Package 'ChemoSpecUtils'

March 1, 2019

Type Package

Title Functions Supporting Packages ChemoSpec and ChemoSpec2D

Version 0.2.211

Date 2019-02-28

Description Functions supporting the common needs of packages 'ChemoSpec' and 'ChemoSpec2D'.

License GPL-3

Depends R (>= 3.1)

Suggests ChemoSpec (>= 5.0), ChemoSpec2D, testthat

Imports plyr, robustbase, RColorBrewer, amap

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

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

ByteCompile TRUE

RoxygenNote 6.1.1

NeedsCompilation no

Author Bryan A. Hanson [aut, cre] (ORCID 0000-0003-3536-8246)

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

Repository CRAN

Date/Publication 2019-03-01 05:40:03 UTC

R topics documented:

`ChemoSpecUtils-package` *Functions Supporting Packages ChemoSpec and ChemoSpec2D*

Description

Functions supporting the packages ChemoSpec and ChemoSpec2D.

Author(s)

Bryan A. Hanson.

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

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

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. `beg.indx` and `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

```
check4Gaps(x, y = NULL, silent = FALSE, tol = NULL, ...)
```

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

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

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

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(SrE.IR)
  chkSpectra(SrE.IR)
}

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)
  chkSpectra(MUD1)
}
```

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 MIA/PARAFAC of a Spectra2D object. Many methods for computing the clusters and distances are available.

Usage

```
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") 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. • An object of class parafac produced by function pfacSpectra2D.
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.

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

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(metMUD1)

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

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)

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

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

---

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.

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") 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.
- An object of class parafac produced by function pfacSpectra2D.

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 the most extreme 5 percent. Set to "none" to completely suppress labels.
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	Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
...	Additional parameters to be passed to the plotting functions.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

Examples

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(metMUD1)

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

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)

  res <- miaSpectra2D(MUD1)
  plotScores(MUD1, res, main = "MIA Scores", tol = 0.1, ellipse = "cls")

  set.seed(123)
```

```
  res <- pfacSpectra2D(MUD1, parallel = FALSE, nfac = 2)
  plotScores(MUD1, res, tol = 0.1, leg.loc = "bottomright", main = "PARAFAC Score Plot")
}
```

`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.

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.
`...`	Additional parameters to be passed to plotting functions.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

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 (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(metMUD1)

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

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  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.
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 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).

Author(s)

Bryan A. Hanson, DePauw University.

See Also

removePeaks2D for another way to remove data.

Examples

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(SrE.IR)
  sumSpectra(SrE.IR)

  # 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

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

```
if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)

  plotSpectra2D(MUD1, which = 7, lvls = seq(-1, 1, by = 0.2),
    main = "MUD1 Sample 7: Complete Data Set")

  MUD1a <- removeFreq(MUD1, remF2 = 2 ~ 4)
  sumSpectra(MUD1a) # cannot plot, results would be misleading

  MUD1b <- removeFreq(MUD1, remF1 = low ~ 15)
  sumSpectra(MUD1b)
  plotSpectra2D(MUD1b, which = 7, lvls = seq(-1, 1, by = 0.2),
    main = "MUD1 Sample 7\nRemoved Frequencies: F1 low ~ 15")

  MUD1c <- removeFreq(MUD1, remF2 = low ~ 5)
  sumSpectra(MUD1c)
  plotSpectra2D(MUD1c, , which = 7, lvls = seq(-1, 1, by = 0.2),
    main = "MUD1 Sample 7\nRemoved Frequencies: F2 low ~ 5")

  MUD1d <- removeFreq(MUD1, remF2 = 6 ~ high, remF1 = 13 ~ 17)
  sumSpectra(MUD1d) # not plotted, results would be 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

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(SrE.IR)

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

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)

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

---

removeSample *Remove Samples from a Spectra or Spectra2D Object*

Description

Removes specified samples from a Spectra or Spectra2D object.

Usage

```
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.

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

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  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 (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  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$names
  MUD1b <- removeSample(MUD1, rem.sam = "Sample6")
  MUD1b$names
}
```

rowDist *Compute Distance Between Rows of a Matrix*

Description

This function is a wrapper to compute the distance between rows of a matrix using a number of methods. Some of these are available in package stats and some in Dist from package amap. This function determines which method is requested and then the distance calculation is done by the appropriate method. The exception is the cosine distance which is calculated locally.

Usage

```
rowDist(x, method)
```

Arguments

x	A matrix whose rows will be used for the distance calculation.
method	A character; one of c("pearson", "correlation", "spearman", "kendall", "euclidean", "maximum", "manhattan", "canberra","binary", "minkowski", "cosine").

Details

Methods c("euclidean", "maximum", "manhattan", "canberra","binary", "minkowski") are sent to function dist in package stats while methods c("pearson", "correlation", "spearman", "kendall") are handled by Dist in package amap. See the respective help pages for details. "cosine" is handled locally.

Value

An object of class dist.

Author(s)

Bryan A. Hanson, DePauw University. Suggested by and original code written by Roberto Canteri.

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.

Value

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

group	The name of the group.
no.	The number in the group.
color	The color assigned to the group.
symbol	The symbol assigned to the group. Spectra objects only.
alt.symbol	The alternative symbol assigned to the group. Spectra objects only.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

To summarize the entire object, sumSpectra.

Examples

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(SrE.IR)
  sumGroups(SrE.IR)
}

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  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. Currently not used.

Details

Prior to summarizing, chkSpectra is run with confirm = FALSE. If there are problems, warnings are issued to the console and the summary is not done. The Spectra or Spectra2D object is checked to see if it contains data elements beyond what is required. If so, these extra elements are reported to the console.

Value

None. Results printed at console.

Author(s)

Bryan A. Hanson, DePauw University.

Examples

```
if (requireNamespace("ChemoSpec", quietly = TRUE)) {
  library("ChemoSpec")
  data(SrE.IR)
  sumSpectra(SrE.IR)
}

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)
  sumSpectra(MUD1)
}
```

---

updateGroups *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 readible strings easier.

Usage

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

Value

spectra An updated object of S3 class Spectra or Spectra2D.

Examples

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

if (requireNamespace("ChemoSpec2D", quietly = TRUE)) {
  library("ChemoSpec2D")
  data(MUD1)
  MUD1a <- updateGroups(MUD1, new.grps = c("control", "treatment"))
}
```

Index