Package 'ChemoSpec'

December 12, 2017

Type Package

Title Exploratory Chemometrics for Spectroscopy

Version 4.4.97

Date 2017-12-12

Description A collection of functions for top-down exploratory data analysis of spectral data obtained via nuclear magnetic resonance (NMR), infrared (IR) or Raman spectroscopy. Includes functions for plotting and inspecting spectra, peak alignment, hierarchical cluster analysis (HCA), principal components analysis (PCA) and model-based clustering. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed with metabolomics data sets in mind, where the samples fall into groups such as treatment and control. Graphical output is formatted consistently for publication quality plots. ChemoSpec is intended to be very user friendly and help you get usable results quickly. A vignette covering typical operations is available.

License GPL-3

Depends R (>= 3.1)

Imports plyr, stats, utils, grDevices

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

ByteCompile TRUE

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

VignetteBuilder knitr

Suggests mvbutils, sna, knitr, IDPmisc, js, NbClust, lattice, baseline, mclust, pls, clusterCrit, R.utils, RColorBrewer, seriation, MASS, robustbase, grid, pcaPP, jsonlite, gsubfn, amap, signal, rgl, readJDX, speaq

RoxygenNote 6.0.1

NeedsCompilation no

Author Bryan A. Hanson [aut, cre], Mike Bostock [cph, ctb] (author of the d3.js library used in plotSpectraJS, http://d3js.org)

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

Repository CRAN

Date/Publication 2017-12-12 17:15:08 UTC

R topics documented:

`ChemoSpec-package` *Exploratory Chemometrics for Spectroscopy*

Description

A collection of functions for top-down exploratory data analysis of spectral data obtained via nuclear magnetic resonance (NMR), infrared (IR) or Raman spectroscopy. Includes functions for plotting and inspecting spectra, peak alignment, hierarchical cluster analysis (HCA), principal components analysis (PCA) and model-based clustering. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed with metabolomics data sets in mind, where the samples fall into groups such as treatment and control. Graphical output is formatted consistently for publication quality plots. ChemoSpec is intended to be very user friendly and help you get usable results quickly. A vignette covering typical operations is available.

Author(s)

Bryan A. Hanson and Matthew J. Keinsley.

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

References

https://github.com/bryanhanson/ChemoSpec

aovPCAloadings *Plot aovPCAscores Loadings of a Spectra Object*

Description

Uses the results from aovPCAscores to plot the corresponding loadings.

Usage

```
aovPCAloadings(spectra, LM, pca, plot = 1, loads = 1, ref = 1, ...)
```

Arguments

spectra	An object of S3 class Spectra.
LM	List of matrices created by aovPCAscores.
pca	PCA output from aovPCAscores.
plot	An integer specifying the desired plot. names(LM) will show which matrix has which data in it.
loads	An integer vector giving the loadings to plot. More than 3 loadings creates a useless plot using the default graphics window.
ref	An integer specifying the reference spectrum to plot, which appears at the bottom of the plot.
...	Additional parameters to be passed to plotting functions.

Value

None. Side effect is a plot.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University.

References

Pinto, Bosc, Nocairi, Barros, and Rutledge. "Using ANOVA-PCA for Discriminant Analysis: ..." Analytica Chimica Acta 629.1-2 (2008): 47-55.

Harrington, Vieira, Espinoza, Nien, Romero, and Yergey. "Analysis of Variance–Principal Component Analysis: ..." Analytica Chimica Acta 544.1-2 (2005): 118-27.

https://github.com/bryanhanson/ChemoSpec

See Also

An example using this function can be seen in aov_pcaSpectra. See also plotLoadings.

aovPCAscores *Plot ANOVA-PCA Scores from a Spectra Object*

Description

Uses the results from aov_pcaSpectra to conduct PCA and plot the scores. Argument plot is used to select a matrix from those in LM. The residual error matrix is then added to the selected matrix before performing PCA. Use names(LM) to see which factor is stored in which matrix.

Usage

```
aovPCAscores(spectra, LM, plot = 1, type = "class", choice = NULL, ...)
```

Arguments

spectra	An object of S3 class Spectra.
LM	List of matrices created by aov_pcaSpectra.
plot	An integer specifying which scores to plot.
type	Either classical ("cls") or robust ("rob"); Results in either c_pcaSpectra or r_pcaSpectra being called on the Spectra object.
choice	The type of scaling to be performed. See c_pcaSpectra and r_pcaSpectra for details.
...	Additional parameters to be passed to plotScores. For example, you can plot confidence ellipses this way. Note that ellipses are drawn based on the groups in spectra$groups, but the separation done by aov_pcaSpectra is based on argument fac. These may not correspond, but you can edit spectra$groups to match if necessary.

Value

Returns the PCA results, and creates the requested plot.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University.

References

Pinto, Bosc, Nocairi, Barros, and Rutledge. "Using ANOVA-PCA for Discriminant Analysis: ..." Analytica Chimica Acta 629.1-2 (2008): 47-55.

Harrington, Vieira, Espinoza, Nien, Romero, and Yergey. "Analysis of Variance–Principal Component Analysis: ..." Analytica Chimica Acta 544.1-2 (2005): 118-27.

https://github.com/bryanhanson/ChemoSpec

See Also

The use of this function can be seen in aov_pcaSpectra. See also plotScores.

aov_pcaSpectra *ANOVA-PCA Analysis of Spectra Data*

Description

ANOVA-PCA is a combination of both methods developed by Harrington. The data is partitioned into submatrices corresponding to each experimental factor, which are then subjected to PCA separately after adding the residual error back. If the effect of a factor is large compared to the residual error, separation along the 1st PC in the score plot should be evident. With this method, the significance of a factor can be visually determined (ANOVA-PCA is not blind to group membership). ANOVA-PCA with only one factor is the same as standard PCA and gives no additional separation.

Usage

```
aov_pcaSpectra(spectra, fac)
```

Arguments

spectra	An object of S3 class Spectra.
fac	A vector of character strings giving the factors to be used in the analysis. These should be elements of Spectra. Note that there should be 2 or more factors, because ANOVA-PCA on one factor is the same as standard PCA. See the example.

Value

A list of matrices for each factor and their interactions, along with the residual error and mean centered data matrix.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University.

References

Pinto, Bosc, Nocairi, Barros, and Rutledge. "Using ANOVA-PCA for Discriminant Analysis: ..." Analytica Chimica Acta 629.1-2 (2008): 47-55.

Harrington, Vieira, Espinoza, Nien, Romero, and Yergey. "Analysis of Variance–Principal Component Analysis: ..." Analytica Chimica Acta 544.1-2 (2005): 118-27.

https://github.com/bryanhanson/ChemoSpec

See Also

This function calls avgFacLvls, and the results are used in aovPCAscores and aovPCAloadings.

Examples

```
data(metMUD2)

# Original factor encoding:
levels(metMUD2$groups)

# Split those original levels into 2 new ones (re-code them)
new.grps <- list(geneBb = c("B", "b"), geneCc = c("C", "c"))
mM3 <- splitSpectraGroups(metMUD2, new.grps)

# run aov_pcaSpectra
mats <-aov_pcaSpectra(mM3, fac = c("geneBb", "geneCc"))
apca1 <- aovPCAscores(mM3, mats, plot = 1, main = "aovPCA: B vs b")
apca2 <- aovPCAscores(mM3, mats, plot = 2, main = "aovPCA: C vs c")
apca3 <- aovPCAscores(mM3, mats, plot = 3, main = "aovPCA: Interaction Term")
apca4 <- aovPCAloadings(spectra = mM3, LM = mats, pca = apca1,
  main = "aov_pcaSpectra: Bb Loadings")
```

---

avgFacLvls *Average Levels of a Factor in a Data Matrix*

Description

avgFacLvls takes a matrix and calculates the column means for each level of each factor given. It then replaces the original matrix rows with the means corresponding to the factor/level memership of a particular sample (row).

Usage

```
avgFacLvls(matrix, fac)
```

Arguments

matrix	A matrix.
fac	A vector of character strings with length = nrow(matrix)

Value

A matrix whose rows are composed of the column means for each level of the factor.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

aov_pcaSpectra for full details.

Examples

```
M1 <- matrix(rnorm(100), nrow = 20, byrow = TRUE)
facs <- factor(c(rep("A",5), rep("B",5), rep("C", 5), rep("D", 5)))
M2 <- avgFacLvls(M1, fac = facs)
```

---

baselineSpectra *Baseline Correction of a Spectra Object*

Description

This function mostly wraps functions in package **baseline** which carries out a variety of baseline correction routines. A simple linear correction method is also available.

Usage

```
baselineSpectra(spectra, int = TRUE, retC = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra to be checked.
int	Logical; if TRUE, do the correction interactively using widgets. No results are saved. Use this for inspection and exploration only.
retC	Logical: shall the baseline-corrected spectra be returned in the Spectra object?
...	Other arguments passed downstream. The relevant ones can be found in baseline. Be sure to pay attention to argument method as you will probably want to use it. You can also use method = "linear" for a simple linear fit, see Details.

Details

In plots using methods from the baseline package, the x axis ticks give the data point index, not the original values from your data. Note that you cannot zoom the non-interactive display of corrected spectra because the underlying function hardwires the display. Try the interactive version instead (int = TRUE), or use plotSpectra on the corrected data. In addition to the methods provided by baseline, you can also use method = "linear". This correction is handled locally, and is very simple: a line is drawn from the first data point to the last, and this becomes the new baseline. This is most suitable for cases in which the baseline rises or falls steadily, as is often seen in chromatograms.

Value

If `int = TRUE`, an interactive plot is created. If `int = FALSE` and `retC = FALSE`, an object of class `baseline` is returned (see `baseline-class`). If `int = FALSE` and `retC = TRUE`, a `Spectra` object containing the corrected spectra is returned. In these latter two cases plots are also drawn.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
require("IDPmisc") # needed specifically for rfbaseline
temp <- baselineSpectra(SrE.IR, int = FALSE, method = "rfbaseline")
```

---

binData *Bin or Bucket Data*

Description

This function accepts a vector of x-values and averages them in groups of `bin.ratio` data points. It also accepts a vector of y-values and sums them in groups of `bin.ratio` data points. Both x and y data can be processed in the same call, or they can be processed separately. An internal function, not generally called by the user.

Usage

```
binData(x = NULL, y = NULL, bin.ratio = 2)
```

Arguments

`x`	An optional vector of x values to be averaged in groups of `bin.data` points.
`y`	An optional vector of y values to be summed in groups of `bin.data` points.
`bin.ratio`	An integer giving the binning ratio, that is, the number of points to be grouped together into one subset of data.

Details

The x and y values must be contiguous in the sense that there are no gaps in the values (i.e., x[n + 1] - x[n] must be the same for the entire data set; this can be checked with `diff` and is checked internally. Note that this function is normally called by `binSpectra` and that function can handle gaps, sending each continuous piece of data here to be binned. If length(x or y) is not divisible by bin.ratio to give a whole number, data points are removed from the beginning of x or y until it is, and the number of data points removed is reported at the console. The algorithm forces the requested bin.ratio to be used.

Value

Depending upon the input, a data frame containing one or both of the following elements:

`mean.x`	A vector of the averaged x values. Length will be approximately length(x)/bin.ratio, with length(x) adjusted as described above if this does not give a whole number.
`sum.y`	A vector of the summed y values. Length will be approximately length(y)/bin.ratio, with length(y) adjusted as described above if this does not give a whole number.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
x <- seq(0, 1000, length.out = 3000); y <- rnorm(3000)
res <- binData(x, y)
length(res$mean.x) # will be half of the original length

# Now try it with bin.ratio that does not divide into 3000
res <- binData(x, y, bin.ratio = 7)
length(res$mean.x)
```

`binSpectra` *Bin or Bucket a Spectra Object*

Description

This function will bin a `Spectra` object by averaging every `bin.ratio` frequency values, and summing the corresponding intensity values. The net effect is a smoothed and smaller data set. If there are gaps in the frequency axis, each data chunk is processed separately. Note: some folks refer to binning as bucketing.

Usage

```
binSpectra(spectra, bin.ratio)
```

Arguments

spectra	An object of S3 class Spectra to be binned.
bin.ratio	An integer giving the binning ratio, that is, the number of points to be grouped together into one subset of data.

Details

If the frequency range is not divisible by bin.ratio to give a whole number, data points are removed from the beginning of the frequency data until it is, and the number of data points removed is reported at the console. If there are gaps in the data where frequencies have been removed, each continuous piece is sent out and binned separately (by binSpectra).

Value

An object of S3 class Spectra.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(metMUD1)
sumSpectra(metMUD1)
res <- binSpectra(metMUD1, bin.ratio = 4)
sumSpectra(res)
```

---

check4Gaps *Check for Missing Values (Gaps) in a Spectra Object*

Description

This function may be used with a Spectra object to see if there are any gaps or discontinuities in the frequency axis. Gaps may arise when unwanted frequencies are removed (e.g, water peaks in 1H NMR, or uninteresting regions in any kind of spectroscopy). As written, it can be used to check for gaps in any appropriate numeric vector. A plot of the gaps is optional.

Usage

```
check4Gaps(x, y = NULL, tol = 0.01, plot = FALSE, silent = FALSE, ...)
```

Arguments

x	A numeric vector to be checked for gaps.
y	An optional vector of y-values which correspond to the x values. Only needed if plot = TRUE.
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".
plot	Logical indicating if a plot of the gaps should be made. If TRUE, y must be provided. The plot is labeled consistent with calling this function on a Spectra object.
silent	Logical indicating a "no gap" condition (return value is FALSE) should not be reported to the console. Important because check4Gaps is called iteratively by other functions.
...	Other parameters to be passed to the plot routines if plot = TRUE, e.g. xlim.

Details

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.

Value

A data frame giving the data chunks found, with one chunk per line. Also a plot if requested. In the event there are no gaps found, FALSE is returned.

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.
eng.indx	The index of the last frequency value in the data chunk.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
x <- seq(from = 5, to = 12, by = 0.1)
remove <- c(8:11, 40:45); x <- x[-remove]
gaps <- check4Gaps(x)

data(SrE.IR)
newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 1800 & SrE.IR$freq < 2500)
check4Gaps(newIR$freq, newIR$data[1,], plot = TRUE)
```

---

chkSpectra *Verify the Integrity of a Spectra Object*

Description

Utility function to verify that the structure of a Spectra object (an S3 object) is internally consistent. This function can be used after manual editting of a Spectra object. However, in most cases rather than directly manipulating a Spectra object, one should manipulate it via removeFreq, removeGroup or removeSample.

Usage

```
chkSpectra(spectra, confirm = FALSE)
```

Arguments

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

Details

This function is similar in spirit to validObject in the S4 world. 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. When used in a function, this function is silent (assuming confirm = FALSE) unless there is a problem.

Value

None; messages will be printed at the console if there is a problem.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(metMUD1)
chkSpectra(metMUD1, confirm = TRUE) # OK

# What's next is the wrong way to manipulate a Spectra object.
# One should removeSample instead.
# We won't run during checking as an error is raised

## Not run:

remove <- c(20:40)
metMUD1$freq <- metMUD1$freq[-remove]
chkSpectra(metMUD1, confirm = TRUE) # not OK, you didn't listen to me!

## End(Not run)
```

---

clupaSpectra *Hierarchical Cluster-Based Peak Alignment on a Spectra Object*

Description

This function is a wrapper to several functions in the **speaq** package. It implements the CluPA algorithm described in the reference.

Usage

```
clupaSpectra(spectra, bT = NULL, ...)
```

Arguments

spectra	An object of S3 class Spectra.
bT	Numeric. The baseline threshold. Defaults to five percent of the range of the data, in spectra$data. Passed to detectSpecPeaks.
...	Other arguments to be passed to the underlying functions.

Value

A modifed Spectra object.

Author(s)

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

References

Vu TN, Valkenborg D, Smets K, Verwaest KA, Dommisse R, Lemiere F, Verschoren A, Goethals B, Laukens K. "An integrated workflow for robust alignment and simplified quantitative analysis of NMR spectrometry data" BMC Bioinformatics vol. 12 pg. 405 (2011).

https://github.com/bryanhanson/ChemoSpec

Examples

```
# July 2017: This function requires pkg speaq which in turn
# requires pkg data.table, which is broken in the CRAN build
# chain.  We make a check of the status so we can pass CRAN!

if (requireNamespace("speaq", quietly = TRUE)) {

data(alignMUD)

plotSpectra(alignMUD, which = 1:20, lab.pos = 4.5, offset = 0.1,
  yrange = c(0, 1900), amp = 500, xlim = c(1.5, 1.8),
  main = "Misaligned NMR Spectra (alignMUD)")

aMUD <- clupaSpectra(alignMUD)
plotSpectra(aMUD, which = 1:20, lab.pos = 4.5, offset = 0.1,
  yrange = c(0, 1900), amp = 500, xlim = c(1.5, 1.8),
  main = "Aligned NMR Spectra (alignMUD)")

} # end of namespace check
```

---

colLeaf *Color the Leaves of a Dendrogram Based on a Spectra Object*

Description

This function colors the leaves of a dendrogram object. The code was taken from the help files. An internal function, not generally called by the user.

Usage

```
colLeaf(n, spectra)
```

Arguments

n	A node in a dendrogram object.
spectra	An object of S3 class Spectra.

Value

Returns a node with the label color properties set.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

---

colorSymbol *Colors and Symbols in ChemoSpec and Spectra Objects*

Description

In ChemoSpec, the user may use any color name/format known to R. For ease of comparison, it would be nice to plan ahead and use the same color scheme for all your plots. The current color scheme of a Spectra object may be determined using sumGroups or changed using conColScheme. Also, splitSpectraGroups has another means of changing the color scheme, but this is intended for the situations when you are creating new categories/groups for your samples.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

---

conColScheme *Change the Color Scheme of a Spectra Object*

Description

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

Usage

```
conColScheme(spectra, old = levels(as.factor(spectra$colors)), new = NULL)
```

Arguments

spectra	An object of S3 class Spectra whose color scheme is to be changed.
old	A character vector of the old color names; will be searched for and replaced one-for-one by the character vector in new.
new	A character vector of the new (replacement) color schemes.

Details

A grepping process is used. Depending upon the nature of the old color names to be searched for, you might need to add some grep pattern matching strings to the color name.

Value

An object of S3 class Spectra whose color scheme has been changed.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

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

Examples

```
data(metMUD1)
sumSpectra(metMUD1)
newSpec <- conColScheme(metMUD1, new = c("pink", "violet"))
sumSpectra(newSpec)
```

---

coordProjCS *Modified Version of coordProj from mclust*

Description

This is a modified version of the function coordProj from package **mclust**. In this version, the original symbol scheme for the error plot is changed to simply plot an X over the letters identifying the groups. An internal function, not generally called by the user.

Usage

```
coordProjCS(data, dimens = c(1, 2), parameters = NULL, z = NULL,
  classification = NULL, truth = NULL, uncertainty = NULL,
  what = c("classification", "errors", "uncertainty"), quantiles = c(0.75,
  0.95), symbols = NULL, colors = NULL, scale = FALSE, xlim = NULL,
  ylim = NULL, CEX = 1, PCH = ".", identify = FALSE, ...)
```

Arguments

data	See coordProj.
dimens	See coordProj.
parameters	See coordProj.
z	See coordProj.
classification	See coordProj.
truth	See coordProj.
uncertainty	See coordProj.
what	See coordProj.
quantiles	See coordProj.
symbols	See coordProj.
colors	See coordProj.
scale	See coordProj.
xlim	See coordProj.
ylim	See coordProj.
CEX	See coordProj.
PCH	See coordProj.
identify	See coordProj.
...	See coordProj.

Value

See coordProj.

Author(s)

Bryan A. Hanson, DePauw University. Derived from coordProj.

References

https://github.com/bryanhanson/ChemoSpec

cv_pcaSpectra *Cross-Validation of Classical PCA Results for a Spectra Object*

Description

This function carries out classical PCA on the data in a Spectra object using a cross-validation method. A simple re-write of Peter Filzmoser's pcaCV method with some small plotting changes.

Usage

```
cv_pcaSpectra(spectra, pcs, choice = "noscale", repl = 50, segments = 4,
  segment.type = c("random", "consecutive", "interleaved"), length.seg,
  trace = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pcs	As per pcaCV where it is called amax; an integer giving the number of PC scores to include.
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto").
repl	As per pcaCV; the number of replicates to perform.
segments	As per pcaCV.
segment.type	As per pcaCV.
length.seg	As per pcaCV.
trace	As per pcaCV.
...	Parameters to be passed to the plotting routines.

Value

A list as described in pcaCV, so the result must be assigned or it will appear at the console. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University. Derived from pcaCV.

References

K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

https://github.com/bryanhanson/ChemoSpec

See Also

pcaCV for the underlying function.

Examples

```
data(SrE.IR)
pca <- cv_pcaSpectra(SrE.IR, pcs = 5)
```

---

c_pcaSpectra *Classical PCA of Spectra Objects*

Description

A wrapper which carries out classical PCA analysis on a Spectra object. The user can select various options for scaling. There is no normalization by rows - do this manually using normSpectra. There is an option to control centering, but this is mainly for compatibility with the aov_pcaSpectra series of functions. Centering the data should always be done in PCA and it is the default here.

Usage

```
c_pcaSpectra(spectra, choice = "noscale", cent = TRUE)
```

Arguments

spectra	An object of S3 class Spectra.
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto").
cent	Logical: whether or not to center the data. Always center the data unless you know it to be already centered.

Details

The scale choice autoscale scales the columns by their standard deviation. Pareto scales by the square root of the standard deviation.

Value

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 plots which you might make).

Author(s)

Bryan A. Hanson, DePauw University.

References

K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

https://github.com/bryanhanson/ChemoSpec

See Also

prcomp for the underlying function, r_pcaSpectra for analogous robust PCA calculations.

For displaying the results, plotScree, plotScores, plotLoadings, plot2Loadings, sPlotSpectra, plotScores3D, plotScoresRGL.

Examples

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

---

evalClusters *Evaluate or Compare the Quality of Clusters Quantitatively*

Description

This function is a wrapper to two functions: intCriteria function in package **clusterCrit**, and NbClust in package **NbClust**. It can be used to quantitatively compare different clustering options.

Usage

```
evalClusters(spectra, pkg = "NbClust", hclst = NULL, k = NULL, h = NULL,
  crit = "Dunn", ...)
```

Arguments

spectra	An object of S3 class Spectra.
pkg	Character. One of c("NbClust", "clusterCrit"). The package to use for comparing clusters.
hclst	An object of S3 class hclust. Only applies to pkg = "clusterCrit".
k	Integer. The number of groups in which to cut the tree (hclust). Only applies to pkg = "clusterCrit".
h	Numeric. The height at which to cut the tree. Either k or h must be given, with k taking precedence. See cutree. Only applies to pkg = "clusterCrit".
crit	String. A string giving the criteria to be used in evaluating the quality of the cluster. See liintCriteria. Only applies to pkg = "clusterCrit".

... Other parameters to be passed to the functions. In particular, the default NbClust package will need some parameters. See the example.

Details

Both of the packages used here compute very similar quantities. For details, see the publication and respective vignettes. Package **clusterCrit** takes the approach in which you cluster in a separate step using whatever parameters you like, then the tree is cut either at a given height or in such a way as to produce a fixed number of groups. One or more indices are then computed. Then, you repeat this process with different clustering criteria, and compare. Package **NbClust** allows one to specify a range of possible number of clusters and a few other parameters and will return indices corresponding to each set options, which is somewhat more automated.

Value

A list giving the results, as described in intCriteria or NbClust.

Author(s)

Bryan A. Hanson, DePauw University.

References

M. Charrad et. al. "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set." J. Stat. Soft. vol. 61 no. 6 October 2014.

https://github.com/bryanhanson/ChemoSpec

See Also

hclust for the underlying base function. hcaSpectra for HCA analysis of a Spectra object. hcaScores for HCA analysis of PCA scores from a Spectra object. plotHCA for the plotting function in ChemoSpec.

Examples

```
# These are a little slow for CRAN checking
## Not run:

data(metMUD2)

# Using clusterCrit
res1 <- hcaSpectra(metMUD2) # default clustering and distance methods
res2 <- hcaSpectra(metMUD2, d.method = "cosine")
# The return value from hcaSpectra is a list with hclust as the first element.
crit1 <- evalClusters(metMUD2, pkg = "clusterCrit", hclst = res1[[1]], k = 2)
crit2 <- evalClusters(metMUD2, pkg = "clusterCrit", hclst = res2[[1]], k = 2)
# crit1 and crit2 can now be compared.

# Using NbClust
res3 <- evalClusters(metMUD2, min.nc = 2, max.nc = 5, method = "average", index = "kl")
```

```
## End(Not run)
```

`files2SpectraObject` *Import Data into a Spectra Object*

Description

These functions import data into a `Spectra` object. They use `read.table` to read files so they are very flexible in regard to file formatting. **Be sure to see the ...argument below for important details you need to provide.**

Usage

```
files2SpectraObject(gr.crit = NULL, gr.cols = c("auto"),
  freq.unit = "no frequency unit provided",
  int.unit = "no intensity unit provided",
  descrip = "no description provided", fileExt = "\\.(csv|CSV)$",
  out.file = "mydata", debug = FALSE, ...)

matrix2SpectraObject(gr.crit = NULL, gr.cols = c("auto"),
  freq.unit = "no frequency unit provided",
  int.unit = "no intensity unit provided",
  descrip = "no description provided", in.file = NULL,
  out.file = "mydata", chk = TRUE, ...)
```

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. A maximum of 8 groups can automatically be assigned colors and symbols. If you have more than 8 groups, you will need to provide a vector of colors (see below) and manually fix the symbols after the `Spectra` object is created.
`gr.cols`	Group Colors. Either the word "auto", in which case colors will be automatically assigned, or a vector of acceptable color names with the same length as `gr.crit`. In the latter case, colors will be assigned one for one, so the first element of `gr.crit` is assigned the first element of `gr.col` and so forth. A maximum of 8 colors can be assigned automatically, after that, you must give a vector of colors. See details below for some other issues to consider.
`freq.unit`	A character string giving the units of the x-axis (frequency or wavelength).
`int.unit`	A character string giving the units of the y-axis (some sort of intensity).

descrip — A character string describing the data set that will be stored. This string is used in some plots so it is recommended that its length be less than about 40 characters.

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 Spectra will be written to this file.

debug — Logical. Applies to files2SpectraObject only. Set to TRUE for troubleshooting when an error is thrown during import. In addition, values of 1-5 will work when importing a JCAMP-DX file via fileExt = ".jdx" etc. These will be passed through to the readJDX function. See there for much more info on importing JCAMP-DX files.

... — 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.

in.file — Character. Applies to matrix2SpectraObject only. Input file name, including extension.

chk — Logical. Applies to matrix2SpectraObject only. Should the Spectra object be checked for integrity? If you are having trouble importing your data, set this to FALSE and do str(your object) to troubleshoot.

Value

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

Functions

- files2SpectraObject: Import data from separate csv files
- matrix2SpectraObject: Import a matrix of data

files2SpectraObject

files2SpectraObject acts on all files in the current working directory with the specified fileExt so there should be no extra files of that type hanging around (except see next paragraph). The first column should contain the frequency values and the second column the intensity values. The files may have a header or not (supply header = TRUE/FALSE as necessary). The frequency column is assumed to be the same in all files.

If fileExt contains any of "dx", "DX", "jdx" or "JDX", then the files will be processed by readJDX. Consider setting debug = TRUE for this format, as there are many options for JCAMP, and many are untested. See readJDX for known limitations.

matrix2SpectraObject

This function takes a csv-like file, containing frequencies in the first column, and samples in additional columns, and processes it into a Spectra object. The file MUST have a header row which includes the sample names. There need not be a header for the first (frequency) column.

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 Spectra object, but they are not assigned a group or color. 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.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

The linking of groups with colors is handled by `groupNcolor`.

Examples

```
# Grab an included file
ed <- system.file("extdata", package = "ChemoSpec")
tf <- "PCRF.jdx"
chk <- file.copy(from = file.path(ed, tf), to = file.path(getwd(), tf),
overwrite = TRUE)

# Now read in the file, and plot
spec <- files2SpectraObject(gr.crit = "PCRF", freq.unit = "ppm", int.unit = "intensity",
  descrip = "test import", fileExt = ".jdx")
sumSpectra(spec)
plotSpectra(spec, lab.pos = 3.5, main = "Reduced Fat Potato Chip")
```

---

groupNcolor *Assign Group Membership and Colors for a Spectra Object*

Description

A utility function which looks for `gr.crit` in the file names of .csv files and assigns group membership (max 8 groups automatically). Also assigns a color, a symbol and an alternate symbol to each group. Warnings are given if there are file names that don't match entries in `gr.crit` or there are entries in `gr.crit` that don't match any file names. An internal function, not generally called by the user.

Usage

```
groupNcolor(spectra, gr.crit = NULL, gr.cols = c("auto"))
```

Arguments

`spectra`	An object of S3 class `Spectra`. Until this function acts on `spectra` it is not quite complete.
`gr.crit`	As per `files2SpectraObject`.
`gr.cols`	As per `files2SpectraObject`.

Value

A *complete* object of S3 class `Spectra`. This function is the last internal step in creating a `Spectra` object. Until this function has done its job, an object of class `Spectra` will not pass checks as the assembly is not complete (see `chkSpectra`).

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

files2SpectraObject for details; sumGroups to see the outcome.

---

hcaScores *HCA on PCA scores from a Spectra Object*

Description

A wrapper which performs HCA on the scores from a PCA of a Spectra object, color-coding the results as specified in the object. Many methods for computing the clusters and distances are available.

Usage

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

Arguments

spectra	An object of S3 class Spectra.
pca	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.
scores	A vector of integers specifying which scores to use for the HCA.
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.
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.

References

https://github.com/bryanhanson/ChemoSpec

See Also

hclust for the underlying function. See hcaSpectra for HCA of the entire data set stored in the Spectra object. plotHCA for the function that actually does the plotting.

Examples

```
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
res <- hcaScores(SrE.IR,  pca, scores = c(1:5), main = myt)
```

---

hcaSpectra *Plot HCA Results of a Spectra Object*

Description

A wrapper which carries out HCA and plots a dendrogram colored by the information in a Spectra object. Many methods for computing the clusters and distances are available.

Usage

```
hcaSpectra(spectra, c.method = "complete", d.method = "euclidean",
  use.sym = FALSE, leg.loc = "topright", ...)
```

Arguments

spectra	An object of S3 class Spectra.
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.
leg.loc	Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
...	Other 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.

References

https://github.com/bryanhanson/ChemoSpec

See Also

hclust for the underlying function. hcaScores for similar analysis of PCA scores from a Spectra object. plotHCA for the function that actually does the plotting.

Examples

```
data(SrE.IR)
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
res <- hcaSpectra(SrE.IR, main = myt)
```

---

hmapSpectra *Seriated Heat Map for a Spectra Object*

Description

Creates a heat map with marginal dendrograms using seriation procedures. Very briefly, the samples that are most like each other occur in one corner, and the frequencies that are most informative with respect to the samples are in that corner as well. This is achieved by using heirchical cluster analysis and then re-ordering the clusters in a coordinated way across each dimension. See the vignette for package **seriation**.

Usage

```
hmapSpectra(spectra, ...)
```

Arguments

spectra	An object of S3 class Spectra.
...	Additional arguments to be passed downstream. A great deal of control is available - check hmap for details. Most of the control actually derives from the heatmap2 function in package **gplots**.

Value

A list composed of two data frames. One is the frequencies and their rankings, the other is samples and their rankings. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

hmap which will get you to the package (there is no package index page); the vignette is a good place to begin (browseVignettes("seriation")).

Examples

```
data(SrE.IR)
# Let's look just at the carbonyl region
IR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 1850 | SrE.IR$freq < 1650)
res <- hmapSpectra(IR, col = heat.colors(5), labCol = FALSE)
```

---

hypTestScores *Conduct MANOVA using PCA Scores and Factors in a Spectra Object*

Description

This function provides a convenient interface for carrying out manova using the scores from PCA and the factors (groups) stored in a Spectra object. The function will do anova as well, if you only provide one vector of scores, though this is probably of limited use. A Spectra object contains group information stored in its spectra$groups element, but you can also use splitSpectraGroups to generate additional groups/factors that might be more useful than the original.

Usage

```
hypTestScores(spectra, pca, pcs = 1:3, fac = NULL, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp.
pcs	An integer vector giving the PCA scores to use as the response in the manova analysis.
fac	A character vector giving the factors to be used in the manova. They will be searched for within the Spectra object.
...	Additional arguments to be passed downstream, in this case to aov. Untested.

Details

This function is an extraordinarily thin wrapper which helps the user to avoid writing a very tedious `formula` specification.

Value

The results of the analysis print to the console unless assigned. If assigned, the object class is one of several described in `aov` depending upon the data passed to it.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

`splitSpectraGroups` which can be used to create additional factor elements in the `Spectra` object, which can then be used with this function.

Examples

```
data(metMUD2)

# Original factor encoding:
levels(metMUD2$groups)

# Split those original levels into 2 new ones (re-code them)
new.grps <- list(geneBb = c("B", "b"), geneCc = c("C", "c"))
mM3 <- splitSpectraGroups(metMUD2, new.grps)

# Now do the PCA and anova
pca <- c_pcaSpectra(mM3)
hypTestScores(mM3, pca, fac = c("geneBb", "geneCc"))
```

---

isWholeNo *Determine if a Number is a Whole Number*

Description

This function determines if a given number is a whole number within a given tolerance. Taken from the help page of `is.integer`. An internal function, not generally called by the user.

Usage

```
isWholeNo(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	A number to be tested.
tol	Tolerance for the test.

Value

A logical, indicating the outcome of the test.

Author(s)

Bryan A. Hanson, DePauw University. Carved out of is.integer.

References

https://github.com/bryanhanson/ChemoSpec

See Also

is.integer

---

labelExtremes *Label Extreme Values in a 2D Data Set*

Description

A utility function which plots the sample names next to the sample points. The number of samples labeled can be specified by passing it from the calling function. An internal function, not generally called by the user.

Usage

```
labelExtremes(data, names, tol)
```

Arguments

data	A matrix containing the x values of the points/samples in the first column, and the y values in the second.
names	A character vector of sample names. Length must match the number of rows in x.

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. Note that this is simply based upon quantiles, 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. Note too that while this function could deal with groups separately, the way it is called by `plotScoresDecoration` lumps all groups together.

Value

None. Annotates the plot with labels.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

---

labelExtremes3d *Identify Extreme Values in 3D*

Description

A utility function to identify the extreme values in a 3D plot data set, presumably so that they can be labeled. Algorithm is similar to `labelExtremes`, except that `labelExtremes3d` does not do the plotting (because the results are used by functions that use different plotting paradigms). An internal function, not generally called by the user.

Usage

```
labelExtremes3d(data, names, tol)
```

Arguments

data — A matrix of 3 columns containing x, y and z values for the labels, with rows corresponding to sample names.

names — A character vector of sample names; must have length equal to `nrow(data)`.

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. Note that this is simply based upon quantiles, assumes that x, y and z are each normally distributed, and treats x, y and yz separately. Thus, this is not a formal treatment of outliers, just a means of labeling points. Note too that while this function could deal with groups separately, the way it is called by `plotScoresRGL` lumps all groups together.

Value

A data frame containing the x, y and z coordinates, along with the corresponding labels.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

`loopThruSpectra` *Display the Spectra in a Spectra Object One at a Time*

Description

Plots each spectrum in a `Spectra` object one at a time, and waits for a return in the console before plotting the next spectrum. Use `ESC` to get out of the loop.

Usage

```
loopThruSpectra(Spectra, ...)
```

Arguments

`Spectra`	An object of S3 class `Spectra`.
`...`	Parameters to be passed downstream.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
## Not run:

data(metMUD1)
loopThruSpectra(metMUD1)

## End(Not run)
```

makeEllipsoid *Create Ellipsoid*

Description

Given at least 3 data points, this function creates either classical or robust ellipsoids at a given confidence limit, in either 2D or 3D. The ellipsoids consist of randomly generated points, which if plotted as tiny points, create a sort of transparent surface. An internal function, not generally called by the user.

Usage

```
makeEllipsoid(data, cl = 0.95, rob = FALSE, frac.pts.used = 0.8)
```

Arguments

data	A matrix of at least 3 data points, with x, y and optionally z in columns. See details.
cl	The confidence limit desired.
rob	Logical, indicating if robust ellipsoids are to be computed.
frac.pts.used	If rob = TRUE, this is the fraction of points to be considered the "good" part of the data. See the documentation for cov.rob for details.

Details

If only x and y are provided, at least 3 points must be given, as 2 points defines a line, not an ellipse. For 3D data, and rob = FALSE, at least 4 points must be provided. If rob = TRUE, 5 points would be theoretically required, but the code forces 8 to avoid unusual cases which would fail. If fewer than 8 are given, the computation shifts to classical with a warning. Note that depending upon how this function is called, one may end up with classical and robust ellipsoids in the plot. Remember too that because the points are randomly generated, the x, y pairs or x, y, z triplets are not related to each other, and one cannot plot lines from point to point. See the example for a 2D ellipse. If you want a function that generates x, y points suitable for connecting to each other via lines, see plotScoresCor.

Value

A matrix of 2 or 3 columns, representing x, y and optionally z. These are the coordinates of points specifying an ellipse which has a likelihood of containing the true mean at the given confidence limit.

Note

The idea was taken from "An Introduction to rggobi" found at the ggobi web site: http://www.ggobi.org. I added the robust option.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

cov.rob for the function that does the work.

Examples

```
# 2D example
x <- rnorm(10, 2, 0.5)
y <- rnorm(10, -2, 2)
ell <- makeEllipsoid(cbind(x,y), cl = 0.99)
plot(ell[,1], ell[,2], col = "red", pch = 20, cex = 0.3)
points(x,y)
```

---

mclust3D *mclust Analysis in 3D*

Description

This function conducts an mclust analysis of the data provided, and plots the points in 3D using **rgl** graphics. An option is provided for displaying either classical or robust confidence ellipses. An internal function not generally called by the user. See mclust3dSpectra instead.

Usage

```
mclust3D(data, ellipse = TRUE, rob = FALSE, cl = 0.95,
  frac.pts.used = 0.8, truth = NULL, title = "no title provided",
  t.pos = NULL, lab.opts = FALSE, use.sym = FALSE, ...)
```

Arguments

data	A matrix of 3 columns (corresponding to x, y, z) and samples in rows.
ellipse	Logical indicating if confidence ellipses should be drawn.
rob	Logical; if ellipse = TRUE, indicates that robust confidence ellipses should be drawn. If FALSE, classical confidence ellipses are drawn.
cl	A number indicating the confidence interval for the ellipse.
frac.pts.used	If ellipse = TRUE and rob = TRUE, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.

truth	A character vector indicating the known group membership for reach row of the PC scores. Generally this would be spectra$groups.
title	A character string for the plot title.
t.pos	A character selection from LETTERS[1:8] (= A through H) indicating the desired location for the title.
lab.opts	A logical indicating whether or not to display the locations where the title and legend can be placed. These locations are the corners of a cube surrounding the data.
use.sym	logical; if true, the color scheme is changed to black and symbols are used for plotting.
...	Other parameters to be passed downstream.

Value

The mclust model is returned invisibly, and a plot is produced.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

Mclust for background on the method.

Examples

```
## Not run:

require("mclust")
set.seed(666)
x <- c(rnorm(10, 3, 0.5), rnorm(10, -1, 0.5))
y <- c(rnorm(10, 1, 1), rnorm(10, -4, 0.5))
z <- c(rnorm(10, -2, 0.5), rnorm(10, 3, 0.5))
x[15] <- y[15] <- z[15] <- 4 # screw up one point
my.truth <- c(rep("Z", 10), rep("Q", 10))
mclust3D(cbind(x, y, z), title = "mclust3D demo",
 t.pos = "G", truth = my.truth)

## End(Not run)
```

mclust3dSpectra *mclust Analysis of a Spectra Object in 3D*

Description

This function conducts an mclust analysis of the PCA results of a Spectra object and displays the results in 3D. Classical or robust confidence ellipses can be added if desired. Improperly classified data points can be marked. rgl graphics are employed.

Usage

```
mclust3dSpectra(spectra, pca, pcs = c(1:3), ellipse = TRUE, rob = FALSE,
  cl = 0.95, frac.pts.used = 0.8, truth = NULL,
  title = "no title provided", t.pos = NULL, lab.opts = FALSE,
  use.sym = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp.
pcs	An integer vector describing which PCs to use.
ellipse	Logical indicating if confidence ellipses should be drawn.
rob	Logical; if ellipse = TRUE, indicates that robust confidence ellipses should be drawn. If FALSE, classical confidence ellipses are drawn.
cl	A number indicating the confidence interval for the ellipse.
frac.pts.used	If ellipse = TRUE and rob = TRUE, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.
truth	A character vector indicating the known group membership for reach row of the PC scores. Generally this would be spectra$groups. #' @param title A character string for the plot title.
title	A character string giving the title.
t.pos	A character selection from LETTERS[1:8] (= A through H) indicating the desired location for the title.
lab.opts	A logical indicating whether or not to display the locations where the title and legend can be placed. These locations are the corners of a cube surrounding the data.
use.sym	Logical; if true, the color scheme is changed to black and symbols are used for plotting.
...	Other parameters to be passed downstream.

Details

If you intend to make a hard copy of your plot, use `lab.opts = TRUE` until you have found a good view of your data. Then note corners of the cube where the title won't interfere with viewing the data, and use this for `t.pos`, and add `title`. Adjust as necessary, then turn off label display using `lab.opts = FALSE`. Back at the console, use `> rgl.snapshot("file_name.png")` to create the hardcopy.

Note that the confidence ellipses computed here are generated independently of the `Mclust` results - they do not correspond to the ellipses seen in 2D plots from `Mclust`.

Value

The mclust model is returned invisibly, and a plot is produced.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

`Mclust` for background on the method.

Examples

```
## Not run:

require(mclust)
data(metMUD1)
class <- c_pcaSpectra(metMUD1)
mclust3dSpectra(metMUD1, class, title = "mclust3dSpectra demo",
 lab.opts = FALSE, t.pos = "A")

## End(Not run)
```

---

mclustSpectra *mclust Analysis of a Spectra Object PCA Results*

Description

This function is a wrapper for the `Mclust` function and associated plotting functions.

Usage

```
mclustSpectra(spectra, pca, pcs = c(1:3), dims = c(1, 2), plot = c("BIC",
  "proj", "error"), use.sym = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp.
pcs	An integer vector describing which PCs to use.
dims	A integer vector giving the PCA dimensions to use.
plot	A character string indicating what plot to make. Options are c("BIC", "proj", "error"); see Mclust for details.
use.sym	Logical; if true, the color scheme is changed to black and symbols are used for plotting.
...	Other parameters to be passed downstream.

Value

The Mclust model is returned invisibly, and a plot is made.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

Mclust for background on the method.

Examples

```
require("mclust")
data(SrE.IR)
class <- c_pcaSpectra(SrE.IR, choice = "autoscale")
mclustSpectra(SrE.IR, class, main = "Cuticle IR", plot = "BIC")
mclustSpectra(SrE.IR, class, main = "Cuticle IR", plot = "proj")
mclustSpectra(SrE.IR, class, main = "Cuticle IR", plot = "error",
 truth = metMUD1$groups)
```

metMUD1 *Made Up NMR Data Sets*

Description

These data sets are simulated 300 MHz NMR spectra. They are designed mainly to illustrate certain chemometric methods and are small enough that they process quickly.

Format

The data is stored as a Spectra object.

Details

alignMUD is a series of mis-aligned spectra of a single small organic molecule.

metMUD1 is composed of 20 samples, each a mixture of four typical small organic compounds (we'll leave it to the reader as an exercise to deduce the spin systems!). These compounds are present in varying random amounts. Ten of the samples are control samples, and ten are treatment samples. Thus you can run PCA and other methods on this data set, and expect to see a separation. This data set is normalized.

metMUD2 also consists of 20 samples of mixtures of the same four compounds. However, the concentrations of some of the compounds are correlated with other compounds, both positively and negatively, and some concentrations are random. metMUD2 is divided into different sample groups which correspond conceptually to two genes, each active or knocked out. This data set is designed to be similar to a metabolomics data set in which the concentrations of some compounds co-vary, and others are independent. This data set is normalized.

Author(s)

Bryan A. Hanson, DePauw University.

Source

Created using various tools. Contact the author for a script if interested.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(metMUD1)
sumSpectra(metMUD1)
data(metMUD2)
sumSpectra(metMUD2)
```

normSpectra *Normalize a Spectra Object*

Description

This function carries out normalization of the spectra in a Spectra object. There are currently four options:

- "PQN" carries out "Probabalistic Quotient Normalization" as described in the reference. This is probably the best option for many data sets.
- "TotInt" normalizes by total intensity. In this case, the y-data of a Spectra object is normalized by dividing each y-value by the sum of the y-values in a given spectrum. Thus each spectrum sums to 1. This method assumes that the total concentration of all substances giving peaks does not vary across samples which may not be true.
- "Range" allows one to do something similar but rather than using the sum of the entire spectrum as the denominator, only the sum of the given range is used. This would be appropriate if there was an internal standard in the spectrum which was free of interferance.
- "zero2one" scales each spectrum separately to a [0 . . . 1] scale. This is sometimes useful for visual comparison of chromatograms but is inappropriate for metabolomic data sets.

Usage

```
normSpectra(spectra, method = "PQN", RangeExpress = NULL)
```

Arguments

spectra	An object of S3 class Spectra to be normalized.
method	One of c("PQN", "TotInt", "Range", "zero2one") giving the method for normalization.
RangeExpress	A logical expression giving the frequency range over which to sum intensities, before dividing the entire spectrum by the summed values. For examples of constructing these expressions, see the examples in removeFreq.

Value

An object of S3 class Spectra.

Author(s)

Bryan A. Hanson, DePauw University.

References

Probabalistic Quotient Normalization is reported in F. Dieterle et. al. Analytical Chemistry vol. 78 pages 4281-4290 (2006). The exact same mathematics are called "median fold change normalization" by Nicholson's group, reported in K. A. Veselkov et. al. Analytical Chemistry vol. 83 pages 5864-5872 (2011).

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
res <- normSpectra(SrE.IR)
sumSpectra(res)
```

---

normVec *Normalize a Vector to range -1 to +1*

Description

Each value of the vector passed to the function is divided by the square root of the sum of every value squared, producing a new vector whose range is restricted to, at most, -1 to +1. Note that this assumes that the mean of the original vector is zero. An internal function, not generally called by the user.

Usage

```
normVec(x)
```

Arguments

x	A numeric argument whose values are to be normalized.

Value

The normalized vector.

Note

The idea was taken from "An Introduction to rggobi" found at the ggobi web site: http://www.ggobi.org.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
x1 <- rnorm(20, 2, 2)
range(x1)
sd(x1)/diff(range(x1))

x2 <- normVec(x1)
range(x2)
sd(x2)/diff(range(x2))
```

---

pcaDiag *Outlier Diagnostic Plots for PCA of a Spectra Object*

Description

A function to carry diagnostics on the PCA results for a Spectra object. Basically a wrapper to Filzmoser's pcaDiagplot which colors everything according to the scheme stored in the Spectra object. Works with PCA results of either class prcomp or class princomp. Works with either classical or robust PCA results.

Usage

```
pcaDiag(spectra, pca, pcs = 3, quantile = 0.975, plot = c("OD", "SD"),
  use.sym = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp or prcomp, modified to include a character string ($method) describing the pre-processing carried out and the type of PCA performed.
pcs	As per pcaDiagplot. The number of principal components to include.
quantile	As per pcaDiagplot. The significance criteria to use as a cutoff.
plot	A character string, indicating whether to plot the score distances or orthogonal distances, or both. Options are c("OD","SD").
use.sym	logical; if true, the color scheme is change to black and symbols are used for plotting.
...	Additional parameters to be passed to the plotting functions.

Details

If both plots are desired, the output should be directed to a file rather than the screen. Otherwise, the 2nd plot overwrites the 1st in the active graphics window. Alternatively, just call the function twice, once specifying OD and once specifying SD.

Value

A list is returned as described in pcaDiagplot, so the result must be assigned or it will appear at the console. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

https://github.com/bryanhanson/ChemoSpec

See Also

pcaDiagplot in package chemometrics for the underlying function.

Examples

```
data(SrE.IR)
res <- c_pcaSpectra(SrE.IR, choice = "noscale")
temp <- pcaDiag(SrE.IR, res, pcs = 2, plot = "OD")
temp <- pcaDiag(SrE.IR, res, pcs = 2, plot = "SD")
```

---

plot2Loadings *Plot PCA Loadings from a Spectra Object Against Each Other*

Description

Plots two PCA loadings specified by the user, and labels selected (extreme) points. Typically used to determine which variables (frequencies) are co-varying, although in spectroscopy most peaks are represented by several variables and hence there is a lot of co-varying going on. Also useful to determine which variables are contributing the most to the clustering on a score plot.

Usage

```
plot2Loadings(spectra, pca, loads = c(1, 2), tol = 0.05, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	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.
loads	A vector of two integers specifying which loading vectors to plot.
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.
...	Other parameters to be passed to the plotting routines.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

See plotLoadings to plot one loading against the original variable (frequency) axis. See sPlotSpectra for a different approach.

Examples

```
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR)
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
res <- plot2Loadings(SrE.IR, pca, main = myt,
 loads = c(1,2), tol = 0.001)
```

---

plotHCA *Plot Dendrogram for Spectra Object*

Description

This function plots the results of an HCA analysis of a Spectra object. Not generally called by the user – hcaSpectra and hcaScores use it (see those pages for examples).

Usage

```
plotHCA(spectra, hclst, sub.title, use.sym, leg.loc, ...)
```

Arguments

spectra	An object of S3 class Spectra.
hclst	A hclust object.
sub.title	A character string for the subtitle.
use.sym	Logical; if true, the color scheme will be black and lower-case letters will be used to indicate group membership.
leg.loc	Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
...	Additional parameters to be passed to the plotting routines.

Value

An object of class dendrogram. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

---

plotLoadings *Plot PCA Loadings for a Spectra Object*

Description

Creates a multi-panel plot of loadings along with a reference spectrum.

Usage

```
plotLoadings(spectra, pca, loads = c(1), ref = 1, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	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.
loads	An integer vector giving the loadings to plot. More than 3 loadings creates a useless plot using the default graphics window.

ref	An integer specifying the reference spectrum to plot, which appears at the bottom of the plot.
...	Additional parameters to be passed to plotting functions.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

See plot2Loadings to plot two loadings against each other, and sPlotSpectra for an alternative approach.

Examples

```
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotLoadings(SrE.IR, pca, main = myt,
 loads = 1:2, ref = 1)
```

---

plotScores *Plot PCA Scores of a Spectra Object*

Description

Plots the requested PCA scores using the color scheme derived from a Spectra 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, pca, pcs = c(1, 2), ellipse = "none", tol = "none",
  use.sym = FALSE, leg.loc = "topright", ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp, modified to include a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions c_pcaSpectra or r_pcaSpectra were used to create pca.
pcs	A vector of two integers specifying the PCA scores to plot.
ellipse	A character vector specifying the type of ellipses to be plotted. One of c("both", "none", "cls", "rob cls specifies classical confidence ellipses, rob specifies robust confidence ellipses. An ellipse is drawn for each group in spectra$groups.
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.
use.sym	A logical; if true, the color scheme is set to black and the points plotted with symbols.
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.

References

https://github.com/bryanhanson/ChemoSpec

See Also

For other ways of displaying the results, plotScree, plotLoadings, plot2Loadings. For a 3D plot of the scores, see plotScores3D, or plotScoresRGL for an interactive version.

Examples

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

plotScores3D *3D PCA Score Plot for a Spectra Object*

Description

Creates a basic 3D plot of PCA scores from the analysis of a Spectra object, color coded according the to scheme stored in the object.

Usage

```
plotScores3D(spectra, pca, pcs = c(1:3), ellipse = TRUE, rob = FALSE,
  cl = 0.95, frac.pts.used = 0.8, view = list(y = 34, x = 10, z = 0),
  tol = 0.01, use.sym = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	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.
pcs	A vector of three integers specifying the PCA scores to plot.
ellipse	Logical indicating if confidence ellipses should be drawn.
rob	Logical; if ellipse = TRUE, indicates that robust confidence ellipses should be drawn. If FALSE, classical confidence ellipses are drawn.
cl	A number indicating the confidence interval for the ellipse.
frac.pts.used	If ellipse = TRUE and rob = TRUE, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.
view	A list of viewing transformations to be applied to the data. May contain values for x, y and z axes; keep in mind that the order of the transformations is important. For example, specifying view = list(x = 45, y = 10) produces a different view than view = list(y = 10, x = 45). The list may be as along as you like - the series of transformations representing an accumulation of tweaks to achieve the desired view.
tol	Quantile to be used to label extreme data points. Currently not used - need to fix the code!
use.sym	logical; if true, the color scheme is change to black and symbols are used for plotting.
...	Other parameters to be passed downstream.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

For a 2D plot of the scores, see plotScores. For interactive 3D plots, use plotScoresRGL.

Examples

```
data(metMUD1)
pca <- c_pcaSpectra(metMUD1, choice = "noscale")
plotScores3D(metMUD1, pca, main = "metMUD1 NMR Spectra")
```

---

plotScoresCor *Compute Confidence Ellipses*

Description

A utility function which when given a x,y data set computes both classical and robust confidence ellipses. An internal function, not generally called by the user.

Usage

```
plotScoresCor(x, quan = 1/2, alpha = 0.025)
```

Arguments

x	As per cor.plot.
quan	As per cor.plot.
alpha	As per cor.plot.

Value

A list with the following elements (a simpler version of that in the original function cor.plot):

x.cls	The x values for the classical ellipse.
y.cls	The y values for the classical ellipse.
c	The correlation value for the classical ellipse.
x.rob	The x values for the robust ellipse.
y.rob	The y values for the robust ellipse.
r	The correlation value for the robust ellipse.

Author(s)

Bryan A. Hanson, DePauw University. Derived from cor.plot.

References

https://github.com/bryanhanson/ChemoSpec

See Also

See function cor.plot in package **mvoutlier** on which this function is based.

---

plotScoresDecoration *Decorate PCA Score Plot of a Spectra Object*

Description

Utility function to carry out misc. labeling functions on the PCA score plot of a Spectra object. An internal function, not generally called by the user.

Usage

```
plotScoresDecoration(spectra, pca, pcs = c(1, 2), tol = "none")
```

Arguments

spectra	An object of S3 class Spectra.
pca	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.
pcs	A vector of two integers specifying the PCA scores to plot.
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.

Value

None. The score plot is decorated.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

plotScoresRGL *Interactive 3D Score Plot of a Spectra Object*

Description

This function uses the rgl package to create an interactive plot of PCA scores derived from a Spectra object. A title and legend can be added if desired. Classical or robust confidence ellipses may be added if desired.

Usage

```
plotScoresRGL(spectra, pca, pcs = c(1:3), ellipse = TRUE, rob = FALSE,
  cl = 0.95, frac.pts.used = 0.8, title = NULL, t.pos = NULL,
  leg.pos = NULL, lab.opts = FALSE, tol = 0.01, use.sym = FALSE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	An object of class prcomp.
pcs	A vector of three integers specifying the PCA scores to plot.
ellipse	Logical indicating if confidence ellipses should be drawn.
rob	Logical; if ellipse = TRUE, indicates that robust confidence ellipses should be drawn. If FALSE, classical confidence ellipses are drawn.
cl	A number indicating the confidence interval for the ellipse.
frac.pts.used	If ellipse = TRUE and rob = TRUE, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.
title	A character string for the plot title.
t.pos	A character selection from LETTERS[1:8] (= A through H) indicating the desired location for the title.
leg.pos	A character selection from LETTERS[1:8] (= A through H) indicating the desired location for the legend.
lab.opts	A logical indicating whether or not to display the locations where the title and legend can be placed. These locations are the corners of a cube surrounding the data.
tol	Quantile to be used to label extreme data points.
use.sym	logical; if true, the color scheme is changed to black and symbols are used for plotting.
...	Additional parameters to pass downstream, generally to the plotting routines.

Details

If you intend to make a hard copy of your plot, use `lab.opts = TRUE` until you have found a good view of your data. Then note corners of the cube where the title and legend won't interfere with viewing the data, and use these as arguments for `t.pos` and `leg.pos`, and add `title`. Adjust as necessary, then turn off label display using `lab.opts = FALSE`. Back at the console, use `> rgl.snapshot("file_name.png")` to create the hardcopy.

Value

None. Side effect is a plot

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

Other functions in ChemoSpec that plot PCA scores are: `plotScores` (2D version), and `plotScores3D` (uses `lattice` graphics).

Examples

```
## Not run:

data(metMUD1)
pca <- c_pcaSpectra(metMUD1, choice = "autoscale")
plotScoresRGL(metMUD1, pca, title = "metMUD1 NMR Spectra",
  leg.pos = "A", t.pos = "B")

## End(Not run)
```

---

plotScree *Scree Plots of PCA Results for a Spectra Object*

Description

Functions to draw a traditional scree plot or an alternative that is perhaps more useful. These illustrate the importance of the components in a PCA analysis.

Usage

```
plotScree(pca, ...)

plotScree2(pca, ...)
```

Arguments

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

Details

If you add $method to the PCA results from other packages, this will plot a scree plot for any PCA results, not just those from Spectra objects.

Value

None. Side effect is a plot.

Functions

- plotScree: Traditional scree plot
- plotScree2: Alternate scree 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

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(metMUD1)
pca <- c_pcaSpectra(metMUD1)
plotScree(pca, main = "metMUD1 NMR Data")
plotScree2(pca, main = "metMUD1 NMR Data")
```

plotSpectra *Plot Spectra Object*

Description

Plots the spectra stored in a Spectra object. One may choose which spectra to plot, and the x range to plot. Spectra may be plotted offset or stacked. The vertical scale is controlled by a combination of several parameters.

Usage

```
plotSpectra(spectra, which = c(1), yrange = range(spectra$data),
  offset = 0, amplify = 1, lab.pos = mean(spectra$freq),
  showGrid = TRUE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
which	An integer vector specifying which spectra to plot, and the order.
yrange	A vector giving the limits of the y axis desired, for instance c(0, 15). This parameter depends upon the range of values in the stored spectra and defaults to the height of the largest peak in the data set. Interacts with the next two arguments, as well as the number of spectra to be plotted as given in which. Trial and error is used to adjust all these arguments to produce the desired plot.
offset	A number specifying the vertical offset between spectra if more than one is plotted. Set to 0.0 for a stacked plot.
amplify	A number specifying an amplification factor to be applied to all spectra. Useful for magnifying spectra so small features show up (though large peaks will then be clipped, unless you zoom on the x axis).
lab.pos	A number giving the location for the identifying label. Generally, pick an area that is clear in all spectra plotted. If no label is desired, give lab.pos outside the plotted x range.
showGrid	Logical. Places light gray vertical lines at each tick mark if TRUE.
...	Additional parameters to be passed to plotting functions.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

plotSpectraJS for the interactive version.

Examples

```
data(metMUD1)
plotSpectra(metMUD1, main = "metMUD1 NMR Data",
 which = c(10, 11), yrange = c(0,1.5),
 offset = 0.06, amplify = 10, lab.pos = 0.5)
```

---

plotSpectraDist *Plot the Distance Between Spectra in a Spectra Object*

Description

This function plots the distance between a reference spectrum and all other spectra in a Spectra object. Distance can be defined in a number of ways (see Arguments).

Usage

```
plotSpectraDist(spectra, method = "pearson", ref = 1, labels = TRUE, ...)
```

Arguments

spectra	An object of S3 class Spectra.
method	Character. Any method acceptable to rowDist.
ref	Integer. The spectrum to be used as a reference.
labels	Logical. Shall the points be labeled?
...	Plot parameters to be passed to the plotting routines.

Value

A data frame containing the data plotted (sample names, sample colors, distances).

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.NMR)
txt1 <- paste("Distance from", SrE.NMR$names[1]) # capture before padding
txt2 <- paste("Rank Distance from", SrE.NMR$names[1])
SrE.NMR$names <- paste("  ", SrE.NMR$names, sep = "") # pad the names for better appearance
temp <- plotSpectraDist(SrE.NMR, xlab = txt2, ylab = txt1, main = txt1,
  xlim = c(1,16), ylim = c(0, 0.3), srt = 90)
```

---

plotSpectraJS *Plot a Spectra Object Interactively*

Description

This function uses the d3.js JavaScript library by Mike Bostock to plot a Spectra object interactively. This is most useful for data exploration. For high quality plots, consider plotSpectra.

Usage

```
plotSpectraJS(spectra, which = NULL, browser = NULL, minify = TRUE)
```

Arguments

spectra	An object of S3 class Spectra to be checked.
which	Integer. If not NULL, specifies by number which spectra to plot. If greater control is needed, use removeSample which is more flexible before calling this function.
browser	Character. Something that will make sense to your OS. Only necessary if you want to override your system specified browser as understood by R. See below for further details.
minify	Logical. Shall the JavaScript be minified? This improves performance. However, it requires package js which in turn requires package V8. The latter is not available on all platforms. Details may be available at https://github.com/jeroenooms/v8

Details

The spectral data are incorporated into the web page. Keep in mind that very large data sets, like NMR spectra with 32K points, will bog down the browser. In these cases, you may need to limit the number of samples in passed to this function. See removeSample or use argument which.

Value

None; side effect is an interactive web page. The temporary directory containing the files that drive the web page is written to the console in case you wish to use those files. This directory is deleted when you quit R. If you wish to read the file, don't minify the code, it will be unreadable.

Browser Choice

The browser is called by browseURL, which in turn uses options("browser"). Exactly how this is handled is OS dependent.

RStudio Viewer

If browser is NULL, you are using RStudio, and a viewer is specified, this will be called. You can stop this by with options(viewer = NULL).

Browser Choice (Mac)

On a Mac, the default browser is called by /bin/sh/open which in turn looks at which browser you have set in the system settings. You can override your default with browser = "/usr/bin/open -a 'Google Chrome'" for example.

Browser Choice & Performance

You can check the performance of your browser at peacekeeper.futuremark.com The most relevant score is the rendering category.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

plotSpectra for non-interactive plotting. Details about d3.js are at www.d3js.org.

Examples

```
if (interactive()) {
  require("jsonlite")
  require("js")
  data(metMUD2)
  plotSpectraJS(metMUD2)
}
```

q2rPCA *Conversion Between PCA Classes*

Description

Utility to convert objects of S3 class prcomp (Q-mode PCA) to objects of S3 class princomp (R-mode PCA) or *vice-versa*. An internal function, not generally called by the user.

Usage

```
q2rPCA(x)
```

Arguments

x	An object of either class prcomp or class princomp. It will be converted to a form that can be used by functions expecting either class.

Details

In the conversion, the necessary list elements are added; the old elements are not deleted (and user added list elements are not affected). To indicate this, the class attribute is updated to include class conPCA. The new object can then be used by functions expecting either class prcomp or princomp. For details of the structure of prcomp or princomp, see their respective help pages.

Value

A list of class conPCA. Note that the order of the elements will vary depending upon the direction of conversion.

loadings	The loadings from princomp, or a copy of the rotations from prcomp.
scores	The scores from princomp, or a copy of the x values from prcomp.
call	The call. Objects of class prcomp do not store the original call, so a place holder is used. Otherwise the unchanged call from princomp.
n.obs	The number of observations from princomp, or computed from the 1st dimension of x in prcomp.
class	conPCA is pre-pended to the existing class.
sdev	Unchanged from original.
center	Unchanged from original.
scale	Unchanged from original.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

prcomp, princomp

---

removeFreq *Remove Frequencies from a Spectra Object*

Description

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

Usage

```
removeFreq(spectra, rem.freq)
```

Arguments

spectra	An object of S3 class Spectra from which to remove selected frequencies.
rem.freq	A valid R statement describing the frequencies to be removed. This must comply with Comparison and Logic. See the examples below for common usage.

Details

rem.freq can be any valid R statement that leads to a vector of logicals. In the examples below, the | and & operators seem backward in a sense, but R evaluates them one at a time and combines the result to give the required output.

Value

An object of S3 class Spectra.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
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)
check4Gaps(newIR$freq, newIR$data[1,], plot = TRUE)
```

`removeGroup` *Remove Groups or Samples from a Spectra Object*

Description

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

Usage

```
removeGroup(spectra, rem.group)

removeSample(spectra, rem.sam)
```

Arguments

`spectra`	An object of S3 class `Spectra`.
`rem.group`	A character vector giving the groups to be removed.
`rem.sam`	Either an integer vector specifying the samples to be removed, or a character vector giving the sample names to be removed.

Details

Both functions 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. `rem.group` also uses grep. 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

A modified object of S3 class `Spectra`.

Functions

- `removeGroup`: Remove groups from a `Spectra` object
- `removeSample`: Remove samples from a `Spectra` object

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

`removeFreq` to remove selected frequencies.

Examples

```
data(metMUD1)

# removeGroup
sumSpectra(metMUD1)
trmt <- removeGroup(metMUD1, rem.group = "Cntrl")
sumSpectra(trmt)

# removeSample
# Removes the 20th spectrum/sample:
new1 <- removeSample(metMUD1, rem.sam = 20)

# Removes one spectrum/sample with this exact name:
new2 <- removeSample(metMUD1, rem.sam = "metMUD1_20")
```

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.

r_pcaSpectra *Robust PCA of a Spectra Object*

Description

A wrapper which carries out robust PCA analysis on a Spectra object. The data are row- and column-centered, and the user can select various options for scaling.

Usage

```
r_pcaSpectra(spectra, choice = "noscale")
```

Arguments

spectra	An object of S3 class Spectra.
choice	A character vector describing the type of scaling to be carried out. One of c("noscale", "mad").

Value

An object of classes conPCA and princomp (see q2rPCA). It includes a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (it appears on plots which you might make).

Author(s)

Bryan A. Hanson, DePauw University.

References

K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

https://github.com/bryanhanson/ChemoSpec

See Also

See PCAgrid on which this function is based. For the classical version, see c_pcaSpectra.

For displaying the results, plotScree, plotScores, plotLoadings, plot2Loadings, sPlotSpectra, plotScores3D, plotScoresRGL.

Examples

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

`sampleDistSpectra` *Compute the Distance Between Samples in a Spectra Object*

Description

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

Usage

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

Arguments

`spectra`	An object of S3 class `Spectra`.
`method`	Character. A string giving the distance method. See `rowDist` for options.
`plot`	Logical. Shall a level plot be made?
`...`	Arguments to be passed to the plotting function.

Value

A numeric matrix giving the correlation coefficients.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

The sample distances can be used to cluster the samples. See for example `hcaSpectra`.

Examples

```
require("lattice")
data(SrE.IR)
M <- sampleDistSpectra(SrE.IR, method = "cosine",
  main = "SrE.IR Spectral Angle Between Samples")
```

seX *Functions to Compute Measures of Central Tendency and Spread. seX!*

Description

These functions compute various measures of central tendency and spread. These functions return a vector containing the measure of central tendency, as well as that measure +/- the requested spread. seX is a little different from the others in that it simply returns the standard error of x, hence seX. Haven't we always needed a function for seX?

Usage

```
seX(x)

seXy(x)

seXy95(x)

seXyIqr(x)

seXyMad(x)
```

Arguments

x	A vector of numeric values whose measure of central tendency and spread are to be computed.

Details

These functions include na.omit.

Value

For all but seX, a vector of 3 numeric values, giving the measure of central tendency, that measure + the spread, and that measure - the spread. For seX, a single value giving the standard error of x.

Functions

- seX: standard error of x
- seXy: mean +/- the standard error
- seXy95: mean +/- the standard error at 95% conf. interval
- seXyIqr: median +/- the 1st and 3rd quantile
- seXyMad: median +/- median absolute deviation

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
x <- rnorm(100)
seX(x)
seXy(x)
seXy95(x)
seXyMad(x)
seXyIqr(x)
```

`sgfSpectra` *Apply Savitzky-Golay filters to a Spectra object*

Description

This function is a simple wrapper around the function `sgolayfilt`. It allows one to apply Savitzky-Golay filters to a `Spectra` object in a convenient way.

Usage

```
sgfSpectra(spectra, m = 0, ...)
```

Arguments

`spectra`	An object of S3 class `Spectra` to be checked.
`m`	The desired m-th derivative. `m = 0` smooths the data (i.ei a rolling average), `m = 1` gives the first derivative etc.
`...`	Other parameters to be passed to `sgolayfilt`.

Value

A object of class `Spectra`.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
myt1 <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
myt2 <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra~(Smoothed)))

par(mfrow = c(2, 1))
plotSpectra(SrE.IR, xlim = c(1900, 2100), yrange = c(0, 0.05), main = myt1)
temp <- sgfSpectra(SrE.IR)
plotSpectra(temp, xlim = c(1900, 2100), yrange = c(0, 0.05), main = myt2)
par(mfrow = c(1, 1))
```

---

shrinkLeaf *Shrink the Leaves of a Dendrogram Based on a Spectra Object*

Description

This function shrinks the size of leaves of a dendrogram object. The code was taken from the help files. An internal function, not generally called by the user.

Usage

```
shrinkLeaf(n, spectra)
```

Arguments

n	A node in a dendrogram object.
spectra	An object of S3 class Spectra.

Value

Returns a node with the label size properties set.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Spectra *Spectra Objects*

Description

In ChemoSpec, spectral data sets are stored in an S3 class called Spectra, which contains a variety of information in addition to the spectra themselves. Spectra objects are created by files2SpectraObject or matrix2SpectraObject.

Structure

The structure of a Spectra object is a list of 7 elements and an attribute as follows:

element	*type*	*description*
$freq	num	A common frequency (or wavelength) axis for all the spectra.
$data	num	The intensities for the spectra. A matrix of dimension no. samples x no. frequency points.
$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	chr	The colors for each sample; length must be no. samples. Groups and colors correspond.
$sym	integer	As for colors, but symbols for plotting (if b/w is desired).
$alt.sym	chr	Lower-case letters as alternate symbols for plotting.
$unit	chr	Two entries, the first giving the x axis unit, the second the y axis unit.
$desc	chr	A character string describing the data set. This appears on plots and therefore should probably be kept to 40 characters or less.
- attr	chr "Spectra"	The S3 class designation.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

sumSpectra to summarize a Spectra object. sumGroups to summarize group membership of a Spectra object. chkSpectra to verify the integrity of a Spectra object. colorSymbol for a discussion of color options.

splitSpectraGroups *Create New Groups from an Existing Spectra Object*

Description

This function takes an existing Spectra object and uses your instructions to split the existing spectra$groups into new groups. The new groups are added to the existing Spectra object (a list) as new elements. This allows one to use different combinations of factors than were originally encoded in the Spectra object. The option also exists to replace the color scheme with one which corresponds to the new factors.

Usage

```
splitSpectraGroups(spectra, inst = NULL, rep.cols = NULL, ...)
```

Arguments

spectra	An object of S3 class Spectra.
inst	A list giving the name of the new element to be created from a set of target strings given in a character vector. See the example for the syntax.
rep.cols	Optional. A vector giving new colors which correspond to the levels of inst. Only possible if inst has only one element, as the possible combinations of levels and colors may get complicated.
...	Additional arguments to be passed downstream. Currently not used.

Details

The items in the character vector are grepped among the existing spectra$groups entries; when found, they are placed in a new element of Spectra. In the example, all spectra$groups entries containing "G" are coded as "G" in a new element called spectra$env, and any entries containing "T" are handled likewise. This amounts to a sort of recoding of factors (the example demonstrates this). Every entry in spectra$groups should be matched by one of the entries in the character vector. If not, you will get <NA> entries. Also, if the targets in the character vector are not unique, your results will reflect the order of the levels. Since this is a grep process, you can pass any valid grep string as the target.

If rep.cols is provided, these colors are mapped one for one onto the levels of the the first element of inst. This provides a different means of changing the sample color encoding than conColScheme.

Value

An object of S3 class Spectra, modified to have additional elements as specified by inst.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

conColScheme.

Examples

```
data(metMUD2)
levels(metMUD2$groups) # original factor encoding

# Split those original levels into 2 new ones (re-code them)
new.grps <- list(geneBb = c("B", "b"), geneCc = c("C", "c"))
res <- splitSpectraGroups(metMUD2, new.grps)
str(res) # note two new elements, "geneBb" and "geneCc"
sumSpectra(res) # reports on extra elements

# Note that if you want to use a newly created group in
# plotScores and other functions to drive the color scheme
# and labeling, you'll have to update the groups element:
res$groups <- as.factor(paste(res$geneBb, res$geneCc, sep = ""))
```

---

sPlotSpectra *s-Plot of Spectra Data (Post PCA)*

Description

Produces a scatter plot of the correlation of the variables against their covariance for a chosen principal component. It allows visual identification of variables driving the separation and thus is a useful adjunct to traditional loading plots.

Usage

```
sPlotSpectra(spectra, pca, pc = 1, tol = 0.05, ...)
```

Arguments

spectra	An object of S3 class Spectra.
pca	The result of a pca calculation on Spectra (i.e. the output from c_pcaSpectra or r_pcaSpectra).
pc	An integer specifying the desired pc plot.
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.
...	Additional parameters to be passed to plotting functions.

Value

A data frame containing the frequency, covariance and correlation of the selected pc for the Spectra object. A plot of the correlation vs. covariance is created.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University.

References

Wiklund, Johansson, Sjostrom, Mellerowicz, Edlund, Shockcor, Gottfries, Moritz, and Trygg. "Visualization of GC/TOF-MS-Based Metabololomics Data for Identification of Biochemically Interesting Compounds Usings OPLS Class Models" Analytical Chemistry Vol.80 no.1 pgs. 115-122 (2008).

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
IR.pca <- c_pcaSpectra(SrE.IR)
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
splot <- sPlotSpectra(spectra = SrE.IR, pca = IR.pca, pc = 1, tol = 0.001,
  main = myt)
```

SrE.IR — *IR and NMR Spectra of Serenoa repens (Saw Palmetto) Oil Extracts and Reference Oils*

Description

A collection of 14 IR and NMR spectra of essential oil extracted from the palm *Serenoa repens* or Saw Palmetto, which is commonly used to treat BPH in men. The 14 spectra are of different retail samples, and are divided into two categories based upon the label description: adSrE, adulterated extract, and pSrE, pure extract. The adulterated samples typically have olive oil added to them, which is inactive towards BPH. There are two additional spectra included as references/outliers: evening primrose oil, labeled EPO in the data set, and olive oil, labeled OO. These latter two oils are mixtures of triglycerides for the most part, while the SrE samples are largely fatty acids. As a result, the spectra of these two groups are subtly different.

Format

The data are stored as a Spectra object.

Source

IR data collected in the author's laboratory. NMR data collected at Purdue University with the generosity and assistance of Prof. Dan Raftery and Mr. Tao Ye.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
sumSpectra(SrE.IR)
data(SrE.NMR)
sumSpectra(SrE.NMR)
```

---

sumGroups *Summarize the Group Parameters of a Spectra Object*

Description

This function summarizes the group membership and descriptive parameters of a Spectra object.

Usage

```
sumGroups(spectra)
```

Arguments

spectra	An object of S3 class Spectra whose group membership information is desired.

Value

A data frame as follows. Note that if there are groups with no members (due to previous use of removeSample), 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.
alt.symbol	The alternative symbol, a lower-case letter, assigned to the group.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

See Also

For a discussion of general issues of color, see colorSymbol. To summarize the entire object, sumSpectra.

Examples

```
data(metMUD1)
sumGroups(metMUD1)
```

---

sumSpectra *Summarize a Spectra Object*

Description

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

Usage

```
sumSpectra(spectra, ...)
```

Arguments

spectra	An object of S3 class Spectra.
...	Arguments to be passed downstream.

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. If sumSpectra thinks there is a gap between every data point, add the argument tol = xx which will pass through to check4Gaps and alleviate this problem (which has to do with rounding when subtracting two adjacent frequency values). The Spectra 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, perhaps a plot as well.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(metMUD1)
sumSpectra(metMUD1)
```

---

surveySpectra *Plot Measures of Central Tendency and Spread for a Spectra Object*

Description

Compute and plot various measures of central tendency and spread for a Spectra object. Several different measures/spreads are available. These are useful as an overview of where a data set varies the most.

Usage

```
surveySpectra(spectra, method = c("sd", "sem", "sem95", "mad", "iqr"),
  by.gr = TRUE, ...)

surveySpectra2(spectra, method = c("sd", "sem", "sem95", "mad", "iqr"),
  lab.pos = 0.9 * max(spectra$freq), ...)
```

Arguments

spectra	An object of S3 class Spectra to be analyzed.
method	Character. One of c("sd", "sem", "sem95", "mad","iqr").
by.gr	Logical, indicating if the analysis is to be done by group or not. Applies to surveySpectra only.
...	Additional parameters to be passed to the plotting routines.
lab.pos	Numeric, giving the frequency where the label should be drawn. Applies to surveySpectra2 only.

Details

For surveySpectra the method choice works as follows: sd plots the mean spectrum +/- the standard deviation, sem plots the mean spectrum +/- the standard error of the mean, sem95 plots the mean spectrum +/- the standard error at the 95 percent confidence interval, mad plots the median spectrum +/- the median absolute deviation, and finally, iqr plots the median spectrum + the upper hinge and - the lower hinge.

For surveySpectra2, the spectra are mean centered and plotted. Below that, the relative summary statistic is plotted, offset, but on the same scale.

Value

None; side effect is a plot

Functions

- `surveySpectra`: Spectral survey emphasizing mean or median spectrum, optionally by group.
- `surveySpectra2`: Spectral survey emphasizing variation among spectra.

Author(s)

Bryan A. Hanson, DePauw University.

References

https://github.com/bryanhanson/ChemoSpec

Examples

```
data(SrE.IR)
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(Extract~IR~Spectra))
surveySpectra(SrE.IR, method = "iqr", main = myt)
surveySpectra2(SrE.IR, method = "iqr", main = myt)
```

Index