Package ‘PepsNMR’

May 28, 2024

Type Package
Title Pre-process 1H-NMR FID signals
Version 1.22.0
Description This package provides R functions for common pre-processing steps that are applied on 1H-NMR data. It also provides a function to read the FID signals directly in the Bruker format.
License GPL-2 | file LICENSE
URL https://github.com/ManonMartin/PepsNMR
Imports Matrix, ptw, ggplot2, gridExtra, matrixStats, reshape2, methods, graphics, stats
Encoding UTF-8
NeedsCompilation no
Depends R (>= 3.6)
Suggests knitr, markdown, rmarkdown, BiocStyle, PepsNMRData
LazyData true
Note This package originates from a previous work of Eli Lilly together with Paul Eilers that have developed an automated Matlab library with innovating methods for 1H NMR pre-treatment that was called "Bubble". (J. Vanwinsberghe. Bubble: development of a matlab tool for automated 1H-NMR data processing in metabonomics. Master's thesis Strasbourg University, 2005.)
Contact Manon Martin <manon.martin@uclouvain.be>, Bernadette Govaerts <bernadette.govaerts@uclouvain.be> or Benoît Legat <benoit.legat@gmail.com>
VignetteBuilder knitr
biocViews Software, Preprocessing, Visualization, Metabolomics, DataImport
BugReports https://github.com/ManonMartin/PepsNMR/issues
Contents

PepsNMR-package .......................................................... 3
Apodization ............................................................... 4
BaselineCorrection ...................................................... 7
Bucketing ................................................................. 9
Draw ....................................................................... 10
DrawPCA ................................................................. 12
DrawSignal ............................................................ 13
FirstOrderPhaseCorrection ............................................ 14
FourierTransform ....................................................... 16
GroupDelayCorrection .................................................. 17
InternalReferencing .................................................... 18
NegativeValuesZeroing ................................................. 21
Normalization .......................................................... 22
PepsNMR-internal ....................................................... 24
PreprocessingChain ..................................................... 24
ReadFids ............................................................... 26
RegionRemoval ......................................................... 28
SolventSuppression .................................................... 29
Warping ................................................................. 31
WindowSelection ....................................................... 33
ZeroFilling ............................................................. 34
ZeroOrderPhaseCorrection .......................................... 35
ZoneAggregation ....................................................... 37

Index 38
Metabolomic data pre-processing strategy for 1H NMR spectroscopic data

Description

This package provides R functions for classic and advanced pre-processing steps that are applied on 1H NMR data. It also provides the function `ReadFids` to read the FID directly from the Bruker format. Those pre-processing are cited below in the advised order of their application:

- **GroupDelayCorrection** Corrrect for the first order phase correction.
- **SolventSuppression** Remove solvent signal from the FIDs.
- **Apodization** Increase the sensitivity/resolution of the FIDs.
- **ZeroFilling** Improve the visual representation of the spectra.
- **FourierTransform** Transform the FID into a spectrum and convert the frequency scale (Hertz -> ppm).
- **ZeroOrderPhaseCorrection** Correct for the zero order phase correction.
- **InternalReferencing** Calibrate the spectra with internal compound referencing.
- **BaselineCorrection** Remove the spectral baseline.
- **NegativeValuesZeroing** Set negatives values to 0.
- **Warping** Warp the samples according to a reference spectrum.
- **WindowSelection** Select the informative part of the spectrum.
- **Bucketing** Data reduction by integration.
- **RegionRemoval** Set intensities of a desired region to 0.
- **ZoneAggregation** Aggregate a region to a single peak.
- **Normalization** Normalize the spectra.

Details

- Package: PepsNMR
- Type: Package
- Version: 0.99.0
- License: GPLv2

The FIDs are read using `ReadFids` which also gives a matrix with meta-information about each FID. The other functions apply different pre-processing steps on these signals, and some need the info matrix as outputted from `ReadFids`. During this pre-processing, the signal is transformed through fourier transformation and the frequency scale is expressed in ppm. For more details and illustrated explanations about those pre-treatment steps, see the documentation of each function and/or the chapter 1 of the reference below.
Apodization

Author(s)

Benoît Legat, Bernadette Govaerts & Manon Martin

Maintainer: Manon Martin <manon.martin@uclouvain.be>

References


Examples

```r
path <- system.file("extdata", package = "PepsNMRData")
dir(path)

fidList <- ReadFids(file.path(path, "HumanSerum"))
Fid_data <- fidList[["Fid_data"]]
Fid_info <- fidList[["Fid_info"]]
Fid_data <- GroupDelayCorrection(Fid_data, Fid_info)
Fid_data <- SolventSuppression(Fid_data)
Fid_data <- Apodization(Fid_data, Fid_info)
Fid_data <- ZeroFilling(Fid_data)
Spectrum_data <- FourierTransform(Fid_data, Fid_info)
Spectrum_data <- ZeroOrderPhaseCorrection(Spectrum_data)
Spectrum_data <- InternalReferencing(Spectrum_data, Fid_info)
Spectrum_data <- BaselineCorrection(Spectrum_data)
Spectrum_data <- NegativeValuesZeroing(Spectrum_data)
Spectrum_data <- Warping(Spectrum_data)
Spectrum_data <- WindowSelection(Spectrum_data)
Spectrum_data <- Bucketing(Spectrum_data)
Spectrum_data <- RegionRemoval(Spectrum_data, typeofspectra = "serum")
Spectrum_data <- Normalization(Spectrum_data, type.norm = "mean")
```

---

### Apodization

#### Apodization of the FID

---

**Description**

The function multiplies the FID by a defined factor to increase the sensibility and/or resolution of the spectra.
Apodization

Usage

Apodization(Fid_data, Fid_info = NULL, DT = NULL, type.apod = c("exp","cos2","blockexp","blockcos2","gauss","hanning","hamming"), phase = 0, rectRatio = 1/2, gaussLB = 1, expLB = 0.3, plotWindow = FALSE, returnFactor = FALSE, verbose=FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fid_data</td>
<td>Matrix containing the FIDs, one row per signal, as outputted by <code>ReadFids</code>.</td>
</tr>
<tr>
<td>Fid_info</td>
<td>Matrix containing the info about the FIDs, one row per signal, as outputted by <code>ReadFids</code>.</td>
</tr>
<tr>
<td>DT</td>
<td>If given, used instead of Fid_info to give the Dwell Time, the time between 2 points of the FID.</td>
</tr>
<tr>
<td>type.apod</td>
<td>Type of apodization, see details.</td>
</tr>
<tr>
<td>phase</td>
<td>Phase at which the apodization window is maximum for cos2, hanning and hamming types. For example, if phase is 0.2, the maximum is at 20% of the signal.</td>
</tr>
<tr>
<td>rectRatio</td>
<td>If there is a rectangular window, ratio between the width of the window and the width of the signal.</td>
</tr>
<tr>
<td>gaussLB</td>
<td>Line Broadening for the gaussian window, see details.</td>
</tr>
<tr>
<td>expLB</td>
<td>Line Broadening for the exponential window, see details.</td>
</tr>
<tr>
<td>plotWindow</td>
<td>If TRUE, a plot of the signal applied to the FID is displayed.</td>
</tr>
<tr>
<td>returnFactor</td>
<td>If TRUE, returns a list with the final FIDs and the apodization function.</td>
</tr>
<tr>
<td>verbose</td>
<td>If &quot;TRUE&quot;, will print processing information.</td>
</tr>
</tbody>
</table>

Details

The apodization is usually performed in order to increase the sensitivity, i.e. the Signal-to-Noise Ratio (SNR) of the spectra. This is based on the fact that the signal intensity is decreasing over time unlike the noise that keeps a constant amplitude, leaving a noisy tail at the end of the FID. Multiplying the FID with a decaying signal will then increase the SNR. Since the area under the spectral peak remains unchanged, a faster decay will also result in a reduced peak height in spectra, lowering the spectral resolution. Optimal trade-off parameters for the apodization signal are thus needed to prevent high losses in sensitivity/resolution.

A FID of the form $s_0 \exp(i2\pi vt) \exp(-t/T)$ has a peak in its spectrum at the frequency $\nu$ of width that is inversely proportional to $T$. This peak is called a spectral line and its width a spectral width.

In the case of the exponential multiplication ("exp"), which is the default apodization, the decaying exponential becomes:

$$\exp(-t(1/T + LB))$$

The new decay $T^*$ which satisfies $1/T^* = 1/T + LB$ is therefore smaller so the spectral line is broader. That is why we call this parameter the Line Broadening.

If $LB$ increases, the SNR increases but at the expense of the spectral resolution. Usual values in proton NMR for "LB" found in the literature are 0.3 for the NOESY presat pulse sequence and
Apodization

-0.01 for the CMPG presat pulse sequence. It should not exceed the value of 1 to avoid information loss.

The different types of apodization are:

- **exp** The signal is multiplied by a decreasing exponential \( \exp(-t/\text{expLB}) \).
- **cos2** The signal is multiplied by the value of a \( \cos^2 \) from 0 (where its value is 1) until \( \pi/2 \) (where its value is 0).
- **blockexp** The first part of the signal (defined by \text{rectRatio}) is left unchanged and the second is multiplied by \( \exp(-t/\text{expLB}) \) starting at value 1.
- **blockcos2** The first part is left unchanged as with blockexp and the second part is multiplied by a \( \cos^2 \) where its value starts at 1 at the end of the block and ends at 0 at the end of the signal.
- **gauss** The signal is multiplied by a gaussian window centered at the beginning of the FID and with \( \sigma = 1/\text{gaussLB} \).
- **hanning** The signal is multiplied by a hanning window: \( 0.5 + 0.5 \cos \cos \).
- **hamming** The signal is multiplied by a hamming window: \( 0.54 + 0.46 \cos \cos \).

**Value**

If \text{returnFactor} is \text{TRUE}, will return a list with the following elements: \text{Fid_data} and \text{Factor}. Otherwise, the function will just return \text{Fid_data}.

- **Fid_data** The apodized FIDs.
- **Factor** The apodization signal.

**Author(s)**

Benoît Legat & Manon Martin

**References**

Inspired from the matNMR library.


**Examples**

```r
require(PepsNMRData)
Apod_res <- Apodization(Data_HS_sp$FidData_HS_2, FidInfo_HS, plotWindow=FALSE)

# or
Apod_res <- Apodization(Data_HS_sp$FidData_HS_2, FidInfo_HS, plotWindow=FALSE, returnFactor=TRUE)
Apod_fid = Apod_res[['Fid_data']] plot(Apod_res[['Factor']], type="l")
```
BaselineCorrection

Set the baseline to a uniform zero signal.

Description

The function estimates and removes the smoothed baseline from the spectra.

Usage

BaselineCorrection(Spectrum_data, ptw.bc = TRUE, maxIter = 42,
lambda.bc = 1e7, p.bc = 0.05, eps = 1e-8,
ppm.bc = TRUE, exclude.bc = list(c(5.1,4.5)),
returnBaseline = FALSE, verbose = FALSE)

Arguments

Spectrum_data Matrix containing the spectra, one row per spectrum.
ptw.bc If TRUE, calculates the baseline in C using the ptw library which is a lot faster. The R version is only kept because it is easier to understand than C and in case of problems with the installation of the ptw package.
maxIter Maximum number of iterations for the R version (if ptw.bc is set to FALSE).
lambda.bc Smoothing parameter (generally 1e5 – 1e8). See details.
p.bc Asymmetry parameter. See details.
eps Numerical precision for convergence when estimating the baseline.
ppm.bc If TRUE, the values in exclude.zopc represent frequencies in ppm value (column names of spectra), if FALSE these values are column indices.
exclude.bc If not NULL and ptw.bc == FALSE, a list containing the extremities of the intervals excluded for the baseline estimation, either expressed in ppm (decreasing values) OR in column indices (increasing values), e.g. exclude.bc = list(c(0,10000)) if ppm.bc == FALSE or exclude.bc = list(c(1,-1)) if ppm.bc == TRUE.
returnBaseline If TRUE, returns the estimated baselines.
verbose If "TRUE", will print processing information.

Details

The signal should be an addition of positive peaks which represent metabolites from the samples. These peaks are added to the baseline which is the signal representing the absence of any metabolite and should therefore be uniformly zero. For each spectrum, its baseline is thus estimated and removed. Let F be our initial spectrum and Z be its baseline. Once Z is approximated, the corrected spectrum is F − Z.

A negative signal doesn’t make sense and creates problems with the statistical analysis. The estimated baseline should then not be such that F − Z < 0. Hence, in the objective function to be minimized, the squared difference F − Z are weighted by p if F − Z > 0 or 1 − p if F − Z < 0. p is indeed
taken very small, e.g. 0.05, to avoid negative intensities. The function `NegativeValuesZeroing` is used thereafter to set the remaining negative intensities to zero after the baseline correction.

With this function to minimize, we would simply have \( F = Z \) as a solution which would make \( F - Z \) uniformly zero. Therefore, a roughness penalty term on \( Z \) is applied so that it does not match exactly the peaks. The importance of this smoothness constraint in the objective function is tuned by \( \lambda \) which is typically equal to 1e7.

**In summary**, useful parameters are:

- \( \text{p.bc} \): The default value is 0.05. The smaller it is, the less \( Z \) will try to follow peaks when it is under the function and the more it will try to be under the function.
- \( \lambda \text{mbda.bc} \): The default value is 1e7. The larger it is, the smoother \( Z \) will be. With \( \lambda \text{mbda} = 0 \), the baseline will be equal to the signal and the corrected signal will be zero.

The algorithm used to find the baseline is iterative. In ptw, the iteration is done until the baseline is found but if ptw.bc is set to FALSE, we stop after maxIter iterations.

More details and motivations are given in the articles mentioned in the References.

**Value**

If `returnBaseline` is TRUE, will return a list with the following elements: `Spectrum_data` and `Baseline`. Otherwise, the function will just return `Spectrum_data`.

- `Spectrum_data`: The matrix of spectra with the baseline removed.
- `Baseline`: Estimation of the baseline.

**Author(s)**

Benoît Legat, Manon Martin & Paul H. C. Eilers

**References**


**See Also**

See also `SolventSuppression` which also uses the Whittaker smoother.
Bucketing

Examples

```r
require(PepsNMRData)
BC_res <- BaselineCorrection(Data_HS_sp$Spectrum_data_HS_5,
                            lambda.bc=5e+06, p.bc=0.05)
#or
BC_res <- BaselineCorrection(Data_HS_sp$Spectrum_data_HS_5,
                            lambda.bc=5e+06, p.bc=0.05, returnBaseline=TRUE)
BC_spec = BC_res[["Spectrum_data"]]
plot(BC_res[["Baseline"]], type="l")
```

Bucketing  Spectral data reduction

Description

Reduces the number of data points by aggregating intensities into buckets.

Usage

```r
Bucketing(Spectrum_data, width = FALSE, mb = 500, boundary = NULL,
           intmeth = c("r", "t"), tolbuck = 10^-4, verbose = FALSE)
```

Arguments

- `Spectrum_data`: Matrix containing the spectra in ppm, one row per spectrum.
- `width`: If `width` is `TRUE`, then `m` represents the buckets' width, otherwise, it represents the number of buckets.
- `mb`: The number of buckets OR the buckets' width. If `mb` represents the number of buckets, it should be an integer smaller or equal to the number of frequencies in `Spectrum_data`.
- `boundary`: Numeric vector of left and right boundaries for ppm integration.
- `intmeth`: Type of bucketing: rectangular ("r") or trapezoidal ("t"). See details below.
- `tolbuck`: Tolerance threshold to check if the buckets of the original spectra are of constant length.
- `verbose`: If "TRUE", will print processing information.

Details

It is important to note that the input spectrum can have its ppm axis in increasing or decreasing order and it does not have to be equispaced.

Bucketing has two main interests:

- Ease the statistical analysis
• Decrease the impact of peaks misalignments between different spectra that should be aligned; assuming we are in the ideal case where they fall in the same bucket. Of course, the better the prior warping is, the larger \( m \) can be without major misalignment and the more informative the spectra will be.

The ppm interval of \( \text{Spectrum}_\text{data} \), let’s say \([a, b]\) where \( a > b \), is divided into \( mb \) buckets of size \((a - b)/mb\). The new ppm scale contains the \( m \) centers of these intervals. The spectral intensity at these centers is the integral of the initial spectral intensity on this bucket using either trapezoidal or rectangular integration.

**Value**

\( \text{Spectrum}_\text{data} \)  The matrix of spectra with their new ppm axis.

**Author(s)**

Benoît Legat, Bernadette Govaerts & Manon Martin

**References**


**Examples**

```r
require(PepsNMRData)
Bucket.spec <- Bucketing(Data_HS_sp$Spectrum_data_HS_10, mb = 500)
```

**Draw**

`Draw(Signal_data, type.draw = c("signal","pca"),
       output = c("default","window","png","pdf"),
       dirpath = ".",filename = "%03d", height = 480,
       width = 640, pdf.onefile = TRUE, ...)`

**Description**

Draws FIDs, spectra or their PCA scores/loadings.

**Usage**

`Draw(Signal_data, type.draw = c("signal","pca"),
       output = c("default","window","png","pdf"),
       dirpath = ".",filename = "%03d", height = 480,
       width = 640, pdf.onefile = TRUE, ...)`
Arguments

Signal_data Matrix containing the FIDs or spectra, one line per FID/spectrum.

type.draw Either "signal" or "pca", which calls respectively DrawSignal or DrawPCA to do the drawing.

output Specifies how to display the drawings:

default The output is the default one.
window Create a new window for each page.
png Create and save a new png image for each image.
pdf Create and save a new pdf image for each image.

dirpath The path to the directory where the png or pdf are outputted.

filename The filenames of the png and pdf, see argument filename in grDevices::png for more details.

height Height of the png and pdf in pixels.

width Width of the png and pdf in pixels.

pdf.onefile Wen output is set to "pdf" and there are multiples pages, if pdf.onefile is TRUE, all the pages are in the same file and if it is FALSE all the pages are in a different pdf file.

... The remaining arguments are passed either to DrawSignal or DrawPCA.

Details

Depending on the type.draw value, it can draw each row of Signal_data in a way described by subtype or the PCA scores or loadings (depending on the type.pca value) of all the FIDs/spectra in Signal_data.

Author(s)

Benoît Legat & Manon Martin

See Also

See Also DrawSignal and DrawPCA.

Examples

# Draw each signal Real part and Mod in separate png with name end001 end002, ...
# Draw the spectra
require(PepsNMRData)
Draw(FinalSpectra_HS, type.draw = "signal", output="window", subtype="together")

# Draw a PCA
Draw(FinalSpectra_HS, type.draw="pca",output="window")
DrawPCA

*Draw the PCA scores or loadings of the signals*

**Description**

The function draws the PCA scores or loadings of the FIDs/spectra given in the matrix `Signal_data`. Do not call this function directly but rather call `Draw` to specify how the plot will be returned.

**Usage**

```r
DrawPCA(Signal_data, drawNames = TRUE, main = "PCA score plot", Class = NULL,
         axes = c(1,2), type.pca = c("scores", "loadings"),
         loadingstype=c("l", "p"), num.stacked = 4, xlab = "rowname",
         createWindow)
```

**Arguments**

- `Signal_data` Matrix containing the FIDs or spectra, one line per FID/spectrum.
- `drawNames` If TRUE, the names of the spectra have to be shown alongside the points on the scores plot.
- `main` Plot title.
- `Class` Vector (numeric or character) indicating the class of each spectra. Used for scores plot only.
- `axes` Vector of score or loading numbers to be plotted. If it represents the score’s numbers, only the first two elements are used.
- `type.pca` The type of plot, either "scores" or "loadings".
- `loadingstype` The type of loadings plot, either a line plot ("l") or points with histogram-like vertical lines ("p").
- `num.stacked` Number of stacked plots for the loadings plots.
- `xlab` Label of the x-axis of loadings plots.
- `createWindow` If TRUE, will open a new window to display the graphs.

**Author(s)**

Benoît Legat & Manon Martin

**See Also**

See also `Draw` and `DrawSignal`.
Examples

```r
require(PepsNMRData)
# Draw loadings
DrawPCA(FinalSpectra_HS, main = "PCA loadings plot",
        Class = NULL, axes = c(1,3, 5), type = "loadings", loadingstype="l",
        num.stacked=4, xlab="ppm", createWindow = TRUE)

# Draw scores
class = substr(rownames(FinalSpectra_HS),5,5)
DrawPCA(FinalSpectra_HS, drawNames = TRUE, main = "PCA scores plot",
        Class = class, axes = c(1,2), type = "scores", createWindow = TRUE)
```

Description

Depending on the subtype, will draw the different parts of the complex FIDs/spectra.

Usage

```r
DrawSignal(Signal_data, subtype = c("stacked", "together", "separate", "diffmean", "diffmedian", "diffwith"),
            ReImModArg = c(TRUE, FALSE, FALSE, FALSE),
            vertical = TRUE , xlab = "index",
            RowNames = NULL, row = 1, num.stacked = 4, main = NULL,
            createWindow)
```

Arguments

- **Signal_data**: Matrix containing the FIDs or spectra, one line per FID/spectrum.
- **subtype**: Specifies the drawing array:
  - **together**: Plots all the signals in the same plot.
  - **separate**: Plots each signal on a different page.
  - **stacked**: Plots num.stacked signals on stacked plots with the same x-axis.
  - **diffmean**: Plots all the signals in the same plot but subtracted by their mean at each point.
  - **diffmedian**: Plots all the signals in the same plot but subtracted by their median at each point.
  - **diffwith**: Plots all the signals in the same plot but subtracted by the rowth signal at each point.
- **ReImModArg**: Specifies which of the real, imaginary, modulus, or argument part of the complex signal has to be plotted. Those plots are on the same page.
- **vertical**: Specifies whether the parts of the complex signal have to be put vertically or horizontally on the page if there are only 2 parts. If more, there will be 2 horizontally and 2 vertically anyway.
**FirstOrderPhaseCorrection**

**Perform a first order phase correction.**

**Description**

The function removes the group delay at the beginning of the FIDs.

**Usage**

FirstOrderPhaseCorrection(Fid_data, Fid_info = NULL, group_delay = NULL, verbose = FALSE)

---

**Details**

Don’t call this function directly but rather call `Draw` to specify how the plot will be outputted.

**Author(s)**

Benoît Legat & Manon Martin

**See Also**

See also `Draw` and `DrawPCA`.

**Examples**

```r
require(PepsNMRData)
plots <- DrawSignal(FinalSpectra_HS[1:4,], subtype = "together",
                   ReImModArg = c(TRUE, TRUE, FALSE, FALSE), createWindow = TRUE)
grid::grid.draw(plots)
```

---

**FirstOrderPhaseCorrection**

**FirstOrderPhaseCorrection**

**xlab**

Label of the x-axis.

**RowNames**

Strings to use instead of the rownames as labels for the plots if subtype = "separate". It should be a vector of the same length than the number of FIDs.

**row**

row to be compared to if the subtype is "diffwith".

**num.stacked**

Number of stacked plots if subtype is "stacked".

**main**

If not NULL, the main title when subtype is different from "separate".

**createWindow**

If TRUE, will open a new window.
FirstOrderPhaseCorrection

Arguments

- **Fid_data**: Matrix containing the FIDs, one row per signal, as outputted by `ReadFids`.
- **Fid_info**: Matrix containing the info about the FIDs, one row per signal, as outputted by `ReadFids`.
- **group_delay**: If given, it is used instead of `Fid_info` to decide how much the FIDs must be shifted to the left. It can be non-integer, in that case the values are interpolated. However it has to be non-negative since in our practical case, it would make no sense to add a part of the end of the FID at the beginning.
- **verbose**: If "TRUE", will print processing information.

Details

First Order Phase Correction step could also called "removal of Bruker digital filter".

Due to Bruker’s digital filter and to other technical reasons a first order phase shift caused by a group delay is present in the FID and needs to be removed. Luckily, information about this delay is available when loading the FID with `ReadFids` and is written in `Fid_info`.

This function shifts circularly each FID in order to cancel this delay. By circularly, we mean that the starting portion of the FID becomes its ending portion when applied.

Each FID is shifted by the same amount since it can be non-integer and the columns names which are the time coordinates are shared between all the FIDs.

Value

- **Fid_data**: The matrix of FIDs corrected for the first order phase shift.

Author(s)

Benoît Legat & Manon Martin

References


Examples

```r
require(PepsNMRData)
Fopc.fid <- FirstOrderPhaseCorrection(Data_HS_sp$FidData_HS_0,FidInfo_HS)
```
FourierTransform

**Applies the fourier transformation to the FIDs.**

**Description**

The function takes the FIDs in the time domain and translates it into the frequency domain. It also converts the frequency scale from hertz to part per million (ppm).

**Usage**

```r
FourierTransform(Fid_data, Fid_info = NULL, SW_h = NULL, SW = NULL,
                 O1 = NULL, reverse.axis = TRUE, verbose = FALSE)
```

**Arguments**

- **Fid_data**: Matrix containing the FIDs, one row per signal, as outputted by `ReadFids`.
- **Fid_info**: Matrix containing the info about the FIDs, one row per signal, as outputted by `ReadFids`.
- **SW_h**: Sweep Width in hertz. If given, the value in `Fid_info` is ignored.
- **SW**: Sweep width in ppm. If given, the value in `Fid_info` is ignored.
- **O1**: Spectrometer frequency offset. If given, the value in `Fid_info` is ignored.
- **reverse.axis**: If `TRUE`, the frequency scale is reversed.
- **verbose**: If "TRUE", will print processing information.

**Details**

The number of points \( m \) doesn’t change and the frequency interval is from \(-SW/2\) to \(SW/2 - SW/m\) (the \(-SW/m\) is due to the fact that we only have \( m \) points, not \( m + 1 \) and the fourier transform is periodic with period \( SW \) so it is the same at \(-SW/2\) and \(SW/2\) anyway).

\( SW, SW_h \) and \( O1 \) are usually taken from the `Fid_info` matrix. \( SW \) and \( SW_h \) are assumed to be the same for every FID since their column names are shared.

The frequency scale is dependent on the kind of spectrometer used, more precisely on its external magnetic field. We therefore translate it to a ppm (part per million) scale which is independent of this external magnetic field thanks to the recovered transmitter frequency offset value \( (O1) \).

**Value**

- **RawSpect_data**: The matrix of spectra in ppm.

**Author(s)**

Benoit Legat & Manon Martin
References


Examples

```r
require(PepsNMRData)
FT.spec <- FourierTransform(Data_HS_sp$FidData_HS_3,FidInfo_HS_sp, SW_h = 12019.23)
```

---

**GroupDelayCorrection**

Perform a first order phase correction.

**Description**

The function removes the group delay at the beginning of the FIDs.

**Usage**

```
GroupDelayCorrection(Fid_data, Fid_info = NULL, group_delay = NULL, verbose = FALSE)
```

**Arguments**

- `Fid_data`: Matrix containing the FIDs, one row per signal, as outputted by `ReadFids`.
- `Fid_info`: Matrix containing the info about the FIDs, one row per signal, as outputted by `ReadFids`.
- `group_delay`: If given, it is used instead of `Fid_info` to decide how much the FIDs must be shifted to the left. It can be non-integer, in that case the values are interpolated. However it has to be non-negative since in our practical case, it would make no sense to add a part of the end of the FID at the beginning.
- `verbose`: If "TRUE", will print processing information.

**Details**

First Order Phase Correction step could also called "removal of Bruker digital filter".

Due to Bruker’s digital filter and to other technical reasons a first order phase shift caused by a group delay is present in the FID and needs to be removed. Luckily, information about this delay is available when loading the FID with `ReadFids` and is written in `Fid_info`.

This function shifts circularly each FID in order to cancel this delay. By circularly, we mean that the starting portion of the FID becomes its ending portion when applied.

Each FID is shifted by the same amount since it can be non-integer and the columns names which are the time coordinates are shared between all the FIDs.
Value

Fid_data The matrix of FIDs corrected for the first order phase shift.

Author(s)

Benoît Legat & Manon Martin

References


Examples

```r
require(PepsNMRData)
Fopc.fid <- GroupDelayCorrection(Data_HS_sp$FidData_HS_0, FidInfo_HS)
```

Description

Chemical shifts are referenced against a Reference Compound (RC, e.g. TMSP).

Usage

```r
InternalReferencing(Spectrum_data, Fid_info, method = c("max", "thres"), range = c("nearvalue", "all", "window"), ppm.value = 0, direction = "left", shiftHandling = c("zerofilling", "cut", "NAfilling", "circular"), c = 2, pc = 0.02, fromto.RC = NULL, ppm.ir = TRUE, rowindex_graph = NULL, verbose = FALSE)
```

Arguments

- **Spectrum_data** Matrix containing the spectra in ppm, one row per spectrum.
- **Fid_info** Matrix containing the information for each spectrum, one row per spectrum, as returned by `ReadFids`.
- **method** Method used to find the RC peak in the spectra. See the details section.
- **range** How the search zone is defined. Either across the whole ppm axis ("all"), near the 0 ppm location (nearvalue) with parameter pc, or in a manually specified area of the ppm axis ("window") with the non-null parameter fromto.RC.
**InternalReferencing**

**ppm.value**  
By default, the ppm value of the reference compound is set to 0, but any arbitrary value in the ppm interval of spectra can be used instead.

**direction**  
If `method = "thres"`, the direction towards which to search for the RC peak.

**shiftHandling**  
See the details section.

**c**  
If `method = "thres"`, parameter used to fix the threshold for the RC peak.

**pc**  
If `range = "nearvalue"`, percentage of the ppm axis around the ppm.value ppm value to look for the RC peak (e.g. for `pc = 0.02`, intensities whose index values are 0.01% below and above 0 ppm are investigated).

**from.to.RC**  
If `range = "window"`, a list containing numerical vectors indicating the extremities of the intervals within which to search for the RC peak. These extremities are either frequencies in ppm (decreasing values) OR in column indices (increasing values) depending on the `ppm.ir` value (e.g. `from.to.RC = list(c(0,10000))` if `ppm.ir == FALSE` or `from.to.RC = list(c(1,-1))` if `ppm.ir == TRUE`).

**ppm.ir**  
If TRUE, the values in `from.to.RC` represent frequencies in ppm (column names of spectra), if FALSE these values are column indices.

**rowindex.graph**  
If not NULL, a numeric vector with the row numbers of spectra that need to be plotted for inspection.

**verbose**  
If "TRUE", will print processing information.

**Details**

Once the search zone is defined with `range`, the RC is found depending on the `method`. If `method = "thres"`, RC is the first peak in the spectrum higher than a predefined threshold which is computed as: `c*(cumulated_mean/cumulated_sd)`. If `method = "max"`, the maximum intensity in the search zone is defined as the RC.

Since the spectra can be shifted differently, we need to handle misalignment of the left and right of the spectrum.

This can be illustrated here:

| : TMSP peak |

**before**

```
  1  2  3 |  5  6  7  8  9
  1  2  3  4  5 |  7  8  9
  1  2  3  4 |  6  7  8  9
```

**shifted**

```
-5 -4 -3 -2 -1  0  1  2  3  4  5 : ppm scale
  1  2  3 |  5  6  7  8  9
  1  2  3  4  5 |  7  8  9
  1  2  3  4 |  6  7  8  9
```

The different **shift handleings** (shiftHandling) are the following:

**NAfilling**  
The extremities at which a spectrum is not defined are replaced by NA. It is detected by `WindowSelection` which produces a warning if there are NAs in the selected window.
zerofilling The extremities at which a spectrum is not defined are replaced by 0. It makes sense since in practice the spectrum is close to zero at the extremities.

```
-5 -4 -3 -2 -1 0 1 2 3 4 5 ppm scale
NA NA 1 2 3 | 5 6 7 8 9
1 2 3 4 5 | 7 8 9 NA NA
NA 1 2 3 4 | 6 7 8 9 NA
```

circular The spectra are shifted circularly which means that the end of a spectrum is reproduced at the beginning. It makes sense since the spectrum is periodic since it is the result of FFT.

```
-5 -4 -3 -2 -1 0 1 2 3 ppm scale
  0 0 1 2 3 | 5 6 7 8 9
  1 2 3 4 5 | 7 8 9 0 0
  0 1 2 3 4 | 6 7 8 9 0
```

cut The ppm values for which some spectra are not defined are removed.

```
-3 -2 -1 0 1 2 3 ppm scale
  1 2 3 4 5 | 7 8 9
  2 3 4 | 6 7 8
```

The difference between these shift handlings should not be critical in practice since the extremities of the spectra are not used most of the time and are removed in WindowSelection.

**Value**

if rowindex_graph is NULL:

Spectrum_data The matrix of the spectral value in the ppm scale.

if rowindex_graph is not NULL:

Spectrum_data The matrix of the spectral value in the ppm scale.

plots The spectra that need to be plotted for inspection.

**Author(s)**

Benoît Legat & Manon Martin

**References**


NegativeValuesZeroing

Examples

```
require(PepsNMRData)
PpmConv.spec <- InternalReferencing(Data_HS_sp$Spectrum_data_HS_5,
  FidInfo_HS, shiftHandling = "zerofilling")
```

Description

The function sets negative intensities to zero.

Usage

```
NegativeValuesZeroing(Spectrum_data, verbose = FALSE)
```

Arguments

- `Spectrum_data`: Matrix containing the spectra in ppm, one row per spectrum.
- `verbose`: If "TRUE", will print processing information.

Details

As explained in `BaselineCorrection`, negative values do not make sense and can have bad impacts on our statistical analyses. `BaselineCorrection` do its best to avoid negative intensity values but there might be some remaining.

This filter simply sets them to zero. After the `BaselineCorrection` they should be close to zero anyway because of the high penalty given to negative values of the signal after the correction.

Value

- `Spectrum_data`: The matrix of spectrums with the negative values set to zero.

Author(s)

Benoît Legat & Manon Martin

References

Examples

require(PepsNMRData)
Nvz.spec <- NegativeValuesZeroing(Data_HS_sp$Spectrum_data_HS_7)

---

Normalization

Normalizes the spectra

Description

Spectra normalization to correct for the dilution factor common to all biofuid samples.

Usage

Normalization(Spectrum_data, type.norm, fromto.norm = c(3.05, 4.05), ref.norm = "median", returnFactor = FALSE, verbose = FALSE)

Arguments

Spectrum_data Matrix containing the spectra in ppm, one row per spectrum.
type.norm Different types of normalization are available: "mean", "pqn", "median", "firstquartile" or "peak". No default value is provided. See the details section for more info.
fromto.norm Used if type.norm is "peak". See details.
ref.norm The reference spectrum if type.norm is "pqn". See details.
returnFactor If TRUE, returns a vector with the normalization factors.
verbose If "TRUE", will print processing information.

Details

Normalization of spectra before their warping or their statistical analysis is necessary in order to be able to efficiently compare their relative peak intensities.

It is therefore appropriate to call this filter at the end of the preprocessing workflow.

Normalization types can be:

- mean Each spectrum is divided by its mean so that its mean becomes 1.
- median Each spectrum is divided by its median so that its median becomes 1.
- firstquartile Each spectrum is divided by its first quartile so that its first quartile becomes 1.
- peak Each spectrum is divided by the value of the peak of the spectrum contained between "fromto.norm" inclusive (i.e. the maximum value of spectral intensities in that interval).
- pqn Probabilistic Quotient Normalization from Dieterle et al. (2006). If ref.norm is "median" or "mean", will use the median or the mean spectrum as the reference spectrum; if it is a single number, will use the spectrum located at that row in the spectral matrix; if ref.norm is a numeric vector of length equal to the number of spectral variables, it defines manually the reference spectrum.
The choice of a proper normalisation method is a crucial although not straightforward step in a metabolomic analysis.

Applying CSN is accurate in the following situations:

- when working on human/animal sera in the case of not serious pathology, given the homeostasis principle and since no dilution effect is present.
- When working on biopsies, the “metabolome quantity” is set constant across the samples by adding a varying volume of a buffer and the same applies when working with cell media, where the quantity of cells is made constant.

To counteract all the dilution effects and the excretion differences between urine samples, the PQN approach is often recommended in the literature (Dieterle et al., 2006).

For any other situation (large difference between the groups, other kind of sample, etc.), the choice of the normalisation method is not straightforward. A solution is to refer to endogenous stable metabolites that are present in a constant quantity across samples and use them as standards to normalize all spectral profiles. For the urine samples, the creatinine has been considered as such standard (this option is also implemented in PepsNMR), even though it has been shown that the creatinine concentration could fluctuate given specific parameters (Tang et al., 2015). A review on normalization techniques for mass spectroscopy metabolomics from Wu & Li (2015) provides some guidance in the choice on the normalization approach regarding the type of sample analysed and can be transposed to the NMR spectra normalisation.

**Value**

Spectrum_data The matrix of normalized spectra.

**Author(s)**

Benoît Legat & Manon Martin

**References**


Examples

```r
require(PepsNMRData)
Norm.spec <- Normalization(Data_HS_sp$Spectrum_data_HS_12,
  type.norm = "mean")
```

Description

Internal PepsNMR Functions

Details

These are not to be called by the user (or in some cases are just waiting for proper documentation to be written ...).

PreprocessingChain

Preprocessing workflow for 1H-NMR data

Description

The function is a wrapper for all the preprocessing steps available in PepsNMR.

Usage

```r
PreprocessingChain(Fid_data = NULL, Fid_info = NULL, data.path = NULL, readFids = TRUE,
  groupDelayCorr = TRUE, solventSuppression = TRUE, apodization = TRUE,
  zerofilling = TRUE, fourierTransform = TRUE, zeroOrderPhaseCorr = TRUE,
  internalReferencing = TRUE, baselineCorrection = TRUE, negativeValues0 = TRUE,
  warping = TRUE, windowSelection = TRUE, bucketing = TRUE, regionRemoval = TRUE,
  zoneAggregation = TRUE, normalization = TRUE, ..., export = FALSE,
  format = c("Rdata", "csv", "txt"), out.path = ".", filename = "filename",
  writeArg = c("none", "return", "txt"), verbose = FALSE)
```

Arguments

- **Fid_data** If non NULL, matrix containing the complex FIDs, one row per FID.
- **Fid_info** If non NULL, matrix containing the information for each spectrum, one row per spectrum, as returned by `ReadFids`.
- **data.path** A character string specifying the directory where the FIDs are searched.
- **readFids** If TRUE, applies the `ReadFids` function to the data.
- **groupDelayCorr** If TRUE, applies the `GroupDelayCorrection` function to the data.
solventSuppression
   If TRUE, applies the SolventSuppression function to the data.
apodization
   If TRUE, applies the Apodization function to the data.
zerofilling
   If TRUE, applies the ZeroFilling function to the data.
fourierTransform
   If TRUE, applies the FourierTransform function to the data.
zeroOrderPhaseCorr
   If TRUE, applies the ZeroOrderPhaseCorrection function to the data.
internalReferencing
   If TRUE, applies the InternalReferencing function to the data.
baselineCorrection
   If TRUE, applies the BaselineCorrection function to the data.
negativeValues0
   If TRUE, applies the NegativeValuesZeroing function to the data.
warping
   If TRUE, applies the Warping function to the data.
windowSelection
   If TRUE, applies the WindowSelection function to the data.
bucketing
   If TRUE, applies the Bucketing function to the data.
regionRemoval
   If TRUE, applies the RegionRemoval function to the data.
zoneAggregation
   If TRUE, applies the ZoneAggregation function to the data.
normalization
   If TRUE, applies the Normalization function to the data.
...
Other optionnal arguments of the above pre-processing functions.
export
   If TRUE, will export the spectral intensities and the acquisition parameters matrices.
format
   Format chosen to export the spectral intensities and the acquisition parameters matrices.
out.path
   Path used to export the spectral intensities and the acquisition parameters matrices if export == TRUE and the function argument if writeArg == "txt".
filename
   Name given to exported files.
writeArg
   If not "none", will export the function arguments, either in the return of the function ("return") or as a text file ("txt").
verbose
   If "TRUE", will print processing information.

Value

The function will return a list with the spectral intensities and the acquisition parameters matrices. If writeArg == "return", an additional list element is returned (arguments).

Spectrum_data
   The pre-processed spectra.
Fid_info
   The acquisition parameters.
arguments
   The function arguments.
ReadFids

Author(s)

Manon Martin

References


Examples

```r
path <- system.file("extdata", package = "PepsNMRData")
data.path <- file.path(path, "HumanSerum")
res <- PreprocessingChain(Fid_data = NULL, Fid_info = NULL, data.path = data.path,
                          ReadFids = TRUE, type.norm = "mean", export = FALSE, writeArg = "return")
```

---

ReadFids

**Read FIDs in Bruker format from a directory**

Description

Finds all directories of `path` which contain a valid FID (i.e. contain the files `fid`, `acqu` and `acqus`) and loads them in a matrix.

Usage

`ReadFids(path, l = 1, subdirs = FALSE, dirs.names = FALSE, verbose = FALSE)`

Arguments

- `path`: A character string specifying the directory where the FIDs are searched.
- `l`: A positive number indicating which line of the title file to use as spectra names.
- `subdirs`: If TRUE, will search inside subdirectories for FIDs and will merge them to have unique FID and info matrices.
- `dirs.names`: If TRUE, the FID names are recovered from the (sub)directories names, provided one subdirectory corresponds to one FID.
- `verbose`: If "TRUE", will print processing information.
Details

The row names are the first line of the file "pData/1/title" in the directory or the directory name (and subdirectory if subdirs == TRUE) if the title file doesn’t exist or the line 1 is blank. The column names are the time coordinates of the FID. All the FIDs therefore need to have the same length and time interval between points.

Case 1: subdirs = FALSE
DIR1 => 1, 2, 3, ...

Case 2a: subdirs = TRUE
DIR1 => 1 ; DIR2 => 1 ; DIR3 => 1 ; ...

Case 2b: subdirs = TRUE
DIR1 => 1, 2, ... ; DIR2 => 1, 2, ... ; ...

Value

Returns a list with the FIDs and their related information.

Fid_data The matrix containing the FIDs.
Fid_info A matrix containing the information about the FIDs. The naming of the row is the same than for Fid_data.

The columns are:

TD Time domain size
BYTORDA Determine the endiness of stored data. If 0 -> Little Endian; if 1 -> Big Endian
DIGMOD Digitization mode
DECIM Decimation rate of digital filter
DSPFVS DSP firmware version
SW_h Sweep width in Hz
SW Sweep width in ppm
O1 Spectrometer frequency offset
GPRDLY Group Delay
DT Dwell time in microseconds

Author(s)
Benoît Legat & Manon Martin

Examples

path <- system.file("extdata", package = "PepsNMRData")
dir(path)

fidList_HS <- ReadFids(file.path(path, "HumanSerum"))
FidData_HS_0 <- fidList_HS["Fid_data"]
FidInfo_HS <- fidList_HS["Fid_info"]
RegionRemoval

**Removal of non-informative regions**

### Description

Removes the non-informative regions by setting the values of the spectra in these intervals to zero.

### Usage

```r
RegionRemoval(Spectrum_data, typeofspectra = c("manual", "serum", "urine"),
              type.rr = c("zero", "NA"),
              fromto.rr = list(Water = c(4.5, 5.1)), verbose = FALSE)
```

### Arguments

- **Spectrum_data**: Matrix containing the spectra in ppm, one row per spectrum.
- **typeofspectra**: Type of spectra, if not "manual", will automatically remove unwanted regions depending on the nature of spectra.
- **type.rr**: Type of region removal method. If type.rr = "zero", intensities are set to 0; if type.rr = "NA", intensities are set to NA.
- **fromto.rr**: List containing the extremities of the intervals to be removed.
- **verbose**: If "TRUE", will print processing information.

### Details

The presence of non-informative regions can strongly bias the subsequent statistical analysis.

The inclusive ppm interval fromto.rr is set to zero or completed with NAs for every spectrum. The ppm scale can be increasing or decreasing (i.e. from < to or from > to).

The type of spectra can be NULL to manually specify the area to be removed otherwise it is specified as typeofspectra = "serum" or typeofspectra = "urine" and the removed area are for typeofspectra = "serum": water (4.5 - 5.1 ppm) and for typeofspectra = "urine": water, uree and maleic acid (4.5 - 6.1 ppm).

### Value

- **Spectrum_data**: The matrix of spectra with the removed regions.

### Author(s)

Benoît Legat & Manon Martin
**SolventSuppression**

**References**


**Examples**

```r
# Remove the lactate and water regions for serum spectra
require(PepsNMRData)
fromto <- list(Water =c(4.5, 5.1), Lactate=c(1.32, 1.36))
Rr.spec <- RegionRemoval(Data_HS_sp$Spectrum_data_HS_11, fromto.rr = fromto)
```

---

**SolventSuppression**  
*Suppress the Solvent signal present in each FID.*

**Description**

Signal smoothing for water residuals resonance removal.

**Usage**

```r
SolventSuppression(Fid_data, lambda.ss = 1e6, ptw.ss = TRUE, returnSolvent = FALSE, verbose = FALSE)
```

**Arguments**

- **Fid_data**: Matrix containing the FIDs, one row per signal, as outputted by `ReadFids`.
- **lambda.ss**: Penalty on roughness used to calculate the smoothed version of the FID. The higher lambda is, the smoother the estimated solvent signal will be.
- **ptw.ss**: If TRUE, calculates the solvent signal in C using the ptw package which is a lot faster. The R version is only kept in case of problems with the installation of ptw.
- **returnSolvent**: If TRUE, returns a list with the resulting FIDs, the real and imaginary parts of the estimated solvent signal, see the examples.
- **verbose**: If "TRUE", will print processing information.
Details

FIDs usually present a wavy shape. Under the assumption that water is the main compound of the analyzed samples, its signal can be modelled by the smoothing of the FIDs. We then subtract this wave, i.e. the solvent residuals resonance signal, from the original FIDs.

The smoothing is done with a Whittaker smoother which is obtained by the minimization of

$$V + \lambda R$$

where

- $V$ is the sum of the squared differences between the original and the smoothed signal.
- $R$ measures the roughness of the estimated signal.

The larger $\lambda$ is, the smoother the solvent residuals resonance signal. Eilers (2003) and Frasso & Eilers (2015) suggest different ways to tune $\lambda$ in order to optimise the smoothing: either visually, by cross-validation or using the V-curve procedure.

Value

If `returnSolvent = TRUE`, will return a list with the following elements: `Fid_data`, `SolventRe` and `SolventIm`. Otherwise, the function will just return `Fid_data`.

- `Fid_data`: The matrix of FIDs with the solvent residuals signal removed.
- `SolventRe`: The real part of the solvent signal.
- `SolventIm`: The imaginary part of the solvent signal.

Author(s)

Benoît Legat, Manon Martin & Paul H. C. Eilers

References


See Also

See also `BaselineCorrection` which also uses the Whittaker smoother.
Examples

```r
require(PepsNMRData)
Ss.fid <- SolventSuppression(Data_HS_sp$FidData_HS_1, returnSolvent=FALSE)

# or
Ss.res <- SolventSuppression(Data_HS_sp$FidData_HS_1, returnSolvent=TRUE)
Ss.fid = Ss.res[["Fid_data"]]
SolventRe = Ss.res[["SolventRe"]]
plot(SolventRe[1,], type="l")
```

---

**Warping**

**Warping of the spectra**

Description

Warping the frequency x-axis to minimize the pairwise distance between a sample spectrum and a reference spectrum.

Usage

```r
Warping(Spectrum_data, normalization.type = c("median","mean",
"firstquartile","peak","none"), fromto.normW = c(3.05, 4.05),
reference.choice = c("fixed", "before",
"after", "manual"), reference = 1, optim.crit = c("RMS", "WCC"),
ptw.wp = FALSE, K = 3, L = 40, lambda.smooth = 0, deg = 3,
lambda.bspline = 0.01, kappa = 0.0001, max_it_Bspline = 10,
returnReference = FALSE, returnWarpFunc = FALSE, verbose = FALSE)
```

Arguments

- **Spectrum_data** Matrix containing the spectra in ppm, one row per spectrum.
- **normalization.type** Type of normalization applied to the spectra prior to warping. See Normalization for details about the different types. none means that no normalization is applied. It is advised to use median instead of the default mean normalization.
- **fromto.normW** Used by Normalization when normalization.type is peak.
- **reference.choice** Specifies how the reference will be chosen:
  - "fixed" The reference is specified by the rowname given in reference.
  - "before" The reference is taken as the spectrum with the minimum sum of square difference with the other spectra.
  - "after" Each spectrum is taken as the reference and the sum of square difference with the other spectra is calculated after the warping. See details below.
  - "manual" The reference spectrum is specified in the reference argument.
The row number or name of the reference spectrum when `reference.choice` is "fixed" or a numeric vector with the reference spectrum when `reference.choice` is "manual".

If `ptw.wp` is set to `TRUE`, WCC can also be considered as a criterion for optimization, see `ptw::ptw` for details.

If set to `TRUE`, it applies the Parametric Time Warping with the `ptw::ptw` function to the data. In this case, the warping does not use B-splines and the arguments `L`, `deg`, `lambda.bspline`, `kappa` and `max_it_Bspline` are ignored.

It is the degree of the polynomial used for the warping (see details).

This is the number of B-splines that are used for the warping. It should be either 0 or greater than `deg`.

Nonnegative coefficient for the smoothing `lambda.smooth = 0` means no smoothing. See `ptw::difsm` for more details.

Degree of the B-splines.

Nonnegative second-order smoothness penalty coefficient for the B-splines warping. See the reference for more details.

Nonnegative ridge (zero-order) penalty coefficient for the B-splines warping. See the reference for more details.

Maximum number of iterations for the B-splines warping.

If `TRUE`, will return the name of the reference spectrum.

If `TRUE`, will return the warping functions.

If "TRUE", will print processing information.

When `reference.choice` is "after", the reference with the minimum sum is taken as the reference and the warped spectra according to this reference (that have already been calculated at this stage) are returned. This is `n` times slower than the 2 others where `n` is the number of spectra.

**Principle:**

We try to find a warping function between a reference spectrum and a sample. This function is a sum of polynomial of degree `K` and `L` B-splines of degree `deg`. The unknowns are the polynomial and B-splines coefficients.

No warping is equivalent to warping with `a`, the polynomial identity and all the coefficients of the B-splines with value 0. See the reference for details.

First, the polynomial is estimated on the reference and the sample both smoothed with parameter `lambda.smooth`. The B-splines are estimated on the non-smoothed reference and sample using the polynomial just found.

The higher `lambda.bspline` and `kappa` are, the less flexible the warping function will be.
Value

If returnReference = TRUE, the function will return the name of the reference spectrum and if returnWarpingfunc = TRUE, it will also return the warping functions.

Spectrum_data The warped spectra.
Reference The name of the reference spectrum.
Warpingfunc The warping functions.

Author(s)

Benoît Legat, Manon Martin & Paul H. C. Eilers

References


Examples

```r
require(PepsNMRData)
Warp.spec <- Warping(Data_HS_sp$Spectrum_data_HS_8, reference.choice="fixed",
                     reference = row.names(Data_HS_sp$Spectrum_data_HS_8)[1],
                     returnReference = FALSE)

# or
Warp.res <- Warping(Data_HS_sp$Spectrum_data_HS_8, reference.choice="fixed",
                     reference = row.names(Data_HS_sp$Spectrum_data_HS_8)[1],
                     returnReference = TRUE)

Warp.spec <- Warp.res[['Spectrum_data']]
Warp.res[['Reference']]
```

Description

Selects an interval in the ppm scale and returns the value of the spectra in that interval.

Usage

```r
WindowSelection(Spectrum_data, from.ws = 10, to.ws = 0.2, verbose = FALSE)
```
ZeroFilling

Arguments

Spectrum_data   Matrix containing the spectra in ppm, one row per spectrum.
from.ws         The left ppm value of the interval. A typical value is 10. If NULL, default value is the first index without NA.
to.ws           The right ppm value of the interval. A typical value is 0.2. If NULL, default value is the last index without NA.
verbose         If "TRUE", will print processing information.

Details

If from.ws and/or to.ws are not specified we calculate it so that we have the largest window without NA. Those NAs are typically produced by the InternalReferencing function.

Value

Spectrum_data   The matrix of the value of the spectra in the specified interval.

Author(s)

Benoit Legat & Manon Martin

Examples

require(PepsNMRData)
# The interval is chosen so that we have the largest interval without NA
Ws.spec <- WindowSelection(Data_HS_sp$Spectrum_data_HS_9)

# or
Ws.spec <- WindowSelection(Data_HS_sp$Spectrum_data_HS_9, from.ws=10, to.ws=0.2)

ZeroFilling

Zero Filling

Description

The function applies zero filling to the FIDs.

Usage

ZeroFilling(Fid_data, fn = ncol(Fid_data), verbose = FALSE)

Arguments

Fid_data        Matrix containing the FIDs, one row per signal, as outputted by ReadFids.
fn              Number of 0 to be added.
verbose         If "TRUE", will print processing information.
ZeroOrderPhaseCorrection

Details

Zero filling does not improve the spectral resolution but lead to better visually defined lines in the spectra. During zero filling, fn zeros are appended at the end of the FIDs. This number is rounded to the nearest 2^x value to ease the upcoming Fourier Transform of the FIDs.

Value

Fid_data The zero-filled FIDs.

Author(s)

Manon Martin

Examples

require(PepsNMRData)
ZF_fid <- ZeroFilling(Data_HS_sp$FidData_HS_3, fn = ncol(Data_HS_sp$FidData_HS_3))

ZeroOrderPhaseCorrection

Zero Order Phase Correction

Description

The function corrects the spectra in order to have their real part in an absorptive mode.

Usage

ZeroOrderPhaseCorrection(Spectrum_data, type.zopc = c("rms", "manual", "max"), plot_rms = NULL, returnAngle = FALSE, createWindow = TRUE, angle = NULL, plot_spectra = FALSE, ppm.zopc = TRUE, exclude.zopc = list(c(5.1,4.5)), verbose = FALSE)

Arguments

Spectrum_data Matrix containing the spectra in ppm, one row per spectrum.
type.zopc Method used to select the angles to rotate the spectra. See details.
plot_rms Contains a vector of row names for which a debug plot should be made showing the value of the function we try to minimize as a function of the phase.
returnAngle If TRUE, will return the rotation angle used for phase correction.
createWindow If TRUE, will open a new window to draw the rms or spectra plots, if FALSE, plots are drawn in the current device.
angle If not NULL, a numeric vector with angles specified in radian to manually rotate the spectra, one angle per spectrum. By convention, the spectra are rotated with the correction angle + angle.
plot_spectra If TRUE, will draw real and imaginary parts of the rotated spectra.
ZeroOrderPhaseCorrection

ppm.zopc If TRUE, the values in exclude.zopc represent frequencies in ppm value (column names of spectra), if FALSE these values are column indices.

exclude.zopc If not NULL, a list containing the extremities of the intervals excluded for the computation of the positiveness criterion, either expressed in ppm (decreasing values) OR in column indices (increasing values), e.g. exclude.zopc = list(c(0,10000)) if ppm.zopc == FALSE or exclude.zopc = list(c(1,-1)) if ppm.zopc == TRUE.

verbose If "TRUE", will print processing information.

Details

We focus our optimization on the positiveness of the real part which should be in an absorbptive mode.

When type.zopc is "rms", a positiveness criterion is measured for each spectrum. "manual" is used when a vector of angles are specified in angle and "max" will optimize the maximum spectral intensity in the non-excluded window(s). Beware that if exclude.zopc is not NULL, the optimization will only consider the non-excluded spectral window(s).

By default the water region (5.1 - 4.5) is ignored.

BaselineCorrection and NegativeValuesZeroing will take care of the last negative values of the real part of the spectra. See the reference for more details.

Value

Spectrum_data The matrix of rotated spectra.

Author(s)

Benoît Legat & Manon Martin

References


Examples

```{r}
require(PepsNMRData)
Zopc.res <- ZeroOrderPhaseCorrection(Data_HS_sp$Spectrum_data_HS_4,
ppm.zopc = FALSE, exclude.zopc = list(c(5000,15000)))
```
ZoneAggregation

---

**ZoneAggregation**

*Aggregates the values in a given ppm interval.*

---

**Description**

The function replaces the values given in specified intervals by triangular shaped peaks with the same area than the original peaks.

**Usage**

`ZoneAggregation(Spectrum_data, fromto.za = list(Citrate = c(2.5, 2.7)), verbose = FALSE)`

**Arguments**

- `Spectrum_data`: Matrix containing the spectra in ppm, one row per spectrum.
- `fromto.za`: List containing the borders in ppm of the intervals to aggregate.
- `verbose`: If "TRUE", will print processing information.

**Details**

The interval is specified in the unit of the column names (which should be ppm). This aggregation is usually performed with urine samples that contain citrate.

**Value**

- `Spectrum_data`: The matrix of spectra with their zone aggregated.

**Author(s)**

Benoit Legat & Manon Martin

**Examples**

```
require(PepsNMRData)
Spectrum_data <- ZoneAggregation(Data_HU_sp$Spectrum_data_HU_12,
fromto.za = list(Citrate = c(2.5, 2.7)))
```
Index

* **IO**
  * ReadFids, 26

* **array**
  * Bucketing, 9

* **hplot**
  * Draw, 10
  * DrawPCA, 11, 12, 14
  * DrawSignal, 11, 12, 13

* **internal**
  * PepsNMR-internal, 24

* **manip**
  * Apodization, 4
  * BaselineCorrection, 7
  * Bucketing, 9
  * FirstOrderPhaseCorrection, 14
  * FourierTransform, 16
  * GroupDelayCorrection, 17
  * InternalReferencing, 18
  * NegativeValuesZeroing, 21
  * Normalization, 22
  * PreprocessingChain, 24
  * RegionRemoval, 28
  * SolventSuppression, 29
  * Warping, 31
  * WindowSelection, 33
  * ZeroFilling, 34
  * ZeroOrderPhaseCorrection, 35
  * ZoneAggregation, 37

* **package**
  * PepsNMR-package, 3

  Apodization, 3, 4, 25

  BaselineCorrection, 3, 7, 21, 25, 30, 36

  beginTreatment (PepsNMR-internal), 24

  binarySearch (PepsNMR-internal), 24

  B spline (PepsNMR-internal), 24

  B splineWarp (PepsNMR-internal), 24

  Bucketing, 3, 9, 25

  checkArg (PepsNMR-internal), 24

  Draw, 10, 12, 14

  DrawPCA, 11, 12, 14

  DrawSignal, 11, 12, 13

  endTreatment (PepsNMR-internal), 24

  fftshift1D2D (PepsNMR-internal), 24

  FirstOrderPhaseCorrection, 14

  FourierTransform, 3, 16, 25

  getArg (PepsNMR-internal), 24

  getDirsContainingFid (PepsNMR-internal), 24

  getTitle (PepsNMR-internal), 24

  GroupDelayCorrection, 3, 17, 24

  indexInterval (PepsNMR-internal), 24

  InternalReferencing, 3, 18, 25

  Interpol (PepsNMR-internal), 24

  NegativeValuesZeroing, 3, 8, 21, 25, 36

  Normalization, 3, 22, 25, 31

  PepsNMR (PepsNMR-package), 3

  PepsNMR-internal, 24

  PepsNMR-package, 3

  PolyWarp (PepsNMR-internal), 24

  PreprocessingChain, 24

  PrincipalComponentAnalysis (PepsNMR-internal), 24

  ReadFid (PepsNMR-internal), 24

  ReadFids, 3, 5, 15–18, 24, 26, 29, 34

  readParams (PepsNMR-internal), 24

  RegionRemoval, 3, 25, 28

  SingleWarp (PepsNMR-internal), 24

  SolventSuppression, 3, 8, 25, 29

  vec2mat (PepsNMR-internal), 24
INDEX

Warping, 3, 25, 31
WindowSelection, 3, 19, 20, 25, 33

ZeroFilling, 3, 25, 34
ZeroOrderPhaseCorrection, 3, 25, 35
ZoneAggregation, 3, 25, 37