Package 'ChemoSpec2D'

June 9, 2019

Type Package

Title Exploratory Chemometrics for 2D Spectroscopy

Version 0.3.166

Date 2019-06-09

Description A collection of functions for exploratory chemometrics of 2D spectroscopic data sets such as COSY (correlated spectroscopy) and HSQC (heteronuclear single quantum coherence) 2D NMR (nuclear magnetic resonance) spectra. 'ChemoSpec2D' deploys methods aimed primarily at classification of samples and the identification of spectral features which are important in distinguishing samples from each other. Each 2D spectrum (a matrix) is treated as the unit of observation, and thus the physical sample in the spectrometer corresponds to the sample from a statistical perspective. In addition to chemometric tools, a few tools are provided for plotting 2D spectra, but these are not intended to replace the functionality typically available on the spectrometer. 'ChemoSpec2D' takes many of its cues from 'ChemoSpec' and tries to create consistent graphical output and to be very user friendly.

License GPL-3

Depends R (>= 3.5), ChemoSpecUtils (>= 0.3)

Imports tools, utils

Suggests knitr, tinytest, irlba, ThreeWay, multiway, parallel, matrixStats, R.utils, mlrMBO, ParamHelpers, smoof, mlr, lhs, RcppRoll, rmarkdown, pinp, robustbase

URL https://github.com/bryanhanson/ChemoSpec2D

BugReports https://github.com/bryanhanson/ChemoSpec2D/issues

ByteCompile TRUE

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Bryan A. Hanson [aut, cre] (<https://orcid.org/0000-0003-3536-8246>)

Maintainer Bryan A. Hanson <hanson@depauw.edu>

Repository CRAN

Date/Publication 2019-06-09 17:50:02 UTC

R topics documented:

`ChemoSpec2D-package` *Exploratory Chemometrics for 2D Spectroscopy*

Description

Description: A collection of functions for exploratory chemometrics of 2D spectroscopic data sets such as COSY and HSQC NMR spectra. ChemoSpec2D deploys methods aimed primarily at classification of samples and the identification of spectral features which are important in distinguishing samples from each other. Each 2D spectrum (a matrix) is treated as the unit of observation, and thus the physical sample in the spectrometer corresponds to the sample from a statistical perspective. In addition to chemometric tools, a few tools are provided for plotting 2D spectra, but these are not intended to replace the functionality typically available on the spectrometer. ChemoSpec2D takes many of its cues from ChemoSpec and tries to create consistent graphical output and to be very user friendly.

Author(s)

Bryan A. Hanson.

Maintainer: Bryan A. Hanson <hanson@depauw.edu>

`calcLvls` *Calculate Levels for Contour and Image Type Plots*

Description

Given a matrix, this function will assist in selecting levels for preparing contour and image type plots. For instance, levels can be spaced evenly, logrithmically, exponentially or using a cumulative distribution function. `NA` values are ignored.

Usage

```
calcLvls(M, n = 10, mode = "even", lambda = 1, base = 2,
  showHist = FALSE, ...)
```

Arguments

`M`	A numeric matrix.
`n`	For all methods except `ecdf`, an integer giving the number of levels desired. For most modes this is used interally as `floor(n/2)` and the result doubled. In addition, only the positive or negative levels may be selected, so you are not likely to actually get `n` levels most of the time (remember, you can always give your desired levels as a vector). For `ecdf`, `n` should be one or more values in the interval (0...1). For instance, a value of 0.6 corresponds to a single level in which 60 percent of the matrix values are below, and 40 percent above.
`mode`	Character. One of `"even"`, `"log"`, `"exp"`, `"ecdf"`, `"posexp"`, `"negexp"`, `"poslog"`, `"neglog"` or `NMR`. `"even"` will create evenly spaced levels. `"log"` will create levels which are more closely spaced at the high values, while `"exp"` does the opposite. The pos- or neg- versions select just the positive or negative values. `"ecdf"` computes levels at the requested quantiles of the matrix. `NMR` uses `log` but removes the levels closest to zero, which in NMR work will typically be too low for a good contour plot.
`lambda`	Numeric. A non-zero exponent used with `method = "exp"` and relatives.
`base`	Integer. The base used with `method = "log"` and relatives.
`showHist`	Logical. Shall a histogram be drawn showing the location of the chosen levels?
`...`	Arguments to be passed downstream.

Value

A numeric vector giving the levels.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

Examples

```
set.seed(9)
MM <- matrix(runif(100, -1, 1), nrow = 10) # test data
tsts <- c("even", "log", "poslog", "exp", "posexp", "ecdf", "NMR")
for (i in 1:length(tsts)) {
nl <- 10
if(tsts[i] == "ecdf")  nl <- seq(0.1, 0.9, 0.1)
levels <- calcLvls(M = MM, n = nl, mode = tsts[i],
   showHist = TRUE, main = tsts[i])
}
```

`centscaleSpectra2D` *Optionally Center and Optionally Scale a Spectra2D Object Along the Samples Dimension*

Description

This function will optionally center, and optionally scale, a `Spectra2D` object along the samples dimension (i.e. this is pixel-wise scaling in the language of multivariate image analysis). Several scaling options are available.

Usage

```
centscaleSpectra2D(spectra, center = FALSE, scale = "noscale")
```

Arguments

`spectra`	An object of S3 class `Spectra2D`.
`center`	Logical. Should the spectra be centered before possibly scaling? Will give an error if `center = TRUE` and a log function is requested for scaling.
`scale`	A character string indicating the type of scaling to apply. One of `c("autoscale", "Pareto", "log", "l` For the log functions, centering is not carried out since logarithm is not defined for negative values.

Value

An object of S3 class `Spectra2D`.

Author(s)

Bryan A. Hanson, DePauw University.

References

R. Bro and A. K. Smilde "Centering and Scaling in Component Analysis" J. Chemometrics vol. 17 pgs 16-33 (2003).

See Also

link{normSpectra2D} for another means of scaling.

Examples

```
data(MUD1)
tst <- centscaleSpectra2D(MUD1)
```

`check4Gaps` *Check for Discontinuities (Gaps) in a Vector & Optionally Make a Plot*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via check4Gaps.

`chkSpectra` *Verify the Integrity of a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via chkSpectra.

`colorSymbol` *Color and Symbols in ChemoSpec and ChemoSpec2D*

Description

You can access full documentation via colorSymbol.

`files2Spectra2DObject` *Import Data into a Spectra2D Object*

Description

This function imports data into a `Spectra2D` object. It uses `read.table` to read files so it is very flexible in regard to file formatting. **Be sure to see the . . . argument below for important details you need to provide.**

Usage

```
files2Spectra2DObject(gr.crit = NULL, gr.cols = "auto", fmt = NULL,
  nF2 = NULL, x.unit = "no frequency unit provided",
  y.unit = "no frequency unit provided",
  z.unit = "no intensity unit provided",
  descrip = "no description provided", fileExt = "\\.(csv|CSV)$",
  out.file = "mydata", debug = FALSE, ...)
```

Arguments

`gr.crit`	Group Criteria. A vector of character strings which will be searched for among the file/sample names in order to assign an individual spectrum to group membership. This is done using grep, so characters like "." (period/dot) do not have their literal meaning (see below). Warnings are issued if there are file/sample names that don't match entries in `gr.crit` or there are entries in `gr.crit` that don't match any file names.
`gr.cols`	A character vector, giving one color per group.
`fmt`	A character string giving the format of the data. Consult `import2Dspectra` for options.
`nF2`	Integer giving the number of data points in the F2 (x) dimension.
`x.unit`	A character string giving the units for the F2 dimension (frequency or wavelength corresponding to the x dimension).
`y.unit`	A character string giving the units for the F1 dimension (frequency or wavelength corresponding to the y dimension).
`z.unit`	A character string giving the units of the z-axis (some sort of intensity).
`descrip`	A character string describing the data set.
`fileExt`	A character string giving the extension of the files to be processed. `regex` strings can be used. For instance, the default finds files with either `".csv"` or `".CSV"` as the extension. Matching is done via a grep process, which is greedy.
`out.file`	A file name. The completed object of S3 class `Spectra2D` will be written to this file.
`debug`	Logical. Set to `TRUE` for troubleshooting when an error is thrown during import.
`...`	Arguments to be passed to `read.table`. **You MUST supply values for** `sep`**,** `dec` **and** `header` **consistent with your file structure, unless they are the same as the defaults for** `read.table`.

Details

`files2Spectra2DObject` acts on all files in the current working directory with the specified `fileExt` so there should be no extra files of that type.

Value

A object of class `Spectra2D`. An *unnamed* object of S3 class `Spectra2D` is also written to `out.file`. To read it back into the workspace, use `new.name <- loadObject(out.file)` (`loadObject` is package **R.utils**).

gr.crit and Sample Name Gotchas

The matching of `gr.crit` against the sample file names (in `files2SpectraObject`) or column headers/sample names (in codematrix2SpectraObject) is done one at a time, in order, using grep. While powerful, this has the potential to lead to some "gotchas" in certain cases, noted below.

Your file system may allow file/sample names which `R` will not like, and will cause confusing behavior. File/sample names become variables in `ChemoSpec`, and `R` does not like things like "-" (minus sign or hyphen) in file/sample names. A hyphen is converted to a period (".") if found, which is fine for a variable name. However, a period in `gr.crit` is interpreted from the grep point of view, namely a period matches any single character. At this point, things may behave very differently than one might hope. See `make.names` for allowed characters in `R` variables and make sure your file/sample names comply.

The entries in `gr.crit` must be mutually exclusive. For example, if you have files with names like "Control_1" and "Sample_1" and use `gr.crit = c("Control", "Sample")` groups will be assigned as you would expect. But, if you have file names like "Control_1_Shade" and "Sample_1_Sun" you can't use `gr.crit = c("Control", "Sample", "Sun", "Shade")` because each criteria is grepped in order, and the "Sun/Shade" phrases, being last, will form the basis for your groups. Because this is a grep process, you can get around this by using regular expressions in your `gr.crit` argument to specify the desired groups in a mutually exclusive manner. In this second example, you could use `gr.crit = c("Control(.*)Sun", "Control(.*)Shade", "Sample(.*)Sun", "Sample(.*)Shade")` to have your groups assigned based upon both phrases in the file names.

To summarize, `gr.crit` is used as a grep pattern, and the file/sample names are the target. Make sure your file/sample names comply with `make.names`.

Finally, samples whose names are not matched using `gr.crit` are still incorporated into the `Spectra2D` object, but they are not assigned a group. Therefore they don't plot, but they do take up space in a plot! A warning is issued in these cases, since one wouldn't normally want a spectrum to be orphaned this way.

All these problems can generally be identified by running `sumSpectra` once the data is imported.

Advanced Tricks

While argument `fileExt` appears to be a file extension (from its name and the description elsewhere), it's actually just a grep pattern that you can apply to any part of the file name if you know how to contruct the proper pattern.

Author(s)

Bryan A. Hanson, DePauw University.

`hats_alignSpectra2D` *Align the Spectra in a Spectra2D Object using the HATS algorithm.*

Description

Align the spectra in a `Spectra2D` object using an implementation of the HATS algorithm described by Robinette *et al.*. Currently, only global, not local, alignment is carried out.

Usage

```
hats_alignSpectra2D(spectra, maxF2 = NULL, maxF1 = NULL,
  thres = 0.99, no.it = 20L, restarts = 2L, method = "MBO",
  fill = "noise", plot = FALSE, debug = 1)
```

Arguments

`spectra`	An object of S3 class `Spectra2D`.
`maxF2`	Integer. The most extreme positive `F2step` to allow during the alignment process (units are data points). Search for the optimal alignment will cover the region `-maxColShift` ... `maxColShift` and `-maxRowShift` ... `maxRowShift`.
`maxF1`	Integer. As for `maxF2`, but for F1.
`thres`	Numeric. Prior to launching the optimization, the objective function is evaluated for no shift in case this is actually the best alignment (saving a great deal of time). If this initial check exceeds the value of `thres`, no optimization is performed and the unshifted spectra are returned. The objective function is the cosine of the angle between the unstacked spectra, so `thres` should be on [0 ... 1].
`no.it`	Integer. The maximum number of iterations in the optimization.
`restarts`	Integer. The maximum number of independent rounds of optimization.
`method`	Character. Currently only `method = "MBO"` is available which uses the HATS algorithm plus model based optimization (aka Bayesian optimzation) method to align the spectra. Use `plot = TRUE` to see this in action.
`fill`	Aligning spectra requires that at least some spectra be shifted left/right and up/down. When a spectrum is shifted, spaces are opened that must be filled with something: • If `fill = "zeros"` the spaces are filled with zeros. • If `fill = "rnorm"` the spaces are filled with random noise. • If `fill = "noise"` the spaces are filled with an estimate of the noise from the original spectrum.

`plot` Logical. Shall a plot of the alignment progress be made? The plot is useful for diagnostic purposes. Every step of the alignment has a corresponding plot so you should probably direct the output to a pdf file.

`debug` Integer.

- Values >= 1 give messages about alignment progress in black text.
- Values >= 2 print the merge matrix from the `hclust` object, if `plot` is also `TRUE`. This is the guide tree.
- For `method = "MBO"` values less than 2 suppress some messages and warnings from the underlying functions. If the alignment doesn't work well, set `debug = 2`.
- Setting `plot = TRUE` also gives a view of alignment diagnostics.

Value

An object of S3 class `Spectra2D`.

Author(s)

Bryan A. Hanson, DePauw University.

References

Roughly follows the algorithm described in Robinette et al. 2011 *Anal. Chem.* vol. 83, 1649-1657 (2011) dx.doi.org/10.1021/ac102724x

Examples

```
## Not run:
data(MUD2)
sumSpectra(MUD2)
# You might want to direct the diagnostic output here to a pdf file
# This alignment takes about 90 seconds including the plotting overhead
MUD2a <- hats_alignSpectra2D(MUD2, method = "MBO", debug = 1, plot = TRUE)
mylvls <- seq(3, 30, 3)
col1 <- rep("black", length(mylvls))
col2 <- rep("red", length(mylvls))
col3 <- rep("blue", length(mylvls))
col4 <- rep("green", length(mylvls))
plotSpectra2D(MUD2a, which = c(3, 6, 2, 5),
  lvls = list(mylvls, mylvls, mylvls, mylvls),
  cols = list(col1, col2, col3, col4))

## End(Not run)
```

`import2Dspectra` *Import 2D Spectroscopic Data*

Description

This function imports a single file (for instance, a csv) containing a 2D spectroscopic data set. The current version handles various types of ASCII text files as well as a few other types. This function is called by `files2Spectra2DObject` and is exported and documented to assist in developing new format codes.

Usage

```
import2Dspectra(file, fmt, nF2, ...)
```

Arguments

`file`	Character string giving the path to a file containing a 2D spectrum.
`fmt`	Character string giving the format code to use.
`nF2`	Integer giving the number of data points in the F2 (x) dimension.
`...`	Parameters to be passed to `read.table`.

Value

A list with 3 elements:

- A matrix of the z values. The no. of rows = `nF2` and the no. of columns follows from the size of the imported data.
- A vector giving the F2 (x) values.
- A vector giving the F1 (y) values.

ASCII Format Codes

ASCII format codes are constructed in two parts separated by a hyphen. The first part gives the order of the columns in the file, e.g. F2F1Z means the first column has the F2 values, the second column has the F1 values and the third the intensities. The second part of the format code relates to the order of the rows, i.e. which column varies fastest and in what direction. These codes are best understood in relation to how the data is stored internally in a matrix. The internal matrix is organized exactly as the data appears on the screen, with F2 decreasing left-to-right, and F1 increasing top-to-bottom. There are four possibilities:

- Data in the file consists of F2 slices starting at the bottom (max F1). Use `F2decF1`.
- Data in the file consists of F2 slices starting at the top (min F1). Use `F2incF1`.
- Data in the file consists of F1 slices starting on the left (max F2). Use `F1decF2`.
- Data in the file consists of F1 slices starting on the right (max F1). Use `F1incF2`.

Other Format Codes

Here are some other format codes you can use:

- `SimpleM`. Imports simple matrices composed of z values. The F2 and F1 values are taken from the dimension of the matrix. After import, you will have to manually fix the F2 and F1 values. You may also have to transpose the matrices manually, or perhaps invert the order of the rows or columns.

Author(s)

Bryan A. Hanson, DePauw University.

`inspectLvls` *Inspect Levels for Contour Plots of Spectra2D Objects*

Description

Given a `Spectra2D` object, this function will assist in selecting levels for preparing contour and image type plots. The entire range of the data is used in the histogram so you can choose levels that are suitable for any particular spectrum. However, loading matrices are excluded unless specified as the input. Any of the arguments to `calcLvls` can be used to compute the levels, or you can choose by inspection.

Usage

```
inspectLvls(spectra, loading = NULL, ...)
```

Arguments

`spectra`	An object of S3 class `Spectra2D`.
`loading`	Integer. If a loadings entry is present, draw the histogram for it. For example, if `loading = 1` `Loading_1` is used.
`...`	Arguments to be passed downstream.

Value

A numeric vector giving the levels (invisibly).

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

See `pfacSpectra2D` for further examples.

Examples

```
data(MUD1)
inspectLvls(MUD1, ylim = c(0, 300), main = "All MUD1 Data, mode = even")
inspectLvls(MUD1, ylim = c(0, 300), mode = "NMR",  main = "All MUD1 Data, mode = NMR")
```

`miaSpectra2D` *Multivariate Image Analysis (Tucker 1) of a Spectra2D Object*

Description

Carry out multivariate image analysis of a `Spectra2D` object (multivariate image analysis is the same as a Tucker1 analysis). Function `pcasup1` from package **ThreeWay** is used.

Usage

```
miaSpectra2D(spectra)
```

Arguments

`spectra`	An object of S3 class `Spectra2D`.

Value

A list per `pcasup1`. Of particular interest are the elements `C` containing the eigenvectors and `1c` containing the eigenvalues. We add the class `mia` to the list for our use later, as well as a `method` element for annotating plots.

Author(s)

Bryan A. Hanson, DePauw University.

References

A. Smilde, R. Bro and P. Geladi "Multi-way Analysis: Applications in the Chemical Sciences" Wiley (2004). See especially Example 4.5.

P. Geladi and H. Grahn "Multivariate Image Analysis" Wiley (1996). Note that in this text the meanings of scores and loadings are reversed from the usual spectroscopic uses of the terms.

See Also

For other data reduction methods for `Spectra2D` objects, see `pfacSpectra2D` and `popSpectra2D`.

Examples

```
data(MUD1)
res <- miaSpectra2D(MUD1)
plotScores(MUD1, res, main = "MIA Scores", tol = 1.0, ellipse = "cls")
plotScree(res)
MUD1a <- plotLoadings2D(MUD1, res, load_lvls = seq(-90, 0, 10),
  main = "MIA Comp. 1 Loadings")

# Selection of loading matrix levels can be aided by the following

inspectLvls(MUD1a, loading = 1, ylim = c(0, 80),
  main = "Histogram of Loadings Matrix")
```

MUD *Made Up 2D NMR-Like Data Sets*

Description

Made Up Data that resemble simple, HSQC-like 2D NMR data sets. Lean, low resolution and designed primarily to check graphics and test functions. **As this is made up data, there is no underlying tri-linear structure and therefore one should NOT try to interpret the output of** `miaSpectra2D` **or** `pfacSpectra2D` **run on this data**.

- `MUD1` is intended to test and demonstrate data reduction functions. The HSQC-like data is derived from the 1H and 13C spectra of 3-methyl-1-butanol and the corresponding ethyl ether, idealized slightly for simplicity. There are 10 spectra. Sample 1 is the alcohol; samples 2-5 are the alcohol with local shifts (specifically, two peaks have been shifted +/- one data point). Samples 6-10 are the ether, treated in a similar fashion.
- `MUD2` is intended to test and demonstrate alignment algorithms. The HSQC-like data is derived from the 1H and 13C spectra of 3-methyl-1-butanol, idealized slightly for simplicity. There are 10 spectra. The first one is "correct" and the other samples have global shifts on one or both dimensions.

Format

The data are stored as a `Spectra2D` object.

Author(s)

Bryan A. Hanson, DePauw University.

Source

Created from scratch. Contact the author for a script if interested.

See Also

These data sets are used in the examples of many functions.

`normSpectra2D` *Normalize a Spectra2D Object*

Description

This function carries out normalization of the spectra in a `Spectra2D` object. There are currently two options:

- `"zero2one"` normalizes each 2D spectrum to a $[0 \ldots 1]$ scale.
- `"TotInt"` normalizes each 2D spectrum so that the total area is one.

Usage

```
normSpectra2D(spectra, method = "zero2one")
```

Arguments

`spectra`	An object of S3 class `Spectra2D` to be normalized.
`method`	One of `"TotInt"` or `"zero2one"` giving the method for normalization. Other methods may be added in the future.

Value

An object of S3 class `Spectra2D`.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

`link{centscaleSpectra2D}` for another means of scaling.

Examples

```
data(MUD1)
MUD1n <- normSpectra2D(MUD1)
MUD1b <- removeFreq(MUD1, remF2 = 2.5 ~ 3.5)
MUD1bn <- normSpectra2D(MUD1b)
```

pfacSpectra2D *PARAFAC Analysis of a Spectra2D Object*

Description

Carry out PARAFAC analysis of a Spectra2D object. Function parafac from **multiway** is used. For large data sets, computational time may be long enough that it might desirable to run in batch mode and possibly use parallel processing.

Usage

```
pfacSpectra2D(spectra, parallel = FALSE, setup = FALSE, nfac = 2,
  ...)
```

Arguments

spectra	An object of S3 class Spectra2D.
parallel	Logical. Should parallel processing be used? Unless you love waiting, you should use parallel processing for larger data sets.
setup	Logical. If TRUE the parallel environment will be automatically configured for you. If FALSE, the user must configure the environment themselves (desirable for instance if working on Azure or AWS EC2).
nfac	Integer. The number of factors/components to compute.
...	Additional parameters to be passed to function parafac. You should give thought to value of const, allowed options can be seen in const. The default is to compute an unconstrained solution. However, in some cases one may wish to apply a non-negativity constraint. Also, to suppress the progress bar, you can use verbose = FALSE.

Value

An object of class pfac and parafac, modified to include a list element called $method which is parafac.

Warning

To get reproducible results you will need to set.seed(). See the example.

Author(s)

Bryan A. Hanson, DePauw University.

References

R. Bro "PARAFAC. Tutorial and applications" *Chemometrics and Intelligent Laboratory Systems* vol. 38 pgs. 149-171 (1997).

A. Smilde, R. Bro and P. Geladi "Multi-way Analysis: Applications in the Chemical Sciences" Wiley (2004).

See Also

For other data reduction methods for Spectra2D objects, see miaSpectra2D and popSpectra2D.

Examples

```
data(MUD1)
set.seed(123)
res <- pfacSpectra2D(MUD1, parallel = FALSE, nfac = 2)
plotScores(MUD1, res, leg.loc = "topright",
  ellipse = "cls", main = "PARAFAC Score Plot")
res1 <- plotLoadings2D(MUD1, res,
  load_lvls = c(1, 5, 10, 15, 25),
  main = "PARAFAC Comp. 1 Loadings")
res2 <- plotLoadings2D(MUD1, res, load_lvls = c(1, 5, 10, 15, 25),
  ref = 2, ref_lvls = seq(5, 35, 5),
  ref_cols = rep("black", 7),
  main = "PARAFAC Comp. 1 Loadings + Ref. Spectrum")

# Selection of loading matrix levels can be aided by the following

inspectLvls(res1, loading = 1, ylim = c(0, 50),
  main = "Histogram of Loadings Matrix")
```

---

plotLoadings2D *Plot Loadings from a PARAFAC, MIA or POP Analysis of a Spectra2D Object*

Description

Computes (if necessary) and plots loadings from a PARAFAC, MIA or POP analysis of a Spectra2D object. The loadings matrix has has dimensions F1 x F2 and is a 2D pseudo-spectrum. A reference spectrum may also be drawn.

Usage

```
plotLoadings2D(spectra, so, load = 1, ref = NULL, load_lvls = NULL,
  ref_lvls = NULL, load_cols = NULL, ref_cols = NULL, plot = TRUE,
  ...)
```

Arguments

spectra	An object of S3 class Spectra2D.
so	("Score Object") One of the following: • An object of class mia produced by function miaSpectra2D. • An object of class pfac produced by function pfacSpectra2D.

- An object of class pop produced by function popSpectra2D.

load	An integer specifying the loading to plot.
ref	An integer giving the spectrum in spectra to use as a reference spectrum, which is plotted behind the loadings. Defaults to NULL which does not plot a reference spectrum.
load_lvls	A vector specifying the contour levels for the loadings pseudo-spectrum. If NULL, values are computed using calcLvls.
ref_lvls	A vector specifying the levels at which to compute contours for the reference spectrum. If NULL, values are computed using calcLvls.
load_cols	A vector specifying the colors for the contours in the laoding spectrum. If NULL, defaults to a scheme of values running from blue (low) to red (high), centered on green (zero).
ref_cols	A vector specifying the colors for the contours in the reference spectrum. If NULL, set to gray.
plot	Logical. Shall a plot be made? Plotting large data sets can be slow. Run the function with plot = FALSE, then use inspectLvls to figure out desirable levels, then set plot = TRUE.
...	Additional parameters to be passed to plotting functions. For instance showGrid = TRUE.

Value

The modified Spectra2D object is returned invisibly. The loadings matrix will be appended with a sample of name of Loadings_x where x = load. Side effect is a plot.

Scale

You can view the color scale for the plot via showScale.

Levels & Colors

The number of levels and colors must match, and they are used 1 for 1. If you provide n colors, and no levels, the automatic calculation of levels may return a number of levels other than n, in which case the function will override your colors and assign new colors for the number of levels it computed (with a message). To get exactly what you want, specify both levels and colors in equal numbers. Function inspectLvls can help you choose appropriate levels.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

Please see pfacSpectra2D, miaSpectra2D or popSpectra2D for examples.

plotScores *Plot Scores from PCA, MIA or PARAFAC Analysis of a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via plotScores.

plotScree *Scree Plots from PCA or MIA Analysis of a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via plotScree.

plotSlice *Plot a Slice of a Spectra2D Object*

Description

Plots a slice of a 2D spectrum stored in a Spectra2D object.

Usage

```
plotSlice(spectra, which = 1, F2 = NULL, F1 = NULL,
  showGrid = TRUE, ...)
```

Arguments

spectra	An object of S3 class Spectra2D.
which	A single integer specifying which 2D spectrum from which to plot the slice.
F2	A single frequency to plot.
F1	A single frequency to plot.
showGrid	Logical. If TRUE, show a dotted gray line at each x axis tick mark.
...	Additional parameters to be passed to the plotting routines.

Value

Side effect is a plot.

Note

Only one of F2 or F1 should be given.

Author(s)

Bryan A. Hanson, DePauw University.

Examples

```
data(MUD1)
plotSlice(MUD1, F1 = 22, main = "Slice @ F1 = 22 ppm")
```

---

plotSpectra2D *Plot a Spectra2D Object*

Description

Plots a 2D spectrum stored in a Spectra2D object. This is primarily for inspection and for preparation of final plots. If you need to do extensive exploration, you should probably go back to the spectrometer.

Usage

```
plotSpectra2D(spectra, which = 1, lvls = NULL, cols = NULL,
  showNA = TRUE, showGrid = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra2D.
which	An integer specifying which spectrum to plot. May be a vector.
lvls	A numeric vector specifying the levels at which to compute contours. If NULL, values are computed using calcLvls. If argument which gives more than one spectrum to plot, then lvls must be a list of levels of length(which).
cols	A vector of valid color designations. If provided, must be of the the same length as lvls (i.e. each contour is a particular color). If NULL, defaults to using a scheme of up to nine values running from blue (low) to red (high), centered on green (zero). If argument which gives more than one spectrum to plot, then cols must be a list of colors of length(which).
showNA	Logical. Should the locations of peaks removed by removePeaks2D be shown? If present, these are shown by a gray line at each frequency.
showGrid	Logical. If TRUE, show a dotted gray line at each tick mark.
...	Additional parameters to be passed to the plotting routines.

Value

Side effect is a plot.

Warning

One cannot remove frequencies from the interior of a 2D NMR data set and expect to get a meaningful contour plot, because doing so puts unrelated peaks adjacent in the data set. This would lead to contours being drawn that don't exist in the original data set. This function will check for missing frequencies and stops if any are found.

Scale

You can view the color scale for the plot via `showScale`.

Levels & Colors

The number of levels and colors must match, and they are used 1 for 1. If you provide `n` colors, and no levels, the automatic calculation of levels may return a number of levels other than `n`, in which case the function will override your colors and assign new colors for the number of levels it computed (with a message). To get exactly what you want, specify both levels and colors in equal numbers. Function `inspectLvls` can help you choose appropriate levels.

Overlaying Spectra

If you specify more than one spectrum to plot, e.g. `which = c(1,2)`, then arguments `lvls` and `cols` must be lists of levels and colors, one list element for each spectrum to be plotted (if specified at all). See the examples.

Author(s)

Bryan A. Hanson, DePauw University.

Examples

```
data(MUD1)
mylvls <- seq(5, 35, 5)
plotSpectra2D(MUD1, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7")
plotSpectra2D(MUD1, which = c(6, 1), lvls = list(mylvls, mylvls),
  cols = list(rep("black", 7), rep("red", 7)),
  main = "MUD1 Sample 1 (red) & Sample 6 (black)\n(4 of 6 peaks overlap)")
```

popSpectra2D *Plain Old PCA (POP) of Spectra2D Objects*

Description

This function unstacks a Spectra2D object and conducts IRLBA PCA on it. To unstack, each F1 slice (parallel to F2) is concatenated one after the other so that each 2D spectrum becomes a 1D spectrum. The length of this spectrum will be equal to the length of the F2 dimension times the length of the F1 dimension. PCA is performed on the collection of 1D spectra (one spectrum from each 2D spectrum). The IRLBA algorithm is used because the resulting matrix (n samples in rows x F1 * F2 columns) can be very large, and other PCA algorithms can struggle.

Usage

```
popSpectra2D(spectra, n = 3, choice = "noscale", ...)
```

Arguments

spectra	An object of S3 class Spectra2D.
n	Integer. The number of components desired.
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto").
...	Other parameters to be passed to prcomp_irlba.

Details

The scale choice autoscale scales the columns by their standard deviation. Pareto scales by the square root of the standard deviation. "autoscale" is called "standard normal variate" or "correlation matrix PCA" in some literature. This action is performed on the unstacked matrix, as will centering.

Value

An object of classes prcomp, pop and computed_via_irlba modified to include a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (used to annotate plots).

Author(s)

Bryan A. Hanson, DePauw University.

References

J. Baglama and L. Reichel, "Augmented Implicitly Restarted Lanczos Bidiagonalization Methods" *SIAM J. Sci. Comput.* (2005).

See Also

For other data reduction methods for Spectra2D objects, see miaSpectra2D and pfacSpectra2D.

Examples

```
data(MUD1)
res <- popSpectra2D(MUD1)
plotScores(MUD1, res, main = "POP Scores", ellipse = "cls")
plotScree(res)
MUD1a <- plotLoadings2D(MUD1, res, load_lvls = c(-0.2, -0.1, 0.1, 0.2),
  main = "POP Comp. 1 Loadings")
```

---

removeFreq *Remove Frequencies from a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via removeFreq.

---

removeGroup *Remove Groups from a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via removeGroup.

---

removePeaks2D *Remove Peaks in a Spectra2D Object*

Description

This function sets peaks at specified frequencies in a Spectra2D object to NA. This effectively removes these peaks from calculations of contours which can speed things up and clarifies the visual presentation of data. This function is useful for removing regions with large interfering peaks (e.g. the water peak in 1H NMR), or regions that are primarily noise. This function leaves the frequency axes intact. Note that the parafac function does not allow NA in the input data matrices. See removeFreq for a way to shrink the data set without introducing NAs.

Usage

```
removePeaks2D(spectra, remF2 = NULL, remF1 = NULL)
```

Arguments

spectra	An object of S3 class Spectra2D from which to remove selected peaks.
remF2	A formula giving the range of frequencies to be set to NA. May include "low" or "high" representing the extremes of the spectra. Values outside the range of F2 are tolerated without notice and are handled min or max.See the examples.
remF1	As for remF2.

Value

An object of S3 class Spectra2D.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

removeFreq.

Examples

```
# Note we will set contours a bit low to better
# show what is going on.

data(MUD1)
mylvls <- seq(-0.3, 0.3, 0.1)
mylvls[4] <- 0.05

plotSpectra2D(MUD1, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7: Complete Data Set")

MUD1a <- removePeaks2D(MUD1, remF2 = 2.5 ~ 4)
sumSpectra(MUD1a)
plotSpectra2D(MUD1a, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7\nRemoved Peaks: F2 2.5 ~ 4")

MUD1b <- removePeaks2D(MUD1, remF2 = low ~ 2)
sumSpectra(MUD1b)
plotSpectra2D(MUD1b, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7\nRemoved Peaks: F2 low ~ 2")

MUD1c <- removePeaks2D(MUD1, remF1 = high ~ 23)
sumSpectra(MUD1c)
plotSpectra2D(MUD1c, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7\nRemoved Peaks: F1 high ~ 23")

MUD1d <- removePeaks2D(MUD1, remF2 = 2.5 ~ 4, remF1 = 45 ~ 55)
sumSpectra(MUD1d)
plotSpectra2D(MUD1d, which = 7, lvls = mylvls,
  main = "MUD1 Sample 7\nRemoved Peaks: F2 2.5 ~ 4 & F1 45 ~ 55")
```

removeSample *Remove Samples from a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via removeSample.

shiftSpectra2D *Shift the Spectra in a Spectra2D Object*

Description

Shift the spectra in a Spectra2D object manually. During shifting, some rows or columns are thrown away and new rows or columns are introduced. These new entries may be filled with zeros, or random noise.

- (+) shiftF2 - shift right: trim right, fill left
- (-) shiftF2 - shift left: trim left, fill right
- (+) shiftF1 - shift up: trim top, fill bottom
- (-) shiftF1 - shift down: trim bottom, fill top

Usage

```
shiftSpectra2D(spectra, which = NULL, shiftF2 = 0L, shiftF1 = 0L,
  fill = "noise")
```

Arguments

spectra	An object of S3 class Spectra2D.
which	An integer specifying which spectra to shift. May be a vector.
shiftF2	Integer. The number of data points to shift along the F2 dimension. See Details.
shiftF1	As per shiftF2, but for the F1 dimension.
fill	Aligning spectra requires that at least some spectra be shifted left/right and up/down. When a spectrum is shifted, spaces are opened that must be filled with something: • If fill = "zeros" the spaces are filled with zeros. • If fill = "rnorm" the spaces are filled with random noise. • If fill = "noise" the spaces are filled with an estimate of the noise from the original spectrum.

Value

An object of S3 class Spectra2D.

Author(s)

Bryan A. Hanson, DePauw University.

Examples

```
data(MUD2)
# Show the first two spectra, overlaid

mylvls <- seq(5, 35, 5)
plotSpectra2D(MUD2, which = 1:2, lvls = list(mylvls, mylvls),
  cols = list(rep("black", 7), rep("red", 7)),
  main = "MUD2 Sample 1 (black) & Sample 2 (red)")

MUD2s <- shiftSpectra2D(MUD2, which = 2, shiftF1 = -2)
plotSpectra2D(MUD2s, which = 1:2, lvls = list(mylvls, mylvls),
  cols = list(rep("black", 7), rep("red", 7)),
  main = "MUD2 Sample 1 (black) & Sample 2 (red)\n(samples aligned/overlap)")
```

---

showScale *Display a pdf Version of the Contour Scale*

Description

This function opens the file scale.pdf in an appropriate viewer.

Usage

```
showScale()
```

Examples

```
## Not run:
showScale()

## End(Not run)
```

Spectra2D *Spectra2D Objects*

Description

In ChemoSpec2D, spectral data sets are stored in an S3 class called Spectra2D, which contains a variety of information in addition to the spectra themselves. Spectra2D objects are created by files2Spectra2DObject.

Structure

The structure of a Spectra2D object is a list of eight elements and an attribute as follows: w

element	*type*	*description*
$F2	num	A common frequency (or wavelength) axis corresponding to the F2 dimension in NMR or the x axis more generally. Must be sorted ascending.
$F1	num	A common frequency (or wavelength) axis corresponding to the F1 dimension in NMR or the y axis more generally. Must be sorted ascending.
$data	num	A list of matrices. Each matrix contains a 2D spectrum. Each matrix should have length(F1) rows and length(F2) columns. The matrix must not have dimnames. The low end of the F2\ dimension is last column of the last row (lower right hand corner as typically displayed). The low end of the F1 dimension is the last column of the first row (upper right hand corner). In other words, the spectrum is stored as typically displayed. The list of matrices, if named, should have the same names as names. However, this is not currently enforced.
$names	chr	The sample names for the spectra; length must be no. samples.
$groups	factor	The group classification of the samples; length must be no. samples.
$colors	character	Colors for plotting; length must be no. samples. Colors correspond to groups.
$units	chr	Three entries, the first giving the F2 (x) axis unit, the second the F1 (y) axis unit, and the third the z axis unit, usually some kind of intensity.
$desc	chr	A character string describing the data set.
- attr	chr	"Spectra2D" The S3 class designation.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

sumSpectra to summarize a Spectra2D object. sumGroups to summarize group membership of a

Spectra2D object. chkSpectra to verify the integrity of a Spectra2D object.

Examples

```
data(MUD1)
str(MUD1)
sumSpectra(MUD1)
```

---

sumGroups *Summarize the Group Membership of a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via sumGroups.

---

sumSpectra *Summarize a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via sumSpectra.

Index