Package 'ChemoSpec'

July 2, 2014

Type Package

Title Exploratory Chemometrics for Spectroscopy

Version 2.0-2

Date 2014-05-31

Author Bryan A. Hanson DePauw University, Greencastle Indiana USA

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

Description

A collection of functions for plotting spectra (NMR, IR etc) and carrying out various forms of top-down exploratory data analysis, such as HCA, PCA and model-based clustering. The design allows comparison of data from samples which fall into groups such as treatment vs. control. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed to be very user friendly and suitable for people with limited background in R.

License GPL-3

Depends R (>= 3.0)

Imports pls, amap, chemometrics, robustbase, RColorBrewer, plyr,pcaPP, mvtnorm, mvoutlier, grid, rgl, R.utils, mclust, MASS,baseline, IDPmisc, gsubfn, lattice, seriation

URL http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

ByteCompile TRUE

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

VignetteBuilder knitr

Suggests mvbutils, sna, knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-31 15:55:29

R topics documented:

`ChemoSpec-package` *Exploratory Chemometrics for Spectroscopy*

Description

A collection of functions for plotting spectra (NMR, IR etc) and carrying out various forms of top-down exploratory data analysis, such as HCA, PCA and model-based clustering. The design permits comparison of data from samples which fall into groups such as treatment vs. control. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed to be very user friendly and suitable for people with limited background in R. A vignette illustrating typical operations is available.

Details

Package:	ChemoSpec
Type:	Package
Version:	2.0-2
Date:	2014-05-31
License:	GPL-3

Author(s)

Bryan A. Hanson, DePauw University, Greencastle Indiana USA

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

---

aovPCA *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.)

Usage

```
aovPCA(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.

Details

ANOVA-PCA with only one factor is the same as standard PCA and gives no additional separation.

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. <hanson@depauw.edu>

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.

See Also

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

Examples

```
data(CuticleIR)
# Create new groups from existing groups
# These are used as the factors
n.groups <-list(genotype = c("G", "T"), treatment = c("C", "E"))
NewIR <- splitSpectraGroups(CuticleIR, n.groups)
# run aovPCA
mats <-aovPCA(NewIR, fac = c("genotype", "treatment"))
apca1 <- aovPCAscores(NewIR, mats, plot = 1, main = "aovPCA: Genotype")
apca2 <- aovPCAscores(NewIR, mats, plot = 2, main = "aovPCA: Treatment")
apca3 <- aovPCAscores(NewIR, mats, plot = 3, main = "aovPCA: Genotype x Treatment")
apca5 <- aovPCAloadings(spectra = NewIR, LM = mats, pca = apca1, main = "aovPCA: Genotype Loadings")
```

---

aovPCAloadings *Plot aovPCA 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 aovPCA
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. <hanson@depauw.edu>

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.

See Also

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

---

aovPCAscores *Conduct PCA and plot aovPCA Scores of a Spectra Object*

Description

Uses the results from aovPCA to conduct PCA and plot the scores.

Usage

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

Arguments

spectra	An object of S3 class "Spectra".
LM	List of matrices created by aovPCA.
plot	An integer specifying which scores to plot.
type	Either classical ("cls") or robust ("rob"); Results in either classPCA or robPCA being called on the Spectra object.
choice	The type of scaling to be performed. See classPCA and robPCA for details.
...	Additional parameters to be passed to plotScores.

Details

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.

Value

Returns the PCA results, and creates the requested plot.

Author(s)

Matthew J. Keinsley and Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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.

See Also

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

---

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. <hanson@depauw.edu>

See Also

aovPCA 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)
```

baselineSpec *Carry out baseline correction on a Spectra object*

Description

This function is a simple wrapper to functions in package baseline which carries out a variety of baseline correction routines.

Usage

```
baselineSpec(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.

Details

In the plots, 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).

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(SrE.IR)
temp <- baselineSpec(SrE.IR, int = FALSE, method = "rfbaseline")
par(mfrow = c(1,1)) # needed to cancel 2 panel plot
## Not run:
baselineSpec(SrE.IR)
par(mfrow = c(1,1)) # needed to cancel 2 panel plot

## End(Not run)
```

---

binBuck *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

```
binBuck(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 binBuck).

Value

An object of S3 class "Spectra".

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
sumSpectra(CuticleIR)
res <- binBuck(CuticleIR, bin.ratio = 4)
sumSpectra(res)
```

`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.

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 `binBuck` 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

As appropriate, a data.frame containing 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
x <- seq(0, 1000, length.out = 3000); y <- rnorm(3000)
res <- binData(x, y)
length(res$mean.x) # will be half of the original length
# Now try it with bin.ratio that does not divide into 3000
x <- seq(0, 1000, length.out = 3000); y <- rnorm(3000)
res <- binData(x, y, bin.ratio = 7)
length(res$mean.x)
```

---

calcSN *Calculate the signal-to-noise ratio of a peak in a spectrum.*

Description

This is very simple function to calculate the signal-to-noise ratio of a peak in a spectrum. It is intended for processing NMR spectra in conjunction with findTMS. It is not extensively tested. The SN value returned is just one (simple) way of computing SN. You probably shouldn't trust it.

Usage

```
calcSN(x, span, idx, debug = FALSE)
```

Arguments

x	Vector of intensities of the spectrum. The frequency aspect is handled simply as the index along the intensity vector.
span	Integer; the number of data points to use as a window on either side of a peak when calculating the SN. See the code for exactly how this window is created. Larger values find fewer peaks.
idx	Integer; the index giving the location of the peak to have its SN computed.
debug	Logical; Spew forth information, more than you want.

Details

Warnings are issued when peaks are near the edges of the spectrum. The algorithm computes the SN for these peaks differently, and possibly inaccurately. You are more likely to get the warning with higher values of span. See the example.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
# Function to pick peaks:
pp <- function (x, span) {
# Borrowed from ChemometricsWithR by R. Wehrens
span.width <- span * 2 + 1
loc.max <- span.width + 1 - apply(embed(x, span.width), 1,
        which.max)
loc.max[loc.max == 1 | loc.max == span.width] <- NA
pks <- loc.max + 0:(length(loc.max) - 1)
unique(pks[!is.na(pks)])
}
# Some test data:
set.seed(99)
vec <- abs(rnorm(100))
vec[12] <- 10
vec[75] <- 31 # create two larger peaks

# Now demonstrate:
span <- 20
ppVec <- pp(vec, span)
for (i in 1:length(ppVec)) {
tmp <- calcSN(vec, span, ppVec[i])
print(tmp) # SN of each peak found reported
}
plot(vec, type = "l")
abline(v = ppVec, col = "red") # Shows peaks identified
# Try repeating with span = 10
```

---

check4Gaps *Check for Missing Values (Gaps)*

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. Not normally called directly by the user, but may be (see examples). 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 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.
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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
x <- seq(from = 5, to = 12, by = 0.1)
remove <- 40:45; x <- x[-remove]
check4Gaps(x) # really simple
gaps <- check4Gaps(x) # save the result for later use
data(CuticleIR) # has a gap, let's find it and display it
check4Gaps(CuticleIR$freq, CuticleIR$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 instance of an S3 object) is internally consistent. Rather than directly manipulating a "Spectra" object, one should manipulate it via removeFreq or removeSample. Should not see much direct use by users.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
chkSpectra(CuticleIR, confirm = TRUE) # OK
# What's next works, but is the wrong way to manipulate a "Spectra" object.
# Use removeSample instead.
remove <- c(20:40)
CuticleIR$freq <- CuticleIR$freq[-remove]
## Not run: chkSpectra(CuticleIR, confirm = TRUE) # not OK, you didn't listen to me!
```

classPCA *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 aovPCA series of functions. Centering the data should always be done in PCA and it is the default here.

Usage

```
classPCA(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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html
K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

See Also

prcomp for the underlying function, robPCA for analogous robust PCA calculations.
For displaying the results, plotScree, plotScores, plotLoadings, plot2Loadings.

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotScores(SrE.IR, main = myt, results, pcs = c(1,2),
ellipse = "rob", tol = 0.05)
```

`colLeaf` *Color the Leaves of a Dendrogram*

Description

This function colors the leaves of a dendrogram object. The code was taken from the help files. Not for end-user use.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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 for your samples.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

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

Examples

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

---

coordProjCS *Modified Version of coordProj from mclust*

Description

This is a modified version of the function coordProj from 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.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

CuticleIR *IR Spectra of Plant Cuticles by Genotype and Treatment*

Description

IR spectra obtained by ATR sampling on leaves of *Portulaca oleracea* (common purslane). There are 157 spectra, divided into four groups from a G x E experiment. Two genotypes were studied, golden (G), and tall green (T). Two temperature regimes were employed, experimental (E, 35C) and control (C, 22C). Sample name GC10 means golden phenotype, control treatment, plant no. 10.

Usage

```
data(CuticleIR)
```

Format

List of 9
$ freq : num [1:1242] 501 503 505 507 509 ...
$ data : num [1:157, 1:1242] 0.205 0.247 0.219 0.203 0.234 ...
$ names : chr [1:157] "GC10" "GC11" "GC12" "GC14" ...
$ groups : Factor w/ 4 levels "GC","GE","TC",..: 1 1 1 1 1 1 1 1 1 1 ...
$ colors : chr [1:157] "dodgerblue4" "dodgerblue4" "dodgerblue4" "dodgerblue4" ...
$ sym : num [1:157] 2 2 2 2 2 2 2 2 2 2 ...
$ alt.sym: chr [1:157] "b" "b" "b" "b" ...
$ unit : chr [1:2] "Wavenumbers" "Absorbance"
$ desc : chr "Kelly's Complete IR Data Set, Summer 2009"
- attr(*, "class")= chr "Spectra"

Details

Noisy regions at the extremes of the frequency range have been removed. The region from 1800 - 2500 wavenumbers was also removed as it is uninformative.

Source

Data obtained by Kelly Summers at DePauw University, Summer 2009.

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
chkSpectra(CuticleIR)
sumSpectra(CuticleIR) # this also runs chkSpectra() before doing the summary
```

---

files2SpectraObject *Merge Files in a Directory into a Spectra Object*

Description

This function will read all files of a given type in a directory, and use the file names to construct group membership and assign colors and symbols. All the data is placed into an object of S3 class "Spectra". This is the only way to create a "Spectra" object automatically.

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",
format = "csv", alignTMS = FALSE,
out.file = "mydata", debug = FALSE, ...)
```

Arguments

gr.crit	Group Criteria. A vector of character strings which will be searched for among the file names in order to assign an individual spectrum/sample to group membership. Warnings are issued 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. See Details for some nuances.
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. 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.
format	A character string giving the format of the files to be processed. Default is csv for US-style csv files. Alternatively, you can specify csv2 for EU-style csv files, dx for JCAMP-DX files, or Btxt for Bruker NMR text files.
alignTMS	Logical indicating if we should try to align the TMS (or TSP) peak in proton NMR spectra (applies to format = "Btxt" only). See Details.
out.file	A file name acceptable to the save function. The completed object of S3 class "Spectra" will be written to this file.
debug	Logical; set to TRUE for troubleshooting when using format = "Btxt" or "dx".
...	Other arguments to be passed downstream (At times you might want to pass alternate values of span, sn, and thres to findTMS and related functions).

Details

The linking of groups with colors is handled by groupNcolor.

The matching of gr.crit against the sample file names is done one at a time, in order. This means that 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.

`files2SpectraObject` acts on the files in the current working directory. If `format = "csv"` these should be .csv files with the first column containing the frequency values and the second column containing the intensity values. The columns should be unlabeled. The frequency column is assumed to be the same in all .csv files. If `format = "dx"` or `format = "Btxt"`, then the corresponding file type will be processed (consider setting `debug = TRUE` for these formats). See `readJDX` and `readBrukerTxt` for limitations (there are many options with these formats, especially JCAMP, and most are untested).

If `format = "Btxt"` and `alignTMS = TRUE`, the function will try to find the TMS peak and align the spectra on it. Also, spectra of different chemical shift ranges are allowed for this format. In this case, the spectra will first be aligned on TMS and then the set of spectra will be trimmed so that there are no NA's in `Spectra$data`. Warnings are given as this is done. This is experimental so please check your results carefully! Please feel free to submit data sets that give trouble and I can see if I can improve the processing.

There should be no other files of the given format (extension) in the directory except those containing the data to be processed by `files2SpectraObject`, as all files with that format in the directory will be processed.

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)`, found in package `R.utils`.

Warning

Files whose names are not matched using gr.crit are still incorporated into the `"Spectra"` object, but they are not assigned a group or color and therefore don't plot, though they do take up space in a plot!

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

findTMS *Find the TMS/TSP Peak in an NMR Spectrum.*

Description

This is a simple function to find the TMS or TSP peak in proton NMR spectrum. It simply finds the rightmost large peak in the spectrum.

Usage

```
findTMS(x, span, sn, thresh = 0.2, debug = FALSE)
```

Arguments

x	Vector of intensities of the spectrum. The frequency aspect is handled simply as the index along the intensity vector.
span	Integer; the number of data points to use as a window on either side of a peak when calculating the SN. See the code for exactly how this window is created. Larger values find fewer peaks.
sn	Numeric; the minimum value of the signal-to-noise ratio for detecting a peak.
thresh	Numeric; a value in 0...1 which gives the quantile of peaks in the entire spectrum which should be kept on the short list for finding the TMS peak.
debug	Logical; if TRUE write progress and findings to screen.

Details

The rightmost peak is sought, so the spectrum must be presented in the usual high ppm –> low ppm format. Calls calcSN which will issue warnings when peaks are near the edges of the spectrum.

Value

The index of the peak believed to be the TMS peak.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
# Some test data:
set.seed(99)
vec <- abs(rnorm(100))
vec[12] <- 10
vec[75] <- 31 # create two larger peaks

ans <- findTMS(vec, 20, 5)
ans # index as set above (75)
plot(vec, type = "l")
abline(v = ans, col = "red")
```

---

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. 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. Not intended for users.

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"`. 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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.

Usage

```
hcaScores(spectra, pca, scores = c(1:5), c.method = "complete",
d.method = "euclidean", 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 classPCA or robPCA 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.
...	Additional parameters to be passed to the plotting functions.

Value

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

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
hcaScores(SrE.IR,  results, 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. All methods for computing the cluster distances are available.

Usage

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

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.
...	Other parameters to be passed to the plotting functions.

Value

An object of class dendrogram. The side effect is a plot.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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))
hcaSpectra(SrE.IR, main = myt)
```

---

hmapSpectra *Create a Seriated Heat Map Comparing Samples and Spectral Data 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 hierchical cluster analysis and then re-ordering the clusters in a coordinated way across each dimension. See the reference.

Usage

```
hmapSpectra(spectra, no.col = 5, cexRow = 1, cexCol = 1, ...)
```

Arguments

spectra	An object of S3 class "Spectra".
no.col	The number of colors (levels) to use in the heat map. Maximum of 9, generated by RColorBrewer.
cexRow	Scale factor for the row labels.
cexCol	Scale factor for the column labels.
...	Additional arguments to be passed downstream.

Value

None. Side effect is a plot.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

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

Examples

```
data(SrE.IR)
# remove noisy, uninteresting region:
newIR <- removeFreq(SrE.IR, rem.freq =
SrE.IR$freq > 1800 & SrE.IR$freq < 2500)
# now make the heat map:
myt <- expression(italic(S.)~italic(repens)~IR~Spectra)
hmapSpectra(newIR, title = myt, t.pos = c(0.5, 0.5, 0.5),
cexRow = 0.5, cexCol = 0.1, no.col = 9)
```

---

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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(CuticleIR)
res <- classPCA(CuticleIR)
hypTestScores(CuticleIR, res, fac = "groups")
# from splitSpectraGroups:
new.groups <- list(gen = c("G", "T"), trt = c("C", "E"))
new.CuticleIR <- splitSpectraGroups(CuticleIR, new.groups)
hypTestScores(new.CuticleIR, res, fac = c("trt", "gen"))
```

---

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.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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. Never called by the user. Uses base graphics.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

`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).

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 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 `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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

LoopThruSpectra *Display the spectra in a Spectra object one at a time*

Description

Plots each spectrum in a Spectra object one at a time (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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
## Not run:
data(CuticleIR)
LoopThruSpectra(CuticleIR)

## 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 generate points, which if plotted as tiny points, create a sort of transparent surface.

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.

Called by

`plotScoresRGL`, `mclust3dSpectra`, `mclust3D`.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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.

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.

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.

Value

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

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
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))
## Not run:
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`.
`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.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
class <- classPCA(CuticleIR, choice = "noscale")
## Not run:
mclust3dSpectra(CuticleIR, 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

Mclust for the underlying function.

Examples

```
data(CuticleIR)
class <- classPCA(CuticleIR)
mclustSpectra(CuticleIR, class, main = "Cuticle IR", plot = "BIC")
mclustSpectra(CuticleIR, class, main = "Cuticle IR", plot = "proj")
mclustSpectra(CuticleIR, class, main = "Cuticle IR", plot = "error",
truth = CuticleIR$groups)
```

---

normSpectra *Normalize a Spectra Object*

Description

This function carries out normalization of the spectra in a Spectra object. There are currently two options, though others are readily added. "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. This each spectrum sums to 1. This method assumes that the total concentration of 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.

Usage

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

Arguments

spectra	An object of S3 class "Spectra" to be normalized.
method	One of c("TotInt", "Range") 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
res <- normSpectra(CuticleIR)
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.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
x1 <- rnorm(20, 2, 2)
range(x1)
sd(x1)/diff(range(x1))
x2 <- normVec(x1)
range(x2)
sd(x2)/diff(range(x2))
```

---

pcaBoot *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. Nothing more than a wrapper to Peter Filzmoser's pcaCV method with some small plotting changes.

Usage

```
pcaBoot(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".
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto").
pcs	As per pcaCV where it is called amax; an integer giving the number of PC scores to include.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html
K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

See Also

pcaCV for the underlying function.

Examples

```
data(SrE.IR)
results <- pcaBoot(SrE.IR, pcs = 5, choice = "noscale")
```

---

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html
K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

See Also

pcaDiagplot in package chemometrics for the underlying function.

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
temp <- pcaDiag(SrE.IR, results, pcs = 2, plot = "OD")
temp <- pcaDiag(SrE.IR, results, 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 classPCA or robPCA 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

See plotLoadings to plot one loading against the original variable (frequency) axis.

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plot2Loadings(SrE.IR, main = myt, results,
loads = c(1,2), tol = 0.05)
```

`plotHCA` *Plot Dendrogram for Spectra Object*

Description

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

Usage

```
plotHCA(spectra, distance, sub.title, method, use.sym, ...)
```

Arguments

`spectra`	An object of S3 class `"Spectra"`.
`sub.title`	A character string for the subtitle.
`distance`	A distance matrix.
`method`	A method acceptable to the `hclust` function.
`use.sym`	Logical; if true, the color scheme will be black and lower-case letters will be used to indicate group membership.
`...`	Additional parameters to be passed to the plotting routines.

Value

None; side effect is a plot.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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 classPCA or robPCA 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

See plot2Loadings to plot two loadings against each other.

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotLoadings(SrE.IR, results, 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, ...)
```

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 classPCA or robPCA 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.
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.
...	Additional parameters to be passed to the plotting functions.

Value

None. Side effect is a plot.

Calls

plotScoresDecoration, plotScoresCor, chkSpectra

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

For other ways of displaying the results, plotScree, plotLoadings, plot2Loadings. For a 3D plot of the scores, see plotScores3D.

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotScores(SrE.IR, main = myt, results,
pcs = c(1,2), ellipse = "rob", 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 = 0.80,
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 classPCA or robPCA 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

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

Examples

```
data(CuticleIR)
results <- classPCA(CuticleIR, choice = "noscale")
plotScores3D(CuticleIR, results, main = "Cuticle IR Spectra")
```

---

plotScoresCor *Compute Confidence Ellipses*

Description

A utility function which when given a x,y data set computes both classical and robust confidence ellipses. Never 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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. Never 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 classPCA or robPCA 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

---

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

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

Examples

```
data(CuticleIR)
results <- classPCA(CuticleIR, choice = "autoscale")
## Not run:
plotScoresRGL(CuticleIR, results, title = "Cuticle IR 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 `classPCA` or `robPCA` 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.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html The idea for the alternative style plot came from the NIR-Quimiometria blog by jrcuesta, at http://http://www.r-bloggers.com/pca-for-nir-spectra_part-004-projections/

Examples

```
data(SrE.IR)
results <- classPCA(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotScree(results, main = myt)
plotScree2(results, main = myt)
```

`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), ...)
```

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

Value

None. Side effect is a plot.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
myt <- expression(bolditalic(Portulaca)~bolditalic(oleracea)~bold(Cuticle~IR~Spectra))
plotSpectra(CuticleIR, main = myt,
which = c(10:11, 40:41, 100:101, 140:141, 150:151),
yrange = c(0,10),
offset = 0.8, amplify = 1.0, lab.pos = 2000)
```

---

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*. Not likely to be called by most users.

Usage

```
q2rPCA(x)
r2qPCA(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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

prcomp, princomp

---

readBrukerTxt *Read and process a Bruker NMR spectrum saved using Topspin*

Description

These functions read files written by the Bruker Topspin software. This function is not extensively tested. Not normally called by the user; used byfiles2SpectraObject.

Usage

```
readBrukerTxt(file = "", debug = FALSE)
readBrukerAscii(file = "", debug = FALSE)
```

Arguments

file	Character; the path to the file to be processed.
debug	Logical indicating if file names and progress information should be printed to the console. Useful for troubleshooting.

Value

A data frame with the following elements:

x	Extracted frequency values
y	Extracted intensities

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

---

readJDX *Read and process a JCAMP-DX file.*

Description

This function reads files with the JCAMP-DX format (and extension ".dx"). This function is not extensively tested. It does not work with NMR data. Not normally called by the user; used by files2SpectraObject.

Usage

```
readJDX(file = "", debug = FALSE)
```

Arguments

file	Character; the path to the file to be processed.
debug	Logical indicating if file names and progress information should be printed to the console. Useful for troubleshooting.

Details

The data block must be of the type XYDATA=(X++(Y..Y)) It handles AFFN format for the data block and only with '+', '-' or ' ' as the separator.

Value

A data frame with the following elements:

x	Extracted frequency values
y	Extracted intensities

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html The details of the JCAMP-DX formats can be found at at http://www.jcamp-dx.org/

`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 some 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
sumSpectra(CuticleIR) # note the frequency range & existing gap
# remove frequencies from one end:
newIR <- removeFreq(CuticleIR, rem.freq = CuticleIR$freq > 3500)
# remove frequencies from both ends at once:
newIR <- removeFreq(CuticleIR, rem.freq = CuticleIR$freq > 3500
| CuticleIR$freq < 800)
# remove frequencies from the middle:
newIR <- removeFreq(CuticleIR, rem.freq = CuticleIR$freq > 800
& CuticleIR$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)
```

---

removeSample *Remove Samples or Groups from a Spectra Object*

Description

Removes specified samples from a "Spectra" object.

Usage

```
removeSample(spectra, rem.sam)
removeGroup(spectra, rem.group)
```

Arguments

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

Details

If rem.sam or rem.group is a character vector, the sample or group names are grepped for the corresponding values. Remember that the grepping process is greedy, i.e. grepping for "XY" find not only "XY" but also "XYZ". Unused levels in $groups are dropped. removeSample removes samples (objects) based upon the sample names. removeGroup removes entire groups based upon the group name.

Value

A modified object of S3 class "Spectra".

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

removeFreq to remove selected frequencies from a "Spectra" object.

Examples

```
data(CuticleIR)
new1 <- removeSample(CuticleIR, rem.sam = 20)
# removes the 20th spectrum/sample
new2 <- removeSample(CuticleIR, rem.sam = "GE")
# removes all samples whose name contains "GE"
new3 <- removeSample(CuticleIR, rem.sam = "GE10")
# removes one spectrum/sample with this exact name.
```

---

robPCA *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

```
robPCA(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).

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html
K. Varmuza and P. Filzmoser *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, 2009.

See Also

See PCAgrid on which this function is based. For the classical version, see classPCA.
For displaying the results, plotScree, plotScores, plotLoadings, plotScores3D

Examples

```
data(SrE.IR)
results <- robPCA(SrE.IR, choice = "mad")
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
plotScores(SrE.IR, main = myt,
results, pcs = c(1,2), ellipse = "rob", tol = 0.05)
```

---

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 amap. All this function does is determine which method is requested and then the distance calculation is done by the appropriate method.

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", "maxi

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.

Value

An object of class dist.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu> Suggested and original written by Roberto Canteri.

`seXy` *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

```
seXy(x)
seXy95(x)
seXyMad(x)
seXyIqr(x)
seX(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`. `seXy` returns the mean =\- the standard error. `seXy95` returns the mean =\- the 95

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. So for example, `sd` gives the mean +/- the standard deviation. For `seX`, a single value giving the standard error of x.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

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

`shrinkLeaf` *Shrink the Leaves of a Dendrogram*

Description

This function shrinks the size of leaves of a dendrogram object. The code was taken from the help files. Not for end-user use.

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.

Called by

`hcaSpectra`, `hcaScores`.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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

Description

This function computes and plots various measures of central tendency and spread for a "Spectra" object. Several different measures/spreads are available. The computation can be done by group or using the entire data set.

Usage

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

Arguments

spectra	An object of S3 class "Spectra" to be analyzed.
method	One of c("sd", "sem", "sem95", "mad", "iqr"). sd plots the mean +/- the standard deviation, sem computes the mean +/- the standard error of the mean, sem95 computes the mean +/- the standard error at the 95 percent confidence interval, mad computes the median +/- the median absolute deviation, and finally, iqr plots the median + the upper hinge and - the lower hinge.
by.gr	Logical, indicating if the analysis is to be done by group or not.
...	Additional parameters to be passed to the plotting routines.

Value

None; side effect is a plot

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
myt <- expression(bolditalic(Portulaca)~bolditalic(oleracea)~bold(Cuticle~IR~Spectra))
specSurvey(CuticleIR, method = "iqr", main = myt)
```

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 similar functions (no others currently exist).

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

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. See the example for usage. 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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

conColScheme.

Examples

```
data(CuticleIR)
# original factor encoding from when CuticleIR was created:
levels(CuticleIR$groups)
# split those original levels into 2 new ones (re-code them):
new.groups <- list(gen = c("G", "T"), trt = c("C", "E"))
new.CuticleIR <- splitSpectraGroups(CuticleIR, new.groups)
str(new.CuticleIR) # note two new elements, "gen" and "trt"
# split into one new factor and re-color:
new.groups <- list(gen = c("G", "T")) # only one element
new.CuticleIR <- splitSpectraGroups(CuticleIR, new.groups, rep.cols = c("pink", "orange"))
str(new.CuticleIR) # note one new element, "gen" and the colors have changed.

# note that if you want to use a newly created group in
# plotScores and other functions to drive the color scheme, you'll
# have to copy a new group to the groups element:
new.CuticleIR$groups <- new.CuticleIR$gen
```

---

sPlotSpectra *s-Plot of Spectra Data Post PCA*

Description

sPlotSpectra 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 classPCA or robPCA).
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 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. <hanson@depauw.edu>

References

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

Examples

```
data(SrE.IR)
IR.pca <- classPCA(SrE.IR)
myt <- expression(bolditalic(Serenoa)~bolditalic(repens)~bold(IR~Spectra))
splot <- sPlotSpectra(spectra = SrE.IR, pca = IR.pca, pc = 1, tol = 0.005,
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.

Usage

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

Format

The format is: List of 9
$ freq : num [1:1869] 399 401 403 405 407 ...
$ data : num [1:16, 1:1869] 0.0914 0.0921 0.0848 0.0858 0.0929 ...
$ names : chr [1:16] "CVS_adSrE" "ET_pSrE" "GNC_adSrE" "LF_adSrE" ...
$ groups : Factor w/ 4 levels "adSrE","EPO",..: 1 4 1 1 4 4 1 1 4 1 ...
$ colors : chr [1:16] "#984EA3" "#E41A1C" "#984EA3" "#984EA3" ...
$ sym : num [1:16] 15 1 15 15 1 1 15 15 1 15 ...
$ alt.sym: chr [1:16] "d" "a" "d" "d" ...
$ unit : chr [1:2] "wavenumber" "absorbance"
$ desc : chr "Serenoa repens IR quality study"
- attr(*, "class")= chr "Spectra"

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.

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. <hanson@depauw.edu>

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

See Also

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

Examples

```
data(CuticleIR)
sumGroups(CuticleIR)
```

---

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).

Value

None. Results printed at console, perhaps a plot as well.

Author(s)

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

References

http://academic.depauw.edu/~hanson/ChemoSpec/ChemoSpec.html

Examples

```
data(CuticleIR)
sumSpectra(CuticleIR)
```

Index