Package ‘ChemoSpec’

March 16, 2015

Type Package

Title Exploratory Chemometrics for Spectroscopy

Version 4.0.1

Date 2015-03-16

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, Raman) and carrying out various forms of top-down exploratory data analysis, such as HCA, PCA, model-based clustering and STOCSY analysis. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed with metabolomics data sets in mind, where the samples fall into groups such as treatment and control. Graphical output is formatted consistently for publication quality plots. ChemoSpec is intended to be very user friendly and help you get usable results quickly. A vignette illustrating typical operations is available.

License GPL-3

Depends R (>= 3.1)

Imports plyr, rgl

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

ByteCompile TRUE

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

VignetteBuilder knitr

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

NeedsCompilation no

Repository CRAN

Date/Publication 2015-03-16 15:25:08
**R topics documented:**

- ChemoSpec-package ........................................... 3
- aovPCAloadings .............................................. 4
- aovPCAscores ................................................ 5
- aov.pcaSpectra ............................................... 6
- avgFacLvs .................................................. 7
- baselineSpectra ............................................. 8
- binData ..................................................... 9
- binSpectra .................................................. 10
- check4Gaps .................................................. 11
- chkSpectra .................................................. 12
- chooseLvs ................................................... 13
- clupaSpectra ............................................... 15
- colLeaf ..................................................... 16
- colorSymbol ............................................... 16
- conColScheme ............................................... 17
- coordProjCS ............................................... 18
- corSpectra .................................................. 19
- cv_pcaSpectra .............................................. 21
- c_pcaSpectra .............................................. 22
- evalClusters ............................................... 23
- files2SpectraObject ...................................... 25
- getMaxCovByFreq ......................................... 27
- groupNcolor ............................................... 28
- hcaScores .................................................. 29
- hcaSpectra .................................................. 30
- hmapSpectra ............................................... 31
- hypTestScores ............................................. 32
- isWholeNo .................................................. 33
- labelExtremes .............................................. 34
- labelExtremes3d .......................................... 35
- loopThruSpectra ........................................... 36
- makeEllipsoid ............................................. 37
- mclust3D ................................................... 38
- mclust3dSpectra ........................................... 39
- mclustSpectra ............................................. 41
- metMUD1 .................................................... 42
- normSpectra ............................................... 43
- normVec ................................................... 44
- pcaDiag .................................................... 45
- plot2Loadings .............................................. 46
- plotHCA .................................................... 47
- plotLoadings .............................................. 48
- plotScores ................................................ 49
- plotScores3D ............................................. 50
- plotScoresCor ............................................ 52
- plotScoresDecoration .................................... 53
Description

A collection of functions for plotting spectra (NMR, IR, Raman) and carrying out various forms of top-down exploratory data analysis, such as HCA, PCA, model-based clustering and STOCSY analysis. Robust methods appropriate for this type of high-dimensional data are available. ChemoSpec is designed with metabolomics data sets in mind, where the samples fall into groups such as treatment and control. Graphical output is formatted consistently for publication quality plots. ChemoSpec is intended to be very user friendly and help you get usable results quickly. A vignette illustrating typical operations is available.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>ChemoSpec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>4.0.1</td>
</tr>
<tr>
<td>Date</td>
<td>2015-03-16</td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
</tbody>
</table>
Author(s)

Bryan A. Hanson, DePauw University, Greencastle Indiana USA
Maintainer: Bryan A. Hanson <hanson@depauw.edu>

References

https://github.com/bryanhanson/ChemoSpec

aovPCAladings

Plot aovPCAscores Loadings of a Spectra Object

Description

Uses the results from aovPCAscores to plot the corresponding loadings.

Usage

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

Arguments

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

Value

None. Side effect is a plot.

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec
**aovPCAscores**  
*Plot ANOVA-PCA Scores from a Spectra Object*

**Description**  
Uses the results from `aov_pcaSpectra` 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 `aov_pcaSpectra`.
- `plot`  
  An integer specifying which scores to plot.
- `type`  
  Either classical ("cls") or robust ("rob"); Results in either `c_pcaSpectra` or `r_pcaSpectra` being called on the `Spectra` object.
- `choice`  
  The type of scaling to be performed. See `c_pcaSpectra` and `r_pcaSpectra` for details.
- `...`  
  Additional parameters to be passed to `plotScores`.

**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**
- [https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

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

---

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

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

---

### 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

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

### Arguments

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

### 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

- [https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)
avgFacLvls

See Also

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

Examples

data(metMUD2)
# Original factor encoding:
levels(metMUD2$groups)
# Split those original levels into 2 new ones (re-code them)
new.grps <- list(geneBb = c("B", "b"), geneCc = c("C", "c"))
mM3 <- splitSpectraGroups(metMUD2, new.grps)
# run aov_pcaSpectra
mats <- aov_pcaSpectra(mM3, fac = c("geneBb", "geneCc"))
apca1 <- aovPCAscores(mM3, mats, plot = 1, main = "aovPCA: B vs b")
apca2 <- aovPCAscores(mM3, mats, plot = 2, main = "aovPCA: C vs c")
apca3 <- aovPCAscores(mM3, mats, plot = 3, main = "aovPCA: Interaction Term")
apca4 <- aovPCAloadings(spectra = mM3, LM = mats, pca = apca1, main = "aov_pcaSpectra: Bb Loadings")

avgFacLvls

Average Levels of a Factor in a Data Matrix

Description

avgFacLvls takes a matrix and calculates the column means for each level of each factor given. It then replaces the original matrix rows with the means corresponding to the factor/level membership 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>

References

https://github.com/bryanhanson/ChemoSpec
See Also

`aov_pcaSpectra` for full details.

Examples

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

```
baselineSpectra  Baseline Correction of a Spectra Object

Description

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

Usage

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

Arguments

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

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`), or use `plotSpectra` on the corrected data.

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>
**binData**

**References**

https://github.com/bryanhanson/ChemoSpec

**Examples**

```r
data(SrE.IR)
require("IDPmisc") # needed specifically for rfbaseline
temp <- baselineSpectra(SrE.IR, int = FALSE, method = "rfbaseline")
par(mfrow = c(1,1)) # needed to cancel 2 panel plot
```

---

**Bin or Bucket Data**

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

The x and y values must be contiguous in the sense that there are no gaps in the values (i.e., \(x[n] - x[n+1]\) must be the same for the entire data set; this can be checked with `diff` and is checked internally. Note that this function is normally called by `binSpectra` and that function can handle gaps, sending each continuous piece of data here to be binned. If `length(x)` or `length(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

https://github.com/bryanhanson/ChemoSpec

Examples

```r
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)
```

---

**binSpectra**

*Bin or Bucket a Spectra Object*

Description

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

Usage

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

Arguments

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

Details

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

Value

An object of S3 class `Spectra`. 
Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

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

check4Gaps

Check for Missing Values (Gaps)

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

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

Arguments
x A numeric vector to be checked for gaps.
y An optional vector of y-values which correspond to the x values. Only needed
  if plot = TRUE.
tol A number indicating the tolerance for checking if the step between successive x
  values are the same. Depending upon how the x values are stored and rounded,
you may need to change the value of tol to avoid flagging trivial "gaps".
plot Logical indicating if a plot of the gaps should be made. If TRUE, y must be
  provided. The plot is labeled consistent with calling this function on a Spectra
  object.
silent Logical indicating a "no gap" condition (return value is FALSE) should not be
  reported to the console. Important because check4Gaps is called iteratively by
  other functions.
... Other parameters to be passed to the plot routines if plot = TRUE, e.g. xlim.
Details

The basic procedure is to compare \( x[n+1] - x[n] \) for successive values of \( n \). When this value jumps, there is a gap which is flagged. \( \text{begindx} \) and \( \text{endindx} \) 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, \( \text{FALSE} \) is returned.

\[
\begin{align*}
\text{beg.freq} & \quad \text{The first frequency value in a given data chunk.} \\
\text{end.freq} & \quad \text{The last frequency value in a given data chunk.} \\
\text{size} & \quad \text{The length (in frequency units) of the data chunk.} \\
\text{begindx} & \quad \text{The index of the first frequency value in the data chunk.} \\
\text{endindx} & \quad \text{The index of the last frequency value in the data chunk.}
\end{align*}
\]

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

Examples

\begin{verbatim}
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(SrE.IR)
newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 1800 & SrE.IR$freq < 2500)
check4Gaps(newIR$freq, newIR$data[1,], plot = TRUE)
\end{verbatim}

chkSpectra

Verify the Integrity of a Spectra Object

Description

Utility function to verify that the structure of a \textit{Spectra} object (an instance of an S3 object) is internally consistent. Rather than directly manipulating a \textit{Spectra} object, one should manipulate it via \texttt{removeFreq} or \texttt{removeSample}.

Usage

\begin{verbatim}
chkSpectra(spectra, confirm = FALSE)
\end{verbatim}
chooseLvls

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

https://github.com/bryanhanson/ChemoSpec

Examples

data(metMUD1)
chkSpectra(metMUD1, confirm = TRUE) # OK
# What's next is the wrong way to manipulate a Spectra object.
# Use removeSample instead.
# We won't run during checking as an error is raised
## Not run:
remove <- c(20:40)
metMUD1$freq <- metMUD1$freq[-remove]
chkSpectra(metMUD1, confirm = TRUE) # not OK, you didn't listen to me!

## End(Not run)

chooseLvls

Select Levels for Contour and Image Type Plots

Description

Given a matrix, this function will assist in selecting levels for preparing contour and image type plots. For instance, levels can be spaced evenly, logarithmically, or exponentially.

Usage

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

Arguments

M
A numeric matrix.

n
For all methods except ecdf, an integer giving the number of levels desired. For
ecdf, n should be one or more values in the interval (0...1). For instance, a value
of 0.6 corresponds to a single level in which 60 percent of the matrix values are
below, and 40 percent above.

mode
Character. One of c("even", "log", "exp", "ecdf", "posexp", "negexp", "poslog", "neglog")
"even" will create evenly spaced levels. "log" will create levels which are more
closely spaced at the extremes, while "exp" does the opposite. The pos- or neg-
versions select just the positive or negative values. "ecdf" computes levels at
the requested quantiles of the matrix.

lambda
Numeric. A non-zero exponent used with method = "exp" and relatives.

base
Integer. The base used with method = "log" and relatives.

showHist
Logical. Shall a histogram be drawn showing the location of the chosen levels?

... Arguments to be passed downstream.

Value

A numeric vector giving the levels.

Note

This function is called by corSpectra and covSpectra if levels are not provided to those functions.

Author(s)

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

References

https://github.com/bryanhanson/ChemOSpec

Examples

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

| clupaSpectra | Conduct Hierarchical Cluster-Based Peak Alignment on a Spectra Object |

#### Description

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

#### Usage

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

#### Arguments

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

#### Value

A modified `Spectra` object.

#### Author(s)

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

#### References


[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

#### Examples

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

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

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

### Usage
```
colLeaf(n, spectra)
```

### Arguments

- **n**  
  A node in a dendrogram object.

- **spectra**  
  An object of S3 class `Spectra`.

### Value
Returns a node with the label color properties set.

### Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

### References
[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

---

**colorSymbol**

*Colors and Symbols in ChemoSpec and Spectra Objects*

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

### Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

### References
[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)
conColScheme

Change the Color Scheme of a Spectra Object

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

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

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

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

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

Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

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

Examples
data(metMUD1)
sumSpectra(metMUD1)
newSpec <- conColScheme(metMUD1,
new = c("pink", "violet"))
sumSpectra(newSpec)
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. An internal function, not generally called by the user.

Usage

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

Arguments

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

Value

See coordProj.
### Description

These functions provide tools for STOSCY analysis of a `Spectra` object, using the method developed by Nicholson. STOSCY is Statistical Total Correlation Spectroscopy. Briefly, the correlation matrix of an NMR data set of \( n \) samples and \( p \) frequencies is computed (the matrix dimensions are \( p \times p \)). Peaks arising from the same compound are intrinsically positively correlated. This is much like a 1D or 2D TOCSY NMR spectrum. However, peaks that are correlated, positively or negatively, due to metabolic processes, will also appear in the STOSCY plot. `corspectra` computes the correlation and covariance matrices, and can display the correlation matrix in several formats. Detailed inspection/interpretation of this plot is tedious, and producing it can be slow for large data sets, so it’s most useful as an overview. `covspectra` will display a single frequency (i.e. chemical shift) from the covariance matrix, but color it according to the correlation matrix values. This is the point where detailed interpretation is done. See the example.

### Usage

```r
corspectra(spectra, plot = TRUE, limX = NULL, limY = NULL, nticks = 10,
            levels = NULL, pmode = "contour", C = NULL, V = NULL, ...)

covspectra(spectra, freq = spectra$freq[1], C = NULL, V = NULL, ...)
```

### Arguments

- `spectra`: An object of S3 class `Spectra`.
- `plot`: Logical. Should a plot be made? Applies to `corspectra` only.
- `limX`: Numeric vector of length 2. The x limits. Applies to `corspectra` only.
- `limY`: Numeric vector of length 2. The y limits. Applies to `corspectra` only.
- `nticks`: Integer. The number of ticks to be drawn. Applies to `corspectra` only.
- `levels`: Numeric. A vector of values at which to draw the contours or levels. Applies to `corspectra` only. The default is to use `chooseLvl` to compute 5 evenly spaced levels. For most data sets this should only be considered a starting point. A histogram of the correlation matrix can be very helpful in choosing levels.
- `pmode`: Character. The plot mode. Applies to `corspectra` only. One of c("contour", "image", "contourplot", "levelplot", "rgl", "excon"). The last two are interactive. See Details.
corSpectra

V
freq
...

Numeric. The desired frequency to be plotted. Applies to covSpectra only.
Other parameters to be passed to the plotting functions.

Details

If spectra$freq is in decreasing order, it and spectra$data are silently re-ordered to be increasing before plotting.

The calculation of the correlation and covariance matrices may take quite some time for large data sets. It is possible to pre-compute these and pass them into the functions to save time and avoid repetition.

Plotting in corSpectra can be extremely slow for large data sets. The base graphics options (pmode = "contour" or "image") are much faster than the lattice options. These plots are probably best for an overall sense of the data and for publication rather than detailed interpretation. If using pmode = "contour" drawing fewer contours is of course faster for both drawing and computation of the contours. Note too that contour style plots have \( n \) colors for \( n \) contour levels but image style plots have \( n-1 \) colors for \( n \) levels.

The color scale for the plots is blue/low correlation to red/high correlation, anchored at a shade of green for zero correlation. The example shows how to see the color scale.

For covSpectra, the x and y limits can be set simply by passing xlim and ylim via the . . . .

Value

A list giving the covariance and correlation matrices. A plot will also be made if plot = TRUE for corSpectra.

Warning

For corSpectra, the labeling of the axis will be wrong if there is a gap in the data.

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

See Also

chooseLvl1s to select levels automatically.
getMaxCovByFreq is a utility to sort the covariance matrix by the maximum covariance at a given frequency. This can be used to identify the strongest interactions.
Examples

# The color scale used:
# cscale <- c(rev(rainbow(4, start = 0.45, end = 0.66)), rev(rainbow(5, start = 0.0, end = 0.25)))
# pie(rep(1, 9), col = cscale)
#
# This data set is a mixture of compounds, some of
# which have correlated concentrations, both
# positively and negatively. The contour plot shows
# correlations w/i a spin system and among compounds.
# The 2nd plot shows one particular frequency.
# For more info about the data set see ?MUD
#
# data(metMUD2)
# lvs <- c(-0.99, -0.95, -0.9, 0.9, 0.95, 0.99)
# lim <- c(0.6, 4.3)
# res <- corSpectra(metMUD2, levels = lvs, limX = lim, limY = lim, main = "metMUD2 NMR Data")
# jnk <- covSpectra(metMUD2, freq = 1.0, C = res[[2]], V = res[[1]])

cv_pcaSpectra

Cross-Validation of Classical PCA Results for a Spectra Object

Description

This function carries out classical PCA on the data in a Spectra object using a cross-validation method. Nothing more than a wrapper to Peter Filzmoser's pcaCV method with some small plotting changes.

Usage

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

Arguments

spectra [An object of S3 class Spectra.]
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.
c_pcaSpectra

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
https://github.com/bryanhanson/ChemoSpec

See Also
pcaCV for the underlying function.

Examples

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

c_pcaSpectra

Classical PCA of Spectra Objects

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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spectra</td>
<td>An object of S3 class Spectra.</td>
</tr>
<tr>
<td>choice</td>
<td>A character string indicating the choice of scaling. One of c(&quot;noscale&quot;, &quot;autoscale&quot;, &quot;Pareto&quot;).</td>
</tr>
<tr>
<td>cent</td>
<td>Logical: whether or not to center the data. Always center the data unless you know it to be already centered.</td>
</tr>
</tbody>
</table>
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 \texttt{prcomp}, modified to include a list element called $\texttt{method}$, 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


https://github.com/bryanhanson/ChemoSpec

See Also

\texttt{prcomp} for the underlying function, \texttt{r_pcaSpectra} for analogous robust PCA calculations.

For displaying the results, \texttt{plotScree}, \texttt{plotScores}, \texttt{plotLoadings}, \texttt{plot2Loadings}, \texttt{sPlotSpectra}, \texttt{plotScores3D}, \texttt{plotScoresRGL}.

Examples

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

---

**evalClusters**

\textit{Evaluate or Compare the Quality of Clusters Quantitatively}

Description

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

Usage

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

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

**Details**

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

**Value**

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

**Author(s)**

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

**References**


[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

**See Also**

- `hclust` for the underlying base function. `hcaSpectra` for HCA analysis of a `Spectra` object. `hcaScores` for HCA analysis of PCA scores from a `Spectra` object. `plotHCA` for the plotting function in ChemoSpec.
files2SpectraObject

Examples

# These are a little slow for CRAN checking
## Not run:
data(metMUD2)
# Using clusterCrit
res1 <- hcaSpectra(metMUD2) # default clustering and distance methods
res2 <- hcaSpectra(metMUD2, d.method = "cosine")
# The return value from hcaSpectra is a list with hclust as the first element.
crit1 <- evalClusters(metMUD2, pkg = "clusterCrit", hclst = res1[[1]], k = 2)
crit2 <- evalClusters(metMUD2, pkg = "clusterCrit", hclst = res2[[1]], k = 2)
# crit1 and crit2 can now be compared.

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

## End(Not run)

files2SpectraObject  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",
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.

gc.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, or dx for JCAMP-DX files.
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 = "dx".
...  Other arguments to be passed downstream.

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_Sun" and "Sample_1_Shade" you can't use gr.crit = c("Control", "Sample", "Sun", "Shade") because each criteria is greppe 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 (i.e. no header row). The frequency column is assumed to be the same in all .csv files. If format = "dx", then the corresponding file type will be processed (consider setting debug = TRUE for this format). See readJDX for limitations (there are many options for JCAMP, and most are untested).

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!
getMaxCovByFreq

Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

getMaxCovByFreq  Order Rows of a Covariance Matrix by Absolute Value

Description
Given a covariance matrix, the maximum absolute value of each row is computed. The results are ordered, and optionally, limited to values greater than a particular value.

Usage
getMaxCovByFreq(spectra, V = NULL, Quan = NULL)

Arguments
- spectra: An object of S3 class Spectra.
- V: A numeric covariance matrix.
- Quan: Numeric. A value in the interval (0...1) giving the quantile to be selected. For instance, Quan = 0.1 selects the top 10 percent of values.

Value
A data frame containing the frequencies from the Spectra object, the absolute value of the covariance at that frequency, and the relative covariance. Sorted by absolute covariance in descending order.

Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

Examples
```r
data(metMUD1)
V <- cov(metMUD1$data)
# Look at the top 1%
res <- getMaxCovByFreq(metMUD1, V, Quan = 0.01)
res
```
groupNcolor

Assign Group Membership and Colors for a Spectra Object

Description

A utility function which looks for \texttt{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 \texttt{gr.crit} or there are entries in \texttt{gr.crit} that don't match any file names. An internal function, not generally called by the user.

Usage

\begin{verbatim}
groupNcolor(spectra, gr.crit = NULL, gr.cols = c("auto"))
\end{verbatim}

Arguments

\begin{itemize}
\item \textbf{spectra} \hspace{1cm} An object of S3 class \texttt{Spectra}. Until this function acts on \texttt{spectra} it is not quite complete.
\item \textbf{gr.crit} \hspace{1cm} As per \texttt{files2SpectraObject}
\item \textbf{gr.cols} \hspace{1cm} As per \texttt{files2SpectraObject}
\end{itemize}

Value

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

Author(s)

Bryan A. Hanson, DePauw University. \texttt{<hanson@depauw.edu>}

References

\begin{verbatim}
https://github.com/bryanhanson/ChemoSpec
\end{verbatim}

See Also

\begin{verbatim}
files2SpectraObject for details; sumGroups to see the outcome.
\end{verbatim}
hcaScores

**HCA on PCA scores from a Spectra Object**

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

**Usage**

```r
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 `c_pcaSpectra` or `r_pcaSpectra` were used to create `pca`.
- `scores`: A vector of integers specifying which scores to use for the HCA.
- `c.method`: A character string describing the clustering method; must be acceptable to `hclust`.
- `d.method`: A character string describing the distance calculation method; must be acceptable as a method in `rowDist`.
- `use.sym`: A logical; if true, use no color and use lower-case letters to indicate group membership.
- `...`: Additional parameters to be passed to the plotting functions.

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

**Author(s)**
Bryan A. Hanson, DePauw University. `<hanson@depauw.edu>`

**References**

- [https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

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

data(SrE.IR)
results <- c_pcaSpectra(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serenaa)-bolditalic(repens)-bold(IR-Spectra))
res <- 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. Many methods for computing the clusters and 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

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

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

See Also

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

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

hmapSpectra

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 `RCColorBrewer`.
- **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

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

See Also

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

data(SrE.IR)
# remove noisy, uninteresting region:
newIR <- removeFreq(SrE.IR, rem.freq =
SrE.IR$freq > 1000 & SrE.IR$freq < 2500)
# now make the heat map:
hmapSpectra(newIR, cexRow = 0.5, cexCol = 0.1, no.col = 9)

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>
isWholeNo

Determine if a Number is a Whole Number

Description

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

Usage

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

Arguments

  x  A number to be tested.

  tol  Tolerance for the test.

Value

A logical, indicating the outcome of the test.

Author(s)

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

Label Extreme Values in a 2D Data Set

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

Usage
labelExtremes(data, names, tol)

Arguments
- **data**: A matrix containing the x values of the points/samples in the first column, and the y values in the second.
- **names**: A character vector of sample names. Length must match the number of rows in x.
- **tol**: A number describing the fraction of points to be labeled. \( \text{tol} = 1.0 \) labels all the points; \( \text{tol} = 0.05 \) 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
- https://github.com/bryanhanson/ChemoSpec
labelExtremes3d  

Identify Extreme Values in 3D

Description

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

Usage

labelExtremes3d(data, names, tol)

Arguments

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

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

tol  A number describing the fraction of points to be labeled. tol = 1.0 labels all the points; tol = 0.05 labels approximately the most extreme 5 percent. Note that this is simply based upon quantiles, assumes that x, y and z are each normally distributed, and treats x, y and yz separately. Thus, this is not a formal treatment of outliers, just a means of labeling points. Note too that while this function could deal with groups separately, the way it is called by plotScoresRGL lumps all groups together.

Value

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

Author(s)

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

References

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

Display the Spectra in a Spectra Object One at a Time

Description

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

Usage

loopThruSpectra(Spectra, ...)

Arguments

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

Value

None. Side effect is a plot.

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

Examples

## Not run:
data(metMUD1)
loopThruSpectra(metMUD1)

## End(Not run)
**makeEllipsoid**  
*Create Ellipsoid*

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Note**
The idea was taken from "An Introduction to rggobi" found at the ggobi web site: [http://www.ggobi.org](http://www.ggobi.org). I added the robust option.
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.
c1 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 each row of the PC scores. Generally this would be spectra$groups.
mclust3dSpectra

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.
c1 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 each 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 de-
sired 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

https://github.com/bryanhanson/ChemoSpec

Examples

```r
## Not run:
data(metMUD1)
class <- c_pcaSpectra(metMUD1)
mclust3dSpectra(metMUD1, class, title = "mclust3dSpectra demo",
lab.opts = FALSE, t.pos = "A")
## End(Not run)
```

Description

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

Usage

```r
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

https://github.com/bryanhanson/ChemoSpec

See Also

Mclust for the underlying function.

Examples

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

Description

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

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

metMUD1 is composed of 20 samples, each a mixture of four typical small organic compounds (we’ll leave it to the reader as an exercise to deduce the spin systems!). These compounds are present in varying random amounts. Ten of the samples are control samples, and ten are treatment samples. Thus you can run PCA and other methods on this data set, and expect to see a separation. corSpectra analysis of this data set is interpreted as a 2D TOCSY NMR: any resonances on a particular horizontal or vertical line (frequency) is part of a single spin system. Another way to think about this is that all the peaks for a single compound are positively correlated with each other but not with any other compound. It’s useful to keep in mind that the resonances of the four compounds overlap considerably in some cases. This data set is normalized.

metMUD2 also consists of 20 samples of mixtures of the same four compounds. However, the concentrations of some of the compounds are correlated with other compounds, both positively and negatively, and some concentrations are random. metMUD2 is divided into different sample groups which correspond conceptually to two genes, each active or knocked out. This data set is designed to be similar to a metabolomics data set in which the concentrations of some compounds co-vary, and others are independent. Compare the corSpectra plot of this data set to the metMUD1 where there is no co-variation. This data set is normalized.
**normSpectra**

**Usage**

```r
data(metMUD1)
```

**Format**

The data is stored as a `Spectra` object.

**Source**

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

**References**

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

---

**normSpectra**

*Normalize a Spectra Object*

**Description**

This function carries out normalization of the spectra in a `Spectra` object. There are currently three options, though others are readily added. "PQN" carries out "Probabalistic Quotient Normalization" as described in the reference. "TotInt" normalizes by total intensity. In this case, the y-data of a `Spectra` object is normalized by dividing each y-value by the sum of the y-values in a given spectrum. Thus each spectrum sums to 1. This method assumes that the total concentration of substances giving peaks does not vary across samples which may not be true. "Range" allows one to do something similar but rather than using the sum of the entire spectrum as the denominator, only the sum of the given range is used. This would be appropriate if there was an internal standard in the spectrum which was free of interference.

**Usage**

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

**Arguments**

- `spectra` An object of S3 class `Spectra` to be normalized.
- `method` One of c("PQN", "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**


https://github.com/bryanhanson/ChemoSpec

**Examples**

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

**Description**

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

**Usage**

```r
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 gobi web site: http://www.gobi.org.

**Author(s)**

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


https://github.com/bryanhanson/ChemoSpec

See Also

`pcaDiagplot` in package `chemometrics` for the underlying function.

Examples

data(SrE.IR)
results <- c_pcaSpectra(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 ChemSpec functions `c_pcaSpectra` or `r_pcaSpectra` were used to create `pca`. 
loads A vector of two integers specifying which loading vectors to plot.
tol A number describing the fraction of points to be labeled. tol = 1.0 labels all
the points; tol = 0.05 labels the most extreme 5 percent.
... Other parameters to be passed to the plotting routines.

Value
None. Side effect is a plot.

Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

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

Examples
```r
data(SrE.IR)
results <- c_pcaSpectra(SrE.IR)
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
```r
plotHCA(spectra, hclst, sub.title, use.sym, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spectra</td>
<td>An object of S3 class Spectra.</td>
</tr>
<tr>
<td>hclst</td>
<td>A hclust object.</td>
</tr>
<tr>
<td>sub.title</td>
<td>A character string for the subtitle.</td>
</tr>
<tr>
<td>use.sym</td>
<td>Logical; if true, the color scheme will be black and lower-case letters will be used to indicate group membership.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters to be passed to the plotting routines.</td>
</tr>
</tbody>
</table>
plotLoadings

Value

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

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

---

**plotLoadings**  
*Plot PCA Loadings for a Spectra Object*

Description

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

Usage

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

Arguments

- `spectra`: An object of S3 class `Spectra`.
- `pca`: An object of class `prcomp`, modified to include a list element called `$method`, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions `c_pcaspectra` or `r_pcaspectra` were used to create `pca`.
- `loads`: An integer vector giving the loadings to plot. More than 3 loadings creates a useless plot using the default graphics window.
- `ref`: An integer specifying the reference spectrum to plot, which appears at the bottom of the plot.
- `...`: Additional parameters to be passed to plotting functions.

Value

None. Side effect is a plot.

Author(s)

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

References

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

See Also

See plot2Loadings to plot two loadings against each other.

Examples

```r
data(SrE.IR)
results <- c_pcaSpectra(SrE.IR, choice = "noscale")
myt <- expression(bolditalic(Serena)\bolditalic(repens)\bold(IR-Spectra))
plotLoadings(SrE.IR, results, main = myt,
loads = 1:2, ref = TRUE)
```

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

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

Arguments

- `spectra`: An object of S3 class Spectra.
- `pca`: An object of class prcomp, modified to include a list element called $method$, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions c_pcaSpectra or r_pcaSpectra were used to create pca.
- `pcs`: A vector of two integers specifying the PCA scores to plot.
- `ellipse`: A character vector specifying the type of ellipses to be plotted. One of c("both", "none", "cls", "rob", c1s 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.
- `leg.loc`: Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
- `...`: Additional parameters to be passed to the plotting functions.
plotScores3D

Value
None. Side effect is a plot.

Author(s)
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

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

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

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

---

**plotScores3D**

3D PCA Score Plot for a Spectra Object

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

Usage
```r
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 c_pcaSpectra or r_pcaSpectra were used to create pca.
- **pcs**: A vector of three integers specifying the PCA scores to plot.
plotScores3D

ellipsis 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.
c1 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 im-
portant. 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

https://github.com/bryanhanson/ChemoSpec

See Also

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

Examples

data(metMUD1)
results <- c_pcaSpectra(metMUD1, choice = "noscale")
plotScores3D(metMUD1, results, main = "metMUD1 NMR Spectra")
plotScoresCor  Compute Confidence Ellipses

Description

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

Usage

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

Arguments

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

Value

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

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

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

See Also

See function cor.plot in package mvoutlier on which this function is based.
**plotScoresDecoration**  
*Decorate PCA Score Plot of a Spectra Object*

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

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

**Arguments**
- `spectra`: An object of S3 class *Spectra*.
- `pca`: An object of class *prcomp*, modified to include a list element called `$method`, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions `c_pcaSpectra` or `r_pcaSpectra` were used to create `pca`.
- `pcs`: A vector of two integers specifying the PCA scores to plot.
- `tol`: A number describing the fraction of points to be labeled. `tol = 1.0` labels all the points; `tol = 0.05` labels the most extreme 5 percent.

**Value**
None. The score plot is decorated.

**Author(s)**
Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

**References**
https://github.com/bryanhanson/ChemoSpec

---

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

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

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

Arguments

- **spectra**: An object of S3 class `Spectra`.
- **pca**: An object of class `prcomp`.
- **pcs**: A vector of three integers specifying the PCA scores to plot.
- **ellipse**: Logical indicating if confidence ellipses should be drawn.
- **rob**: Logical; if `ellipse = TRUE`, indicates that robust confidence ellipses should be drawn. If `FALSE`, classical confidence ellipses are drawn.
- **cl**: A number indicating the confidence interval for the ellipse.
- **frac(pts).used**: If `ellipse = TRUE` and `rob = TRUE`, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.
- **title**: A character string for the plot title.
- **t.pos**: A character selection from `letters{QZX}` (= A through H) indicating the desired location for the title.
- **leg(pos).**: A character selection from `letters{QZX}` (= 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>
**plotScree**

References

https://github.com/bryanhanson/ChemoSpec

See Also

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

Examples

```r
data(metMUD1)
results <- c_pcaspectra(metMUD1, choice = "autoscale")
## Not run:
plotScoresRGL(metMUD1, results, title = "metMUD1 NMR Spectra",
leg.pos = "A", t.pos = "B")
## End(Not run)
```

---

**plotScree**

*Scree Plots of PCA Results for a Spectra Object*

Description

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

Usage

```r
plotScree(pca, ...)
plotScree2(pca, ...)
```

Arguments

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

- **...**
  
  Additional parameters to be passed to plotting functions.

Details

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

Value

None. Side effect is a plot.
Author(s)

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

References

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

Examples

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

plotSpectra

Plot Spectra Object

Description

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

Usage

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

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).


plotSpectraJS

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

https://github.com/bryanhanson/ChemoSpec

See Also

plotSpectraJS for the interactive version.

Examples

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

plotSpectraJS  Plot a Spectra Object Interactively

Description

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

Usage

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

Arguments

spectra An object of S3 class Spectra to be checked.
browser Character. Something that will make sense to your OS. Only necessary if you want to override your system specified browser as understood by R. See below for further details.
minify Logical. Should the JavaScript be minified (compressed) using package js? This gives code that executes faster, but is pretty-much impossible to read.
Value

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

Browser Choice

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

RStudio Viewer

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

Browser Choice/Mac

On a Mac, the default browser is called by `/bin/sh/open` which in turn looks at which browser you have set in the system settings. You can override your default with `browser = "/usr/bin/open -a "Google Chrome""` for example. Testing shows that on a Mac, Safari and Chrome perform correctly, but in Firefox the mouse cursor is slightly offset from the guides. While it doesn’t look quite right, the value of the cursor displayed is correct.

Browser Choice/Other Systems

`plotSpectraJS` has been tested on a Windows 7 professional instance running in VirtualBox using Firefox and Chrome, and runs correctly (Firefox has the same mouse position issue as mentioned above).

Author(s)

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

References

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

See Also

`plotSpectra` for non-interactive plotting.

Examples

```r
## Not run:
require("jsonlite")
require("js")
data(metMUD2)
plotSpectraJS(metMUD2)

## End(Not run)
```
Conversion Between PCA Classes

**Description**

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

**Usage**

q2rPCA(x)

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 placeholder 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>
readJDX

References

https://github.com/bryanhanson/ChemoSpec

See Also

prcomp, princomp

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

call

file Character; the path to the file to be processed.
dehu 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

The details of the JCAMP-DX formats can be found at at http://www.jcamp-dx.org/
https://github.com/bryanhanson/ChemoSpec
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

https://github.com/bryanhanson/ChemoSpec

Examples

data(SrE.IR)
sumSpectra(SrE.IR)
# remove frequencies from one end:
newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR.freq > 3500)
# remove frequencies from both ends at once:
newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR.freq > 3500 |
| SrE.IR.freq < 800)
# remove frequencies from the middle:
newIR <- removeFreq(SrE.IR, rem.freq = SrE.IR.freq > 800)
& SrE.IR$freq < 1000)

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

# after any of these, inspect the results:
sumSpectra(newIR)
check4Gaps(newIR$freq, newIR$data[1,], plot = TRUE)

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. removeGroup will report if it finds extra data elements. These may be per sample
data. If so, the user will need to manually edit them using the indices reported to the console.

Value

A modified object of S3 class Spectra.

Author(s)

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

References

https://github.com/bryanhanson/ChemoSpec

See Also

removeFreq to remove selected frequencies from a Spectra object.

Examples

data(metMUD1)
new1 <- removeSample(metMUD1, rem.sam = 20)
# removes the 20th spectrum/sample
new2 <- removeSample(metMUD1, rem.sam = "Sample_20")
# removes one spectrum/sample with this exact name.
new3 <- removeSample(metMUD1, rem.sam = "Sample")
# removes all samples!

Description

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

Usage

rowDist(x, method)

Arguments

x
  A matrix whose rows will be used for the distance calculation.

method
  A character; one of c("pearson", "correlation", "spearman", "kendall", "euclidean", "max-
 imum", "manhattan", "canberra", "binary", "minkowski")

Details

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

Value

An object of class dist.
r_pcaSpectra

Author(s)

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

r_pcaSpectra  Robust PCA of a Spectra Object

Description

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

Usage

r_pcaSpectra(spectra, choice = "noscale")

Arguments

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

Value

An object of classes "conPCA" and "princomp" (see qRrpca). 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

  
https://github.com/bryanhanson/ChemoSpec

See Also

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

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

Examples

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

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

Usage

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

Arguments

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

Value

A numeric matrix giving the correlation coefficients.

Author(s)

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

References

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

See Also

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

Examples

```r
require("lattice")
data(metMUD2)
sampleDistSpectra(metMUD2, main = "Sample Correlations for metMUD2")
```
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

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

```r
x <- rnorm(100)
sex(x)
seXy(x)
seXy95(x)
seXymad(x)
seXyIqr(x)
```

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

**Description**

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

**Usage**

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

**Arguments**

- `n`: A node in a dendrogram object.
- `spectra`: An object of S3 class “Spectra”.

**Value**

Returns a node with the label size properties set.

**Author(s)**

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

**References**

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

---

**Spectra**  
*Spectra Objects*

**Description**

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

**Structure**

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

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.

Details

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

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

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

https://github.com/bryanhanson/ChemoSpec

See Also

conColsScheme.

Examples

data(metMUD2)
# Original factor encoding:
levels(metMUD2$groups)
# Split those original levels into 2 new ones (re-code them)
new.grps <- list(geneBb = c("B", "b"), geneCc = c("C", "c"))
res <- splitSpectraGroups(metMUD2, new.grps)
str(res) # note two new elements, "geneBb" and "geneCc"
#
# Note that if you want to use a newly created group in
# plotScores and other functions to drive the color scheme
# and labeling, you’ll have to update the groups element:
res$groups <- as.factor(paste(res$geneBb, res$geneCc, sep = ""))

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 `c_pcaSpectra` or `r_pcaSpectra`).
- `pc`: An integer specifying the desired pc plot.
- `tol`: A number describing the fraction of points to be labeled. `tol = 1.0` labels all the points; `tol = 0.05` labels the most extreme 5 percent.
- `...`: Additional parameters to be passed to plotting functions.

Value

A data frame containing the 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


https://github.com/bryanhanson/ChemoSpec
**Examples**

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

---

**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**

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

**Format**

The data is stored as a `Spectra` object.

**Source**

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

**References**

https://github.com/bryanhanson/ChemoSpec

**Examples**

```r
data(SrE.IR)
sumSpectra(SrE.IR)
data(SrE.NMR)
sumSpectra(SrE.NMR)
```
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
https://github.com/bryanhanson/ChemoSpec

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

Examples
data(metMUD1)
sumGroups(metMUD1)
**sumSpectra**

### Summarize a Spectra Object

**Description**

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

**Usage**

```r
csumSpectra(spectra, ...)
```

**Arguments**

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

**Details**

Prior to summarizing, `chkSpectra` is run with `confirm = FALSE`. If there are problems, warnings are issued to the console and the summary is not done. If `sumSpectra` thinks there is a gap between every data point, add the argument `tol = xx` which will pass through to `check4Gaps` and alleviate this problem (which has to do with rounding when subtracting two adjacent frequency values). The `Spectra` object is checked to see if it contains data elements beyond what is required. If so, these extra elements are reported to the console.

**Value**

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

**Author(s)**

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

**References**

[https://github.com/bryanhanson/ChemoSpec](https://github.com/bryanhanson/ChemoSpec)

**Examples**

```r
data(metMUD1)
ssumSpectra(metMUD1)
```
surveySpectra

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

surveySpectra(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
https://github.com/bryanhanson/ChemoSpec

Examples

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

<table>
<thead>
<tr>
<th>Topic</th>
<th>Class</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Topic suites</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Topic clusters</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>chkSpectra, 12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q2rPCA, 59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectra, 67</td>
<td></td>
</tr>
<tr>
<td><em>Topic cluster</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>colLeaf, 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>coordProjCS, 18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>evalClusters, 23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hcaScores, 29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hcaSpectra, 30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclust3D, 38</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclust3dSpectra, 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclustSpectra, 41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotHCA, 47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shrinkLeaf, 67</td>
<td></td>
</tr>
<tr>
<td><em>Topic color</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>colLeaf, 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>colorSymbol, 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>conColScheme, 17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>groupNcolor, 28</td>
<td></td>
</tr>
<tr>
<td><em>Topic datasets</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>metMUD1, 42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SrE.IR, 71</td>
<td></td>
</tr>
<tr>
<td><em>Topic dynamic</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScoresRGL, 53</td>
<td></td>
</tr>
<tr>
<td><em>Topic file</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>files2SpectraObject, 25</td>
<td></td>
</tr>
<tr>
<td><em>Topic hplot</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>baselineSpectra, 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>corSpectra, 19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>loopThruSpectra, 36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plot2Loadings, 46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotHCA, 47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotLoadings, 48</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScores, 49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScores3D, 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScoresRGL, 53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScree, 55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotSpectra, 56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sampleDistSpectra, 65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sPlotSpectra, 70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>surveySpectra, 74</td>
<td></td>
</tr>
<tr>
<td><em>Topic htest</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>aov_pcaSpectra, 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aovPCAloadings, 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aovPCAscores, 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>avgFacLvlS, 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hypTestScores, 32</td>
<td></td>
</tr>
<tr>
<td><em>Topic manip</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>binData, 9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>binSpectra, 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>normSpectra, 43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>normVec, 44</td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeFreq, 61</td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeSample, 62</td>
<td></td>
</tr>
<tr>
<td><em>Topic multivariate</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>aov_pcaSpectra, 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aovPCAloadings, 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aovPCAscores, 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>avgFacLvlS, 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c_pcaSpectra, 22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ChemoSpec-package, 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>coordProjCS, 18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cv_pcaSpectra, 21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>evalClusters, 23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hcaScores, 29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hcaSpectra, 30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hmapSpectra, 31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hypTestScores, 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>makeEllipsoid, 37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclust3D, 38</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclust3dSpectra, 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mclustSpectra, 41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pcaDiag, 45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plot2Loadings, 46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotHCA, 47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotLoadings, 48</td>
<td></td>
</tr>
<tr>
<td></td>
<td>plotScores, 49</td>
<td></td>
</tr>
</tbody>
</table>
INDEX

legend, 49
Logic, 61
loopThruSpectra, 36

makeEllipsoid, 37
mclust3D, 38
mclust3dSpectra, 39
mclustSpectra, 41
metMUD1, 42
metMUD2 (metMUD1), 42

normSpectra, 22, 43
normVec, 44

pcaCV, 21, 22
pcaDiag, 45
pcaDiagplot, 45, 46
PCAgрид, 64
plot2Loadings, 23, 46, 49, 50, 64
plotHCA, 24, 29, 30, 47
plotLoadings, 5, 23, 47, 48, 50, 64
plotScores, 5, 6, 23, 49, 51, 55, 64
plotScores3D, 23, 50, 50, 55, 64
plotScoresCor, 37, 52
plotScoresDecoration, 34, 53
plotScoresRGL, 23, 35, 50, 51, 53, 64
plotScree, 23, 50, 55, 64
plotScree2 (plotScree), 55
plotSpectra, 8, 56, 57, 58
plotSpectraJS, 57, 57
prcomp, 23, 29, 32, 40, 41, 43, 45, 46, 48–50, 53–55, 59, 60
princomp, 59, 60

q2rPCA, 59, 64
r2qPCA (q2rPCA), 59
r_pcaSpectra, 3, 23, 29, 46, 48–50, 53, 55, 64, 70
readJDX, 26, 60
removeFreq, 12, 43, 61, 63
removeGroup (removeSample), 62
removeSample, 12, 62, 72
rgl, 53
rowDist, 29, 30, 63, 65

sampleDistSpectra, 65
seX (seXy), 66
seXy, 66
seXy95 (seXy), 66

seXyIqr (seXy), 66
seXyMad (seXy), 66
shrinkLeaf, 67
Spectra, 4–6, 8, 10–13, 15–17, 19, 21, 22,
24–32, 36, 39–41, 43, 45–50, 53–57,
61–65, 67, 68–74
splitSpectraGroups, 16, 32, 33, 68
sPlotSpectra, 23, 64, 70, 70
SrE.IR, 71
SrE.NMR (SrE.IR), 71
stats, 63
sumGroups, 16, 28, 68, 72
sumSpectra, 68, 73
surveySpectra, 74

validObject, 13