Package ‘ChemoSpecUtils’

Type Package
Title Functions Supporting Packages ChemoSpec and ChemoSpec2D
Version 1.0.0
Date 2021-10-09
Description Functions supporting the common needs of packages 'ChemoSpec' and 'ChemoSpec2D'.
License GPL-3
Depends R (>= 3.5)
LazyData TRUE
Suggests ChemoSpec (>= 6.0), ChemoSpec2D (>= 0.5), tinytest, robustbase, RColorBrewer, amap, iriba, ThreeWay, multiway, lattice, roxut, reshape2, patchwork
Imports plyr, ggplot2, ggrepel, plotly, magrittr
URL https://github.com/bryanhanson/ChemoSpecUtils
BugReports https://github.com/bryanhanson/ChemoSpecUtils/issues
ByteCompile TRUE
Encoding UTF-8
RoxygenNote 7.1.2
NeedsCompilation no
Author Bryan A. Hanson [aut, cre] (<https://orcid.org/0000-0003-3536-8246>), Tejasvi Gupta [ctb] (author of ggplot2 graphics)
Maintainer Bryan A. Hanson <hanson@depauw.edu>
Repository CRAN
Date/Publication 2021-10-10 15:20:02 UTC

R topics documented:

  ChemoSpecUtils-package .................................................. 2
  check4Gaps ................................................................. 2
  checkForPackageWithVersion ............................................ 4
check4Gaps

Description

The basic procedure is to compare \( x[n+1] - x[n] \) for successive values of \( n \). When this value jumps, there is a gap which is flagged. \( \text{beg.indx} \) and \( \text{end.indx} \) will always be contiguous as indices must be; it is the \( x \) values that jump or have the gap. The indices are provided as they are more convenient in some programming contexts. If not assigned, the result appears at the console.

Usage

\[
\text{check4Gaps}(x, y = \text{NULL}, \text{silent} = \text{FALSE}, \text{tol} = \text{NULL}, \ldots)
\]
Arguments

x  A numeric vector to be checked for gaps.
y  An optional vector of y-values which correspond to the x values. Only used in ChemoSpec. If provided, a plot will be made in the style of a Spectra object showing the gap(s).
silent  Logical indicating a "no gap" message should not be reported to the console. Important because check4Gaps is called iteratively by other functions.
tol  A number indicating the tolerance for checking to see if the step between successive x values are the same. Depending upon how the x values are stored and rounded, you may need to change the value of tol to avoid flagging trivial "gaps". If NULL, a value is chosen which is just above the median difference between x values.

...  Other parameters to be passed to the plot routines if y is provided, e.g. xlim.

Value

A data frame giving the data chunks found, with one chunk per row. Also a plot if y is provided. In the event there are no gaps found, a data frame with one row is returned. The data frame has columns as follows:

beg.freq  The first frequency value in a given data chunk.
end.freq  The last frequency value in a given data chunk.
size  The length (in frequency units) of the data chunk.
beg.indx  The index of the first frequency value in the data chunk.
end.indx  The index of the last frequency value in the data chunk.

Author(s)

Bryan A. Hanson (DePauw University)

See Also

sumSpectra which make extensive use of this function.

Examples

```r
x <- seq(0, 2 * pi, 0.1)
y <- sin(x)
remove <- c(8:11, 40:45)
x <- x[-remove]
y <- y[-remove]
gaps <- check4Gaps(x, tol = 0.11) # tol just larger than orig spacing

gaps <- check4Gaps(x, y, tol = 0.11) # show a plot if y given
```
checkForPackageWithVersion

Check for an Installed Package with a Particular Version or Newer

Description
Utility function for making sure a package is available with a particular version or newer.

Usage
checkForPackageWithVersion(pkg, vers)

Arguments
pkg Character. The name of the package to check.
vers Character. The minimum acceptable version of the package. Will only be checked to the major.minor level.

Value
If the package with the required version or higher is available, TRUE is returned invisibly. Otherwise FALSE is returned.

Author(s)
Bryan A. Hanson (DePauw University)

chkGraphicsOpt

Check the Graphic Output Option/Mode

Description
This function should be used to check the current graphics mode. It simply reports the current setting unless it is invalid in which case the mode is set to base.

Usage
chkGraphicsOpt(silent = TRUE)

Arguments
silent Logical. Silences most messages if TRUE.

Value
Character. The value of the current graphics output option/mode.
chkSpectra

Author(s)
Bryan A. Hanson (DePauw University), Tejasvi Gupta.

See Also
See GraphicsOptions for more information about the graphics options.

chkSpectra

Verify the Integrity of a Spectra or Spectra2D Object

Description
Utility function to verify that the structure of a Spectra or Spectra2D object is internally consistent. This function should be used after manual editing of these objects. However, in most cases rather than directly editing these objects, one should modify them via:

- removeFreq
- removeSample
- removeGroup

Usage
chkSpectra(spectra, confirm = FALSE)

Arguments

spectra An object of S3 class Spectra or Spectra2D.
confirm Logical indicating whether or not to write the results to the console, as would be desirable for interactive use.

Value
None. When used at the console, and the object is OK, no message is written unless confirm = TRUE. At the console, if there is a problem, messages are issued regardless of the value of confirm.

Author(s)
Bryan A. Hanson (DePauw University)
Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(SrE.IR)
  chkSpectra(SrE.IR)
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)
  chkSpectra(MUD1)
}
```

---

colorSymbol

Color in ChemoSpec and ChemoSpec2D

Description

In ChemoSpec and ChemoSpec2D, the user may use any color name/format known to R. The current color scheme of a `Spectra` or `Spectra2D` object may be determined using `sumGroups` or `sumSpectra`. The colors can also be queried and changed using `conColScheme`.

Format

Colors are stored as character vectors and symbols as numeric vectors.

Details

An important fact to keep in mind is that most people with normal vision cannot distinguish more than about 8-12 colors, and doing so depends upon the viewing circumstances: if on paper, printer, ink and paper type all matter, and if on a screen, the background color makes a big difference. Further, color-blind individuals have additional challenges. A great discussion of color issues can be found in the `colorspace` package. The Polychrome package has further discussion and utilities for choosing qualitative colorschemes, including those for color-blind individuals.

ChemoSpec, but not ChemoSpec2D, can also create plots using the built-in symbols and lower case letters. This is useful for color-blind individuals, plots in `rgl` which can’t plot regular symbols, and plots for where there are more groups than could be reasonably coded in color. A good discussion of which symbols are most readily distinguished can be found in Robinson: "Good Plot Symbols by Default" *Journal of Computational and Graphical Statistics* DOI: 10.1080/10618600.2019.1637746

ChemoSpecUtils supplies four color/symbol schemes for your consideration. If the particular order of colors in any of these does not suit your needs, you can always choose the ones you want, and/or rearrange the order, or simply provide your own.

- The colors and symbols produced by `gr.cols = "auto"` in the import functions.
- `Col8` provides eight unique colors. These are more saturated than the automatic colors.
- `Col12` provides a mostly paired set of 12 unique colors suitable for groups that come in pairs.
- `Col7` provides seven color-blind friendly colors. These can be visualized at https://projects.susielu.com/viz-palette by using the hex codes obtained by typing `data(Col7); Col7` in the R console.
Author(s)

Bryan A. Hanson (DePauw University)

Examples

# Make a plot showing all the built-in color options

data(Col7)
data(Col12)
data(Sym12)
data(Col8)
data(Sym8)
auto <- RColorBrewer::brewer.pal(8, "Set1")

sp <- 0.75 # space between major plot elements
tsp <- 0.15 # additional space between points and color swatches/descriptive text
h <- 0.25 # height of the swatch
y <- 0.0 # bottom of the plot, the reference point

# empty plot
plot(1:12, rep(0.0, 12),
     type = "n", yaxt = "n", xaxt = "n", bty = "n",
     xlab = "", ylab = "", ylim = c(0, 3.5))

  text(6.5, y + h + tsp * 4 + sp * 3.5,
       labels = "Automatic Color & Symbol Options", cex = 1.25, font = 2)

# Col12

  for (i in 1:12) {
    rect(i - 0.5, y, i + 0.5, y + h, border = NA, col = Col12[i])
  }

  points(1:12, rep(y + h + tsp, 12), pch = Sym12)

  text(0.6, y + h + tsp * 2, adj = 0,
       labels = "gr.cols = 'Col12'   12 mostly paired distinct colors/symbols")

# Col8

  for (i in 1:8) {
    rect(i - 0.5, y + sp, i + 0.5, y + sp + h, border = NA, col = Col8[i])
  }

  points(1:8, rep(y + h + tsp + sp, 8), pch = Sym8)

  text(0.6, y + h + tsp * 2 + sp, adj = 0,
       labels = "gr.cols = 'Col8'   8 distinct colors/symbols")

# auto (original)

  for (i in 1:8) {
    rect(i - 0.5, y + sp * 2, i + 0.5, y + sp * 2 + h, border = NA, col = auto[i])
  }

  points(1:8, rep(y + h + tsp + sp * 2, 8), pch = Sym8)
conColScheme

Change the Color Scheme of a Spectra or Spectra2D Object

Description

This function permits you to change the color scheme of an existing Spectra or Spectra2D object.

Usage

conColScheme(spectra, new.cols = NULL, silent = FALSE)

Arguments

spectra     An object of S3 class Spectra or Spectra2D.
new.cols    A character vector giving the new color values, of length(unique(spectra$colors)). If not provided, the function will print the old values for reference.
silent      Logical. If TRUE, suppresses all reporting.

Value

spectra An updated object of S3 class Spectra or Spectra2D.

Author(s)

Bryan A. Hanson (DePauw University)

See Also

For a discussion of general issues of color, see colorSymbol.
GraphicsOptions

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(metMUD1)

  sumSpectra(metMUD1)
  newSpec <- conColScheme(metMUD1) # reports old colors
  newSpec <- conColScheme(metMUD1, new = c("pink", "violet"))
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)

  sumSpectra(MUD1)
  newSpec <- conColScheme(MUD1) # reports old colors
  newSpec <- conColScheme(MUD1, new = c("pink", "violet"))
}
```

---

**GraphicsOptions**  
*Graphic Output Options in ChemoSpec and ChemoSpec2D*

**Description**

In ChemoSpec and ChemoSpec2D, the user may chose from the following graphics output options:

- base graphics (also the only style from the early days of ChemoSpec through version 5).
- ggplot2 graphics, *the default*.
- plotly graphics.

**Details**

Here’s how it works:

- Upon starting ChemoSpec or ChemoSpec2D the graphics output mode is set to ggplot2.
- To see the current value, do `chkGraphicsOpt`. If by some chance the value is corrupted it will be set to base.
- To change the graphics output mode, do `options(ChemoSpecGraphics = 'option')`, where 'option' is one of the options listed above.
- ggplot2 graphics are not available for all plots. If ggplot2 graphics are not available, base graphics will be automatically used, regardless of the setting in `options(ChemoSpecGraphics)`.

What you can do with your plots:

- Base graphics are the original graphics option in R. They cannot be modified.
• For ggplot2 graphics, ChemoSpec and ChemoSpec2D employ theme_bw with only a very few modifications. You can add things to your plot, or override the theme used here via the usual ggplot2 methods. A few simple examples are given below but this is not the place for a ggplot2 tutorial. See https://ggplot2.tidyverse.org/ for all things ggplot2.

• plotly graphics is an interactive graphics option where the user can use the tools provided by plotly package and interact with the plot.

Author(s)

Bryan A. Hanson (DePauw University), Tejasvi Gupta.

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  library("ggplot2")
  data(metMUD1)

  # Using the default ggplot2 graphics
  p1 <- plotSpectra(metMUD1,
                   which = c(10, 11), yrange = c(0, 1.5),
                   offset = 0.06, amplify = 10, lab.pos = 0.5)
  p1

  # Modifying ggplot2 graphics
  # Add a title
  p2 <- p1 + ggtitle("metMUD1 NMR Data")
  p2

  # Zoom the axes
  p3 <- p1 + coord_cartesian(xlim = c(1.5, 2.0))
  p3

  # Change the ggplot2 theme
  p4 <- p1 + theme_gray() + theme(legend.position = "none")
  p4

  # plotLoadings uses patchwork, whose plots are modified differently
  pca <- c_pcaSpectra(metMUD1)
  p5 <- plotLoadings(metMUD1, pca, loads = c(1, 2))
  p5
  p6 <- p5 + patchwork::plot_annotation(title = "metMUD1 NMR Data")
  p6

  # but to change the theme on each subplot, use the & operator
  p7 <- p6 & theme_gray() # compare to p6 + theme_gray()
  p7
}
```
**hcaScores**

**HCA on PCA/MIA/PARAFAC scores from a Spectra or Spectra2D Object**

**Description**

A wrapper which performs HCA on the scores from a PCA of a *Spectra* object or POP/MIA/PARAFAC of a *Spectra2D* object. Many methods for computing the clusters and distances are available.

**Usage**

```r
hcaScores(
  spectra,
  so,
  scores = c(1:5),
  c.method = "complete",
  d.method = "euclidean",
  use.sym = FALSE,
  leg.loc = "topright",
  ...
)
```

**Arguments**

- `spectra` An object of S3 class *Spectra* or *Spectra2D* object.
- `so` "Score Object" One of the following:
  - An object of class `prcomp`, created by ChemoSpec functions `c_pcaSpectra`, `r_pcaSpectra`, `irlba_pcaSpectra` or `s_pcaSpectra`.
  - An object of class `mia` produced by function `miaSpectra2D`.
  - An object of class `parafac` produced by function `pfacSpectra2D`.
  - An object of class `pop` produced by function `popSpectra2D`.

Any of the above score objects will have been 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 the plot).

- `scores` A vector of integers specifying the components (scores) to plot.
- `c.method` A character string describing the clustering method; must be acceptable to `hclust`.
- `d.method` A character string describing the distance calculation method; must be acceptable as a method in `rowDist`.
- `use.sym` A logical; if true, use no color and use lower-case letters to indicate group membership. Applies only to Spectra objects.
- `leg.loc` Character; if "none" no legend will be drawn. Otherwise, any string acceptable to `legend`.
- `...` Additional parameters to be passed to the plotting functions.
plotScores

Value

A list, containing an object of class `hclust` and an object of class `dendrogram`. The side effect is a plot.

Author(s)

Bryan A. Hanson (DePauw University)

See Also

`hclust` for the underlying function. See `hcaSpectra` for HCA of the entire data set stored in the `Spectra` object.

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(metMUD1)

  pca <- c_pcaSpectra(metMUD1)
  hca <- hcaScores(metMUD1, pca, main = "metMUD1 NMR Data PCA Scores")
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)

  mia <- miaSpectra2D(MUD1)
  hca <- hcaScores(MUD1, mia, scores = 1:2, main = "MUD1 MIA Scores")

  set.seed(123)
  pfac <- pfacSpectra2D(MUD1, parallel = FALSE, nfac = 2)
  hca <- hcaScores(MUD1, pfac, scores = 1:2, main = "MUD1 PARAFAC Scores")
}
```

plotScores

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

Description

Plots the requested scores using the color scheme derived from the `Spectra` or `Spectra2D` object. Options are provided to add confidence ellipses for each group in the object. The ellipses may be robust or classical. Option to label the extreme points provided.
plotScores

Usage

plotScores(
  spectra,
  so,
  pcs = c(1, 2),
  ellipse = "none",
  tol = "none",
  use.sym = FALSE,
  leg.loc = "topright",
  ...
)

Arguments

spectra An object of S3 class Spectra or Spectra2D object.
so "Score Object" One of the following:

  - An object of class prcomp, created by ChemoSpec functions c_pcaSpectra,
    r_pcaSpectra, irlba_pcaSpectra or s_pcaSpectra.
  - An object of class mia produced by function miaSpectra2D.
  - An object of class parafac produced by function pfacSpectra2D.
  - An object of class pop produced by function popSpectra2D.

Any of the above score objects will have been 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 the plot).

pcs A vector of two integers specifying the components (scores) to plot.

ellipse A character vector specifying the type of ellipses to be plotted. One of c("both",
  "none", "cls", "rob"). cls specifies classical confidence ellipses, rob specifies robust confidence ellipses. An ellipse is drawn for each group unless there
are three or fewer samples in the group.

tol A number describing the fraction of points to be labeled. tol = 1.0 labels all
the points; tol = 0.05 labels approximately the most extreme 5 percent. Set to
"none" to completely suppress labels. Note that a simple approach based upon
quantiles is used, assumes that both x and y are each normally distributed, and
treats x and y separately. Thus, this is not a formal treatment of outliers, just a
means of labeling points. Groups are lumped together.

use.sym A logical; if TRUE, the color scheme is set to black and the points plotted with
symbols. Applies only to Spectra objects.

leg.loc Either a list with elements x and y, or a string like "topright". Values in a list
should be on [0, 1], i.e. the lower left of the plot area is 0, 0 and the upper right
is 1, 1. String values are those described in legend under "Details". A value of
"none" is acceptable as well.

... Additional parameters to be passed to the plotting functions.
**plotScree**

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

**Description**

Functions that draw a traditional scree plot, or an alternative style that is perhaps more informative. These plots illustrate the variance explained by each component in a PCA or MIA analysis.

**Value**

The returned value depends on the graphics option selected (see `GraphicsOptions`).

- **base**: None. Side effect is a plot.
- **ggplot2**: The plot is displayed, and a ggplot2 plot object is returned if the value is assigned. The plot can be modified in the usual ggplot2 manner.

**Author(s)**

Bryan A. Hanson (DePauw University), Tejasvi Gupta.

**Examples**

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  library("ggplot2")
  data(metMUD1)

  pca <- c_pcaSpectra(metMUD1)
  p <- plotScores(metMUD1, pca, pcs = c(1, 2), ellipse = "cls", tol = 0.05)
  p <- p + ggtitle("metMUD1 NMR Data")
  p
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  library("ggplot2")
  data(MUD1)

  mia <- miaSpectra2D(MUD1)
  p1 <- plotScores(MUD1, mia, tol = 0.1, ellipse = "cls")
  p1 <- p1 + ggtitle("MIA Scores")
  p1

  set.seed(123)
  pfac <- pfacSpectra2D(MUD1, parallel = FALSE, nfac = 2)
  p2 <- plotScores(MUD1, pfac, tol = 0.1, leg.loc = "bottomright")
  p2 <- p2 + ggtitle("PARAFAC Score Plot")
  p2
}
```
Usage

plotScree(pca, style = "alt", ...)

Arguments

pca Either:
- An object of class prcomp, modified to include a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions c_pcaSpectra or r_pcaSpectra were used to create pca.
- An object of class mia produced by function miaSpectra2D.

style Character. One of c("trad","alt") giving the style of plot desired (traditional or alternative). "trad" is not supported for mia objects.

... Parameters to be passed to the plotting routines. Applies to base graphics only.

Value

The returned value depends on the graphics option selected (see GraphicsOptions).

base: None. Side effect is a plot.

ggplot2: The plot is displayed, and a ggplot2 plot object is returned if the value is assigned. The plot can be modified in the usual ggplot2 manner.

Author(s)

Bryan A. Hanson (DePauw University), Tejasvi Gupta.

References

The idea for the alternative style plot came from the NIR-Quimiometria blog by jrcuesta, at https://nir-quimiometria.blogspot.com/2012/02/pca-for-nir-spectrapart-004-projections.html

Examples

if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(metMUD1)

  pca <- c_pcaSpectra(metMUD1)
  p1 <- plotScree(pca, style = "trad")
  p1
  p2 <- plotScree(pca, style = "alt")
  p2
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
}
data(MUD1)

mia <- miaSpectra2D(MUD1)
plotScree(mia, style = "alt")
}

removeFreq

Remove Frequencies from a Spectra or Spectra2D Object

Description

This function removes specified frequencies from a Spectra or Spectra2D object. For instance, one might want to remove regions lacking any useful information (to reduce the data size), remove regions with large interfering peaks (e.g., the water peak in 1H NMR) or simply focus on a region of interest.

Usage

removeFreq(spectra, rem.freq = NULL, remF2 = NULL, remF1 = NULL)

Arguments

spectra
An object of S3 class Spectra or Spectra2D from which to remove frequencies.

rem.freq
For a Spectra object, a vector of logicals. rem.freq can be any valid R statement that leads to a vector of logicals (must be of length(Spectra$freq)). This vector should be TRUE for frequencies you want to be removed and FALSE for those frequencies which will be kept. In the examples, the | and & operators may seem backward in a sense, but R evaluates them one at a time and then combines them to give the desired result. You may wish to look at Comparison and Logic. See the examples. In addition, since January 2020 rem.freq may be a formula as described below.

remF2
Applies to Spectra2D objects. A formula giving the range of frequencies to be extracted. May include "low" or "high" representing the extremes of the spectra. Values outside the range of F2 are tolerated without notice and are handled as min or max. See the examples.

remF1
As for remF2.

Value

An object of S3 class Spectra or Spectra2D.

Modifying Spectra2D Objects

Regarding Spectra2D objects, 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. However, one can remove data from the interior and run a PARAFAC analysis on the result, using the spectrum as an abstract object (that is, the spectrum may not plottable, but the resulting scores are still meaningful).
removeFreq

Author(s)

Bryan A. Hanson (DePauw University)

See Also

removeFreq for another way to remove data.

Examples

if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(SrE.IR)
  sumSpectra(SrE.IR)

  # Examples where rem.freq is a logical vector
  # Remove frequencies from one end:
  newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 3500)

  # Remove frequencies from both ends at once:
  newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 3500
  | SrE.IR$freq < 800)

  # Remove frequencies from the middle:
  newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 800
  & SrE.IR$freq < 1000)

  # The logic of this last one is as follows. Any values
  # that are TRUE will be removed.
  values <- 1:7
  values > 2
  values < 6
  values > 2 & values < 6

  # Examples where rem.freq is a formula

  # Remove frequencies from one end:
  newIR <- removeFreq(SrE.IR, rem.freq = 3500 ~ high)

  # Remove frequencies from both ends is a two step process with formulas:
  newIR <- removeFreq(SrE.IR, rem.freq = 3500 ~ high)
  newIR <- removeFreq(newIR, rem.freq = low ~ 800)

  # Remove frequencies from the middle:
  newIR <- removeFreq(SrE.IR, rem.freq = 800 ~ 1000)

  # After any of these, inspect the results:
  sumSpectra(newIR)
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
# Note we will set contours a bit low to better show what is going on.

data(MUD1)

plotSpectra2D(MUD1, which = 7, lvls = 0.1, cols = "black",
           main = "MUD1 Sample 7: Complete Data Set")

MUD1a <- removeFreq(MUD1, remF2 = 2.5 ~ 4)
sumSpectra(MUD1a) # don't plot, removing peaks from interior is misleading

MUD1b <- removeFreq(MUD1, remF2 = low ~ 2)
sumSpectra(MUD1b)
plotSpectra2D(MUD1b, which = 7, lvls = 0.1, cols = "black",
           main = "MUD1 Sample 7\nRemoved Peaks: F2 low ~ 2")

MUD1c <- removeFreq(MUD1, remF1 = high ~ 23)
sumSpectra(MUD1c)
plotSpectra2D(MUD1c, which = 7, lvls = 0.1, cols = "black",
           main = "MUD1 Sample 7\nRemoved Peaks: F1 high ~ 23")

MUD1d <- removeFreq(MUD1, remF2 = 2.5 ~ 4, remF1 = 45 ~ 55)
sumSpectra(MUD1d) # don't plot, removing peaks from interior is misleading

---

removeGroup

Remove a Group from a Spectra or Spectra2D Object

Description

Removes specified groups from a Spectra or Spectra2D object.

Usage

removeGroup(spectra, rem.group)

Arguments

- spectra: An object of S3 class Spectra or Spectra2D.
- rem.group: A character vector (handled as a regex) giving the groups to be removed.

Details

This function will report if extra data elements are found. These will probably need to be edited manually. The indices reported to the console can be helpful in this regard.

If rem.group is a character vector, the sample names are grepped for the corresponding values. Remember that the grepping process is greedy, i.e. grepping for "XY" find not only "XY" but also "XYZ".

Unused levels in $groups are dropped.
Value

An object of S3 class `Spectra` or `Spectra2D`.

Author(s)

Bryan A. Hanson (DePauw University)

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(SrE.IR)

  sumGroups(SrE.IR)
  SrE.IRa <- removeGroup(SrE.IR, rem.group = "pSrE")
  sumGroups(SrE.IRa)
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)

  sumGroups(MUD1)
  MUD1a <- removeGroup(MUD1, rem.group = "Ether")
  sumGroups(MUD1a)
}
```

removeSample

**Remove Samples from a Spectra or Spectra2D Object**

Description

Removes specified samples from a `Spectra` or `Spectra2D` object.

Usage

```r
removeSample(spectra, rem.sam)
```

Arguments

- `spectra` An object of S3 class `Spectra` or `Spectra2D`.
- `rem.sam` Either an integer vector specifying the samples to be removed, or a character vector (handled as a regex) giving the sample names to be removed.
removeSample

Details

This function will report if extra data elements are found. These will probably need to be edited manually. The indices reported to the console can be helpful in this regard.

If `rem.sam` is a character vector, the sample names are grepped for the corresponding values. Remember that the grepping process is greedy, i.e. grepping for "XY" find not only "XY" but also "XYZ".

Value

An object of S3 class `Spectra` or `Spectra2D`.

Author(s)

Bryan A. Hanson (DePauw University)

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
data(SrE.IR)

  # Remove the 9th spectrum/sample:
  SrE.IR$names
  SrE.IRa <- removeSample(SrE.IR, rem.sam = 9)
  SrE.IRa$names

  # Removes a spectrum/sample with this exact name:
  SrE.IRb <- removeSample(SrE.IR, rem.sam = "NW_adSrE")
  SrE.IRb$names
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
data(MUD1)

  # Removes the 5th spectrum:
  MUD1$names
  MUD1a <- removeSample(MUD1, rem.sam = 5)
  MUD1a$names

  # Removes a spectrum/sample with this exact name:
  MUD1$names
  MUD1b <- removeSample(MUD1, rem.sam = "Ether_3")
  MUD1b$names
}
```
rowDist

*Compute Distance Between Rows of a Matrix*

**Description**

This function computes the distance between rows of a matrix using a number of methods. It is primarily a wrapper for `Dist` which provides many options. However, cosine distance is calculated locally. See the reference for an excellent summary of distances and similarities. Keep in mind that distances are always positive by definition. Further, in the literature one can find the same distance defined different ways. For instance, the definition of the "pearson" and "correlation" distances differs slightly between the reference below and `Dist`. So please study the definitions carefully to get the one you want. The example illustrates the behavior of some common distance definitions. Notice that "pearson" and "cosine" are mathematically identical for the particular definition of "pearson" used by `Dist`.

**Usage**

```r
trowDist(x, method)
```

**Arguments**

- `x`: A matrix whose rows will be used for the distance calculation.
- `method`: Character; one of "cosine", "euclidean", "maximum", "manhattan", "canberra", "binary", "pearson", "correlation", "spearman", "kendall", "abspearson", "abscorrelation".

**Value**

An object of class `dist`.

**Author(s)**

Bryan A. Hanson (DePauw University)

**References**


**Examples**

```r
# This examples imagines spectra as a series of vectors on a half unit circle.
# 1. Compute half of a unit circle
theta <- seq(0, pi, length = 100)
x = cos(theta)
y = sin(theta)
```
# 2. Compute some illustrative vectors
# Get tail/origin & tip/head coordinates
lt <- length(theta)
set.seed(6)
tips <- theta[c(1, sample(2:100, 5))]
x0 <- y0 <- rep(0.0, lt) # tail/origin at 0,0
x1 <- cos(tips) # tips/heads
y1 <- sin(tips)

# 3. Compute the distance functions
# Bounded distances
RDcor <- rep(NA_real_, lt) # correlation distance
RDpea <- rep(NA_real_, lt) # pearson distance
RDabp <- rep(NA_real_, lt) # abspearson distance
RDcos <- rep(NA_real_, lt) # cosine distance

# Unbounded distances
RDmeu <- rep(NA_real_, lt) # Euclidean distance
RDman <- rep(NA_real_, lt) # manhattan distance

# Compute all
np <- 5
refVec <- c(seq(0.0, x[1], length.out = np), seq(0.0, y[1], length.out = np))
for (i in 1:lt) {
  Vec <- c(seq(0.0, x[i], length.out = np), seq(0.0, y[i], length.out = np))
  M <- matrix(c(refVec, Vec), nrow = 2, byrow = TRUE)
  RDman[i] <- rowDist(M, method = "manhattan")
  RDmeu[i] <- rowDist(M, method = "euclidean")
  RDcos[i] <- rowDist(M, method = "cosine")
  RDcor[i] <- rowDist(M, method = "correlation")
  RDpea[i] <- rowDist(M, method = "pearson")
  RDabp[i] <- rowDist(M, method = "abspearson")
}

# 4. Plots
# a. Unit circle w/representative vectors/spectra
plot.new()
plot.window(xlim = c(-1, 1), ylim = c(0, 1), asp = 1)
title(main = "Representative 'Spectral' Vectors on a Unit Half Circle
Reference Vector in Red",
sub = "Each 'spectrum' is represented by a series of x, y points")
lines(x, y, col = "gray") # draw half circle
lines(x = x[c(1,100)], y = y[c(1,100)], col = "gray") # line across bottom
arrows(x0, y0, x1, y1, angle = 5) # add arrows & a red reference vector
arrows(x0[1], y0[1], x1[1], y1[1], col = "red", angle = 5, lwd = 2)

# b. Distances
degrees <- theta*180/pi
plot(degrees, RDman, type = "l",
  xlab = "Angle Between Spectral Vectors and Reference Vector in Degrees",
  ylab = "Distance",
  main = "Spectral Distance Comparisons
Using ChemoSpecUtils::rowDist")
abline(h = c(1.0, 2.0), col = "gray")
sampleDist

*Compute the Distances Between Samples in a Spectra or Spectra2D Object*

**Description**

Compute the distances between samples in a *Spectra* or *Spectra2D* object. This is a means to quantify the similarity between samples. A heat map style plot is an option.

**Usage**

```
sampleDist(spectra, method = "pearson", plot = TRUE, ...)```

**Arguments**

- `spectra`: An object of S3 class *Spectra* or *Spectra2D*.
- `method`: Character. A string giving the distance method. See `rowDist` for options.
- `plot`: Logical. Shall a level plot (heat map) be made?
- `...`: Arguments to be passed to the plotting function.

**Value**

A numeric matrix giving the distances between the samples.

**Author(s)**

Bryan A. Hanson (DePauw University)

**See Also**

For *Spectra* objects, see `plotSpectraDist` which compares all spectra to a single reference spectrum.
sumGroups

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

**Description**

This function summarizes the group membership of a Spectra or Spectra2D object.

**Usage**

sumGroups(spectra)

**Arguments**

- **spectra**: An object of S3 class Spectra or Spectra2D whose group membership information is desired.
sumSpectra

Value

A data frame as follows. Note that if there are groups with no members these are dropped.

<table>
<thead>
<tr>
<th>group</th>
<th>The name of the group.</th>
</tr>
</thead>
<tbody>
<tr>
<td>no.</td>
<td>The number in the group.</td>
</tr>
<tr>
<td>color</td>
<td>The color assigned to the group.</td>
</tr>
<tr>
<td>symbol</td>
<td>The symbol assigned to the group. Spectra objects only.</td>
</tr>
<tr>
<td>alt.symbol</td>
<td>The alternative symbol assigned to the group. Spectra objects only.</td>
</tr>
</tbody>
</table>

Author(s)

Bryan A. Hanson (DePauw University)

See Also

To summarize the entire object, sumSpectra.

Examples

```r
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(SrE.IR)
  sumGroups(SrE.IR)
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)
  sumGroups(MUD1)
}
```

sumSpectra Summarize a Spectra or Spectra2D Object

Description

Provides a summary of a Spectra or Spectra2D object, essentially a more spectroscopist-friendly version of str().

Usage

`sumSpectra(spectra, ...)`

Arguments

| spectra | An object of S3 class Spectra or Spectra2D whose group membership information is desired. |
| ...     | Arguments to be passed downstream. Main use is to pass a value for tol to function check4Gaps when using ChemoSpec. Not used in ChemoSpec2D. |
Update Group Names in a Spectra or Spectra2D Object

Description

A convenience function that can be used to update (change) group names. The default group names come from the gr.crit argument in the import functions `files2SpectraObject`, `matrix2SpectraObject` or `files2Spectra2DObject`. In some cases gr.crit may have complex regex patterns, and this function makes updating them to more appropriate/more readable strings easier.

Usage

```r
updateGroups(spectra, new.grps = NULL, silent = FALSE)
```

Arguments

- **spectra**: An object of S3 class `Spectra` or `Spectra2D`.
- **new.grps**: A vector of character values giving the new group names. The new values must correspond to the order of the old values. This vector should give the unique values only (so, it should have `length(unique(spectra$groups))`). If not provided, the function will print the old values for reference.
**updateGroups**

```
silent Logical. If TRUE, suppresses all reporting.
```

**Value**

```
spectra An updated object of S3 class Spectra or Spectra2D.
```

**Author(s)**

Bryan A. Hanson (DePauw University)

**Examples**

```
if (checkForPackageWithVersion("ChemoSpec", 6.0)) {
  library("ChemoSpec")
  data(metMUD1)
  metMUD1a <- updateGroups(metMUD1) # reports old groups
  metMUD1a <- updateGroups(metMUD1, new.grps = c("C", "T"))
}

if (checkForPackageWithVersion("ChemoSpec2D", 0.5)) {
  library("ChemoSpec2D")
  data(MUD1)
  MUD1a <- updateGroups(MUD1, new.grps = c("control", "treatment"))
}
```
Index

* **classes**
  * chkSpectra, 5

* **cluster**
  * hcaScores, 11

* **color**
  * colorSymbol, 6
  * conColScheme, 8

* **datasets**
  * colorSymbol, 6

* **hplot**
  * plotScores, 12
  * plotScree, 14
  * sampleDist, 23

* **multivariate**
  * hcaScores, 11
  * plotScores, 12
  * plotScree, 14

* **robust**
  * plotScores, 12

* **utilities**
  * check4Gaps, 2
  * chkGraphicsOpt, 4
  * chkSpectra, 5
  * colorSymbol, 6
  * conColScheme, 8
  * GraphicsOptions, 9
  * removeFreq, 16
  * removeGroup, 18
  * removeSample, 19
  * rowDist, 21
  * sumGroups, 24
  * sumSpectra, 25

* c_pcaSpectra, 11, 13, 15
  * check4Gaps, 2, 25
  * checkForPackageWithVersion, 4

* ChemoSpecUtils
  * (ChemoSpecUtils-package), 2
  * ChemoSpecUtils-package, 2
  * chkGraphicsOpt, 4, 9

  * chkSpectra, 5, 26
  * Col12 (colorSymbol), 6
  * Col7 (colorSymbol), 6
  * Col8 (colorSymbol), 6
  * ColorScheme (colorSymbol), 6
  * colorSymbol, 6
  * Comparison, 16
  * conColScheme, 6, 8
  * dendrogram, 12
  * Dist, 21

  * files2Spectra2DObject, 26
  * files2SpectraObject, 26

  * GraphicsOptions, 5, 9, 14, 15

  * hcaScores, 11
  * hcaSpectra, 12
  * hclust, 11, 12

  * irlba_pcaSpectra, 11, 13

  * legend, 11, 13
  * Logic, 16

  * matrix2SpectraObject, 26
  * miaSpectra2D, 11, 13

  * pfacSpectra2D, 11, 13
  * plotScores, 12
  * plotScree, 14
  * plotSpectraDist, 23
  * popSpectra2D, 11, 13
  * prcomp, 11, 13, 15

  * r_pcaSpectra, 11, 13, 15
  * removeFreq, 5, 16, 17
  * removeGroup, 5, 18
  * removeSample, 5, 19
  * rowDist, 11, 21, 23
INDEX

s_pcaSpectra, 11, 13
sampleDist, 23
Spectra, 3, 5, 6, 8, 11–13, 16, 18–20, 23–27
Spectra2D, 5, 6, 8, 11–13, 16, 18–20, 23–27
sumGroups, 6, 24
sumSpectra, 3, 6, 25, 25
Sym12 (colorSymbol), 6
Sym8 (colorSymbol), 6

updateGroups, 26