Package 'Rnmr1D'

September 11, 2018

Type Package

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

Version 1.2.1

Date 2018-09-10

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

Maintainer Daniel Jacob <daniel.jacob@inra.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 (>= 2.10)

License GPL (>= 2)

Imports Rcpp (>= 0.12.7), base64enc (>= 0.1), MASS(>= 7.3), Matrix, methods, 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), speaq (>= 2.3.3), XML (>= 3.98)

LinkingTo Rcpp

RoxygenNote 6.1.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Daniel Jacob [cre, aut] (<https://orcid.org/0000-0002-6687-7169>), Catherine Deborde [ctb], Marie Lefebvre [ctb]

Repository CRAN

Date/Publication 2018-09-11 10:40:03 UTC

R topics documented:

`Rnmr1D-package` *Rnmr1D is the main module in the NMRProcFlow web application (nmrprocflow.org) concerning the NMR spectra processing.*

Description

Rnmr1D R package is aimed to 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.

Details

NMRProcFlow allows experts to build their own spectra processing workflow, in order to become re-applicable to similar NMR spectra sets, i.e. stated as use-cases. By extension, the implementation of NMR spectra processing workflows executed in batch mode can be considered as relevant provided that we want to process in this way very well-mastered and very reproducible use cases, i.e. by applying the same Standard Operating Procedures (SOP). A subset of NMR spectra is firstly processed in interactive mode in order to build a well-suited workflow. This mode can be considered as the 'expert mode'. Then, other subsets that are regarded as either similar or being included in the same case study, can be processed in batch mode, operating directly within a R session.

This package consists of two sets of functions depending on the use: a so-called "high level" set and a so-called "low level" set.

* The "high level" group consists of the following functions :

detectCores, doProcessing, doProcCmd, getBucketsDataset, getBucketsTable, getClusters, getMergedDataset, getSnrDataset, getSpectraData, plotClusters, plotCriterion, plotLoadings, plotScores, plotSpecMat

This set of function, dedicated to 1D NMR spectra (1H & 13C) for metabolomics allow user to process a complet set of 1D NMR spectra from the FID (raw data) and visualize the resulting spectra in the frequency domain. In output, a plenty of data are available : Bucket table, Data matrix, Signal-to-noise ratios matrix, spectral data, metadata on both samples and factors. It is possible to reduce the variables size by gathering within clusters two or more buckets likely to be the (partial) spectral signature of a single chemical compound. The example below provides an illustration of a typical use.

* The "low level" group consists of the following functions :

generateMetadata, Spec1rProcpar, Spec1rDoProc, RWrapperCMD1D

This set of functions allows to process spectra one by one (Spec1rDoProc) and to apply the treatments one by one on the spectra set ('RWrapperCMD1D'). Being intended to be integrated into a complex workflow such as implemented in the NMRProcFlow application, these functions are accessible only to maintain compatibility between this package and the NMRProcFlow application, and are therefore very little documented.

Author(s)

Daniel Jacob

Maintainer: Daniel Jacob <djacob65@gmail.com>

References

Jacob, D., Deborde, C., Lefebvre, M., Maucourt, M. and Moing, A. (2017) NMRProcFlow: A graphical and interactive tool dedicated to 1D spectra processing for NMR-based metabolomics, Metabolomics 13:36. doi: 10.1007/s113060171178y

See Also

the NMRProcFlow online documentation https://nmrprocflow.org/

Examples

```
## Not run:
   ## Process the raw data
   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=detectCores())

   ## Plot the spectra for the given ppm range
   plotSpecMat(out$specMat, ppm_lim=c(0.5,9), K=0)
```

```
 ## Generate the output data matrix based on the computed buckets
 ## stored in  out$specMat$bucket_zones.
 outMat <- Rnmr1D::getBucketsDataset(out, norm_meth='CSN')

## Compute the bucket clustering based on a cut threshold  applied on the buckets tree
## resulting of a hierarchical classification analysis. If \code{vcutusr} is equal to
 ## zero then cut threshold automatically estimated; otherwise the specified value
 ## is applied.
 outclust <- Rnmr1D::getClusters(outMat, method='hca', vcutusr=0.15 )

 ## Plot the criterion curves and the cluster boxplot
 dev.new()
 plotCriterion(outclust)
 dev.new()
 plotClusters(outMat,outclust)

 ## ---- PCA  ----
 pca <- prcomp(outMat,retx=TRUE,scale=TRUE, rank=2)

 # plot PCA Scores
 dev.new()
plotScores(pca$x, 1, 2, out$samples, 'Treatment', level=0.95)  # Choose 'Treatment' as factor

 # plot PCA Loadings
 dev.new()
 plotLoadings(pca$rotation, 1, 2, associations=outclust$clustertab, cexlabel=0.6,
              main=sprintf("Loadings - Crit=


## End(Not run)
```

---

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)
```

---

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=detectCores())
# 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,
  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)
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=detectCores())
```

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=detectCores())
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=detectCores())
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)
```

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)

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

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

---

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

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=detectCores())
# 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)
```

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=6)
 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

`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 26.

Value

- DEBUG : Debug - defaut value = TRUE
- LOGFILE : Messages output file - default value = ""
- VENDOR : Instrumental origin of the raw data (bruker, varian) - default value = 'bruker'
- READ_RAW_ONLY : Read Raw file only; do not carry out processing; if 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
- REVTIME : Reverse time points - default value = FALSE
- BLPHC : Intensity offset correction - default value = FALSE
- SOLVPPM : ppm of the solvent (D20 by default) - default value = 4.8
- 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
- ALGO2 : alternative algorithm for phase optimization - default value = FALSE
- FRACPPM : Origin point for adjustment of the first order phase (defined as a fraction of the ppm scale) - default value = 0
- INCFRACPPM1 : Incrementation value for search of optimal point for adjustment of the first order phase : step 1 - default value = 0.125
- INCFRACPPM2 : Incrementation value for search of optimal point for adjustment of the first order phase : step 2 - default value = 0.0125
- JGD_INNER : JEOL : internal or external estimation for Group Delay - default value = TRUE

Index