a vector defining the y names (same order as the y matrix)
`ycolors`	a vector defining the y colors (same order as the y matrix)
`ysel`	a vector defining the visibility of each y element (same order as the y matrix)
`title`	title of the graphic

`plotSpecMat` *plotSpecMat Overlaid/Stacked Plot*

Description

`plotSpecMat` Plot spectra set, overlaid or stacked; if stacked, plot with or without a perspective effect.

Usage

```
plotSpecMat(
  specMat,
  ppm_lim = c(min(specMat$ppm), max(specMat$ppm)),
  K = 0.67,
  pY = 1,
  dppm_max = 0.2 * (max(ppm_lim) - min(ppm_lim)),
  asym = 1,
  beta = 0,
  cols = NULL
)
```

Arguments

`specMat`	a 'specMat' object - Spectra matrix in specMat$int (rows = samples, columns = buckets)
`ppm_lim`	ppm range of the plot
`K`	Graphical height of the stack (0 .. 1),(default=0.67)
`pY`	Intensity limit factor (default 1)
`dppm_max`	Max ppm shift to have a perspective effect
`asym`	Correction of vertical parallax effect (-1 .. 1) -1 : parallelogram 0 : trapeze with maximum asymmetric 1 : symmetric trapeze
`beta`	Correction of horizontal parallax effect (0 .. 0.2) (defaut 0)
`cols`	Vector of colors (same size that the number of spectra, i.e dim(specmat)[1])

Examples

```
data_dir <- system.file("extra", package = "Rnmr1D")
cmdfile <- file.path(data_dir, "NP_macro_cmd.txt")
samplefile <- file.path(data_dir, "Samples.txt")
out <- Rnmr1D::doProcessing(data_dir, cmdfile=cmdfile,
                              samplefile=samplefile, ncpu=2)
# Overlaid plot
plotSpecMat(out$specMat, ppm_lim=c(0.5,9), K=0, pY=0.1)
# Stacked plot with perspective effect
plotSpecMat(out$specMat, ppm_lim=c(-0.1,9),K=0.33)
# Stacked plot with perspective effect with maximum asymmetric
plotSpecMat(out$specMat, ppm_lim=c(0.5,5), K=0.33, asym=0)
cols <- c(rep("red",3), rep("blue",3))
# Stacked plot with colors accordings to group levels
plotSpecMat(out$specMat, ppm_lim=c(0.5,5), K=0.67, dppm_max=0, cols=cols)
```

`pos2ppm` *pos2ppm*

Description

`pos2ppm` convert an index position to the corresponding ppm value

Usage

```
pos2ppm(spec, index)
```

Arguments

`spec`	a 'spec' object
`index`	an index position

Value

the corresponding ppm value

`ppm2pos` *ppm2pos*

Description

`ppm2pos` convert a ppm value to the corresponding index position

Usage

```
ppm2pos(spec, ppm)
```

Arguments

`spec`	a 'spec' object
`ppm`	a ppm value

Value

the corresponding index position

`PVoigt` *PVoigt*

Description

`PVoigt` belongs to the low-level functions group for deconvolution.

Usage

```
PVoigt(ppm, amp, x0, sigma, asym, eta)
```

Arguments

`ppm`	a vector of ppm values
`amp`	amplitude of the lorentzian
`x0`	central value of the lorentzian
`sigma`	half-width of both lorentzian and gaussian
`asym`	asymetric parameter
`eta`	mixing coefficient for the pseudo-voigt function (between 0 and 1)

Value

a vector of the lorentzian values (same size as ppm)

`readSpectrum` *readSpectrum*

Description

`read_spectrum` belongs to the low-level functions group - it processes only one raw spectrum at time.

Usage

```
readSpectrum(
  ACQDIR,
  procParams,
  ppmnoise = c(10.2, 10.5),
  PHC = NULL,
  scaleIntensity = 1,
  verbose = 1
)
```

Arguments

`ACQDIR`	Full directory path of the raw spectrum, i.e the directory containing the FID
`procParams`	a Spec1rProcpar list
`ppmnoise`	the ppm range containing only noise signal in order to estimate the level of the noise (S/N ratio)
`PHC`	the phasing values applied to the spectrum in the frequency domain thus avoiding the automatic phasing step. Only useful if the input signal is an FID (procParams$INPUT_SIGNAL)
`scaleIntensity`	factor applied to the intensities to establish a change of scale.
`verbose`	level of debug information

Value

spec object

`RWrapperCMD1D` *RWrapperCMD1D*

Description

`RWrapperCMD1D` belongs to the low-level functions group - it serves as a wrapper to call internale functions for processing.

Usage

```
RWrapperCMD1D(cmdName, specMat, ...)
```

Arguments

`cmdName`	the name of internal function
`specMat`	a 'specMat' object
`...`	specific parameters of the requested function

Value

`specMat` : a 'specMat' object

`setLogFile` *setLogFile*

Description

`setLogFile` allows to redirect all log messages to a file

Usage

```
setLogFile(con = stdout())
```

Arguments

`con`	a connection object which inherits from class "connection"

Examples

```
# Redirect all log messages to a temporary file
 outtmp <- tempfile()
 con <- file(outtmp, "wt", encoding = "UTF-8")
 setLogFile(con)
 data_dir <- system.file("extra", package = "Rnmr1D")
 RAWDIR <- file.path(data_dir, "CD_BBI_16P02")
 CMDFILE <- file.path(data_dir, "NP_macro_cmd.txt")
 SAMPLEFILE <- file.path(data_dir, "Samples.txt")
 out <- Rnmr1D::doProcessing(RAWDIR, cmdfile=CMDFILE, samplefile=SAMPLEFILE, ncpu=2)
 close(con)
 readLines(outtmp)
```

`setPPMbounds` *setPPMbounds*

Description

Set the PPM bounds for proton (1H) and carbon (13C) to consider in the processing step and then to store in the specMat object

Usage

```
setPPMbounds(proton = c(-0.5, 11), carbon = c(0, 200))
```

Arguments

`proton`	Minimal and Maximal ppm value for 1H NMR
`carbon`	Minimal and Maximal ppm value for 13C NMR

`sliceSpectrum` *sliceSpectrum*

Description

Slice the spectrum in order to define ranges for Local Deconvolution (LSDeconv)

Usage

```
sliceSpectrum(
  spec,
  ppmrange = c(0.5, 9.5),
  flatwidth = 0.004,
  snrfactor = 4,
  maxrange = 0.3,
  excludezones = NULL
)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`flatwidth`	specifies the minimum width of a zone in which the spectrum intensities are close to zero to consider this one as a cutting zone (default 0.003 ppm)
`snrfactor`	specifies factor applied on the Std Dev. of the noise used as threshold for first derivate intensities (default=4)
`maxrange`	specifies the maximum width of a cutting zone (default 0.3 ppm)
`excludezones`	specifies the exclusion zones as a matrix (Nx2), each row specifying a zone with 2 columns (ppm min and ppm max) (default NULL)

Value

a list of ppm range

`Spec1rDoProc` *Spec1rDoProc*

Description

`Spec1rDoProc` belongs to the low-level functions group - it processes only one raw spectrum at time.

Usage

```
Spec1rDoProc(Input, param = Spec1rProcpar)
```

Arguments

`Input`	Full directory path of the raw spectrum
`param`	a Spec1rProcpar list

Value

spec object

`Spec1rProcpar` *Spec1rProcpar*

Description

Initialize Parameter Lists by the default ones

Usage

```
Spec1rProcpar
```

Format

An object of class `list` of length 41.

Value

- `DEBUG` : Debug - defaut value = TRUE
- `LOGFILE` : Messages output file - default value = ""
- `VENDOR` : Instrumental origin of the raw data (bruker, varian, jeol, rs2d) - default value = 'bruker'
- `READ_RAW_ONLY` : Read raw file only; do not carry out processing; raw file is depending on INPUT_SIGNAL - default value = FALSE
- `INPUT_SIGNAL` : What type of input signal: 'fid' or '1r' - default value = 'fid'
- `PDATA_DIR` : subdirectory containing the 1r file (bruker's format only) - default value = 'pdata/1'
- `LB` : Exponantial Line Broadening parameter - default value = 0.3
- `GB` : Gaussian Line Broadening parameter - default value = 0
- `REMLFREQ` : Remove low frequencies by applying a polynomial subtraction method. - default order of the model = 0
- `REVPPM` : Reverse ppm scale - default value = FALSE
- `BLPHC` : Number of points for baseline smoothing during phasing - default value = 50
- `KSIG` : Number of times the noise signal to be considered during phasing - default value = 6
- `CPMG` : Indicate if CPMG sequence - default value = FALSE
- `ZEROFILLING` : Zero Filling - - default value = FALSE
- `ZFFAC` : Max factor for Zero Filling - default value = 4
- `LINEBROADENING` : Line Broading - default value = TRUE
- `TSP` : PPM referencing - default value = FALSE
- `RABOT` : Zeroing of Negative Values - default value = FALSE
- `OPTPHC0` : Zero order phase optimization - default value = TRUE
- `OPTPHC1` : First order phase optimization - default value = FALSE
- `OPTCRIT1` : Criterium for phasing optimization (1 for SSpos, 2 for SSneg, 3 for Entropy - default value = 2
- `JGD_INNER` : JEOL : internal (or external) estimation for Group Delay - default value = TRUE

`specDeconv` *specDeconv*

Description

`specDeconv` = `peakFinder` + `peakOptimize`. `specDeconv` belongs to the low-level functions group for deconvolution.

Usage

```
specDeconv(spec, ppmrange, params = NULL, filter = "none", verbose = 1)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`params`	a list of specific parameters for deconvolution
`filter`	a filter type for filtering the noise and smoothing the signal
`verbose`	level of debug information

Value

a list

`specModel` *specModel*

Description

`specModel` belongs to the low-level functions group for deconvolution.

Usage

```
specModel(spec, ppmrange, peaks)
```

Arguments

`spec`	a 'spec' object
`ppmrange`	a ppm range as a list in order to apply the deconvolution
`peaks`	a matrix defining peaks, one peak by row, with columns defined as : pos, ppm, amp, sigma, eta, integral

Value

a vector with the same size of the spectrum

Index