Package: ChemoSpec (via r-universe)

August 31, 2024

Type Package

Title Exploratory Chemometrics for Spectroscopy

Version 6.1.10

Date 2024-02-03

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

License GPL-3

Depends R (>= 3.5), ChemoSpecUtils (>= 1.0)

Imports stats, utils, grDevices, reshape2, readJDX (>= 0.6), patchwork, ggplot2, plotly, magrittr

Suggests IDPmisc, knitr, js, NbClust, clusterCrit, lattice, baseline, mclust, pls, R.utils, RColorBrewer, seriation, MASS, grid, pcaPP, jsonlite, signal, speaq, tinytest, elasticnet, irlba, rmarkdown, bookdown, chemometrics, hyperSpec, amap, gsubfn, roxut

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

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

ByteCompile TRUE

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.1

Roxygen list(roclets = c(``namespace'', ``rd'', ``roxut::tests_roclet''), markdown = TRUE)

NeedsCompilation no

Repository https://bryanhanson.r-universe.dev

RemoteUrl https://github.com/bryanhanson/chemospec

RemoteRef HEAD

RemoteSha fdaf42bb7e7c4c18eed95f1997ba11df0da16976

Contents

`ChemoSpec-package` *Exploratory Chemometrics for Spectroscopy*

Description

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

Author(s)

Bryan A. Hanson (DePauw University), Tejasvi Gupta & Matthew J. Keinsley.

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

aovPCAloadings *Plot aovPCAscores Loadings of a Spectra Object*

Description

Uses the results from aovPCAscores to plot the corresponding loadings.

Usage

```
aovPCAloadings(spectra, PCA, submat = 1, loads = 1, ref = 1, ...)
```

Arguments

spectra	An object of S3 class Spectra().
PCA	List of pca results created by aov_pcaSpectra.
submat	Integer. Selects list element submat from PCA which is a list of PCA results, each corresponding to the computation in aov_pcaSpectra.
loads	An integer vector giving the loadings to plot.
ref	An integer specifying the reference spectrum to plot, which appears at the bottom of the plot.
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

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

Author(s)

Bryan A. Hanson (DePauw University), Matthew J. Keinsley.

References

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

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

See Also

An example using this function can be seen in aov_pcaSpectra. See also plotLoadings. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

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

Description

Uses the results from aov_pcaSpectra to plot the scores. Argument submat is used to select PCA results from among those stored in argument PCA.

Usage

```
aovPCAscores(
  spectra,
  so,
  submat = 1,
  ellipse = "none",
  tol = "none",
  use.sym = FALSE,
  leg.loc = "topright",
  ...
)
```

Arguments

spectra	An object of S3 class Spectra().
so	*List* of pca results created by aov_pcaSpectra.
submat	Integer. Selects list element submat from PCA which is a list of PCA results, each corresponding to the computation in aov_pcaSpectra.
ellipse	A character vector specifying the type of ellipses to be plotted. One of c("both", "none", "cls", "rob"). cls specifies classical confidence ellipses, rob specifies robust confidence ellipses. An ellipse is drawn for each group unless there are three or fewer samples in the group.
tol	A number describing the fraction of points to be labeled. tol = 1.0 labels all the points; tol = 0.05 labels *approximately* the most extreme 5 percent. Set to 'none' to completely suppress labels. Note that a simple approach based upon quantiles is used, assumes that both x and y are each normally distributed, and treats x and y separately. Thus, this is not a formal treatment of outliers, just a means of labeling points. Groups are lumped together for the computation.
use.sym	A logical; if TRUE, the color scheme is set to black and the points plotted with symbols. Applies only to Spectra objects.
leg.loc	Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
...	Additional parameters to be passed to plotScores. For example, you can plot confidence ellipses this way. Note that ellipses are drawn based on the groups in spectra$groups, but the separation done by aov_pcaSpectra is based on argument fac. These may not correspond, but you can edit spectra$groups to match if necessary.

Value

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

- **base** A data frame or list containing the data plotted. Assign the value and run `str()` or `names()` on it to see what it contains. Side effect is a plot.
- **ggplot2** The plot is displayed, and a `ggplot2` plot object is returned if the value is assigned. The plot can be modified in the usual `ggplot2` manner. If you want the plotted values, you can access them via the base graphics mode.

Author(s)

Bryan A. Hanson (DePauw University), Matthew J. Keinsley.

References

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

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

See Also

The use of this function can be seen in `aov_pcaSpectra`. See also `plotScores`. Additional documentation at `https://bryanhanson.github.io/ChemoSpec/`

`aov_pcaSpectra` *ANOVA-PCA Analysis of Spectra Data*

Description

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

Usage

```
aov_pcaSpectra(spectra, fac, type = "class", choice = NULL, showNames = TRUE)
```

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.
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.
showNames	Logical. Show the names of the submatrices in the console.

Value

A list of PCA results, one for each computed submatrix.

Author(s)

Bryan A. Hanson (DePauw University), Matthew J. Keinsley.

References

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

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

See Also

The output of this function is used in used in aovPCAscores and aovPCAloadings. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
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
PCAs <- aov_pcaSpectra(mM3, fac = c("geneBb", "geneCc"))
```

```
p1 <- aovPCAscores(mM3, PCAs, submat = 1, ellipse = "cls")
p1 <- p1 + ggtitle("aovPCA: B vs b")
p1

p2 <- aovPCAscores(mM3, PCAs, submat = 2)
p2 <- p2 + ggtitle("aovPCA: C vs c")
p2

p3 <- aovPCAscores(mM3, PCAs, submat = 3)
p3 <- p3 + ggtitle("aovPCA: Interaction Term")
p3

p4 <- aovPCAloadings(spectra = mM3, PCA = PCAs)
p4 <- p4 + ggtitle("aov_pcaSpectra: Bb Loadings")
p4

## End(Not run)
```

`averageReplicates` *Average Replicates in a Spectra Object*

Description

Average the replicates in a `Spectra` object and return a new Spectra object with fewer samples. One should probably not do this until each individual sample has been visualized for quality control, in case it is a potential outlier.

Usage

```
averageReplicates(spectra, uniq)
```

Arguments

`spectra`	An object of S3 class `Spectra()`.
`uniq`	Character. A character vector containing strings representing unique sample identifiers. The sample names will be searched for these strings, and all samples matching a given string will be averaged and put into a new Spectra object. For example, consider the case where samples are named `S_1_01`, `S_1_02`, `...`, `S_2_01`, `S_2_02`, `...` where `_01` and so forth signifies replicates of a particular sample. With `uniq = c("S_1", "S_2")` all `S_1` replicates will be averaged and all `S_2` replicates will be averaged. N.B. the strings will be used as regex pattern and grepped.

Value

An object of S3 class `Spectra`.

Author(s)

Bryan A. Hanson (DePauw University).

Examples

```
data(SrE.IR)
averaged <- averageReplicates(SrE.IR, uniq = c("EPO", "OO", "adSrE", "pSrE"))
sumSpectra(SrE.IR)
sumSpectra(averaged)
```

---

baselineSpectra *Baseline Correction of a Spectra Object*

Description

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

Usage

```
baselineSpectra(spectra, int = TRUE, retC = FALSE, show = 1, ...)
```

Arguments

spectra	An object of S3 class Spectra().
int	Logical; if TRUE, do the correction interactively using widgets. No results are saved. Use this for inspection and exploration only. Automatically overriden to FALSE if interactive() returns FALSE. This is necessary so that plots appear in vignettes etc.
retC	Logical: shall the baseline-corrected spectra be returned in the Spectra object?
show	Integer. A (single) sample number for which you wish to see the results of the baseline correction. By "sample number" we mean the rows in the spectra$data matrix. To find a specific sample type spectra$names to see which row contains that sample.
...	Other arguments passed downstream. The relevant ones can be found in baseline. Be sure to pay attention to argument method as you will probably want to use it. You can also use method = "linear" for a simple linear fit, see Details.

Details

In plots using methods from the baseline package, the x axis ticks give the data point index, not the original values from your data. Note that you cannot zoom the non-interactive display of corrected spectra because the underlying function hardwires the display. Try the interactive version instead (int = TRUE), or use plotSpectra on the corrected data. In addition to the methods provided by baseline, you can also use method = "linear". This correction is handled locally, and is very

simple: a line is drawn from the first data point to the last, and this becomes the new baseline. This is most suitable for cases in which the baseline rises or falls steadily, as is often seen in chromatograms.

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "baseline" for this example
if (requireNamespace("baseline", quietly = TRUE)) {
  data(SrE.IR)
  temp <- baselineSpectra(SrE.IR, int = FALSE, method = "modpolyfit")
}
```

---

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()`.
`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).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

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

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via check4Gaps.

---

chkGraphicsOpt — *Check the Graphics Output Option/Mode*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via chkGraphicsOpt.

chkSpectra *Verify the Integrity of a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via chkSpectra.

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

Description

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

Usage

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

Arguments

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

Value

A modifed Spectra object.

Author(s)

Bryan A. Hanson (DePauw University).

References

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package speaq for this example
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
if (requireNamespace("speaq", quietly = TRUE)) {
  library("ggplot2")
  data(alignMUD)

  p1 <- plotSpectra(alignMUD, which = 1:20, lab.pos = 4.5, offset = 0.1,
    yrange = c(0, 5000), amp = 500)
  p1 <- p1 + ggtitle("Misaligned NMR Spectra") +
    coord_cartesian(xlim = c(1.5, 1.8), ylim = c(0, 1900))
  p1

  aMUD <- clupaSpectra(alignMUD)

  p2 <- plotSpectra(aMUD, which = 1:20, lab.pos = 4.5, offset = 0.1,
    yrange = c(0, 5000), amp = 500)
  p2 <- p2 +  ggtitle("Aligned NMR Spectra") +
    coord_cartesian(xlim = c(1.5, 1.8), ylim = c(0, 1900))
  p2
}
```

---

colorSymbol *Color and Symbols in ChemoSpec and ChemoSpec2D*

Description

This help page serves both ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via colorSymbol.

---

conColScheme *Change the Color Scheme of a Spectra or Spectra2D Object*

Description

This help page serves both ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via conColScheme.

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

Description

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

Usage

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

Arguments

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

Value

Invisibly, a list as described in pcaCV. Side effect is a plot.

Author(s)

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

References

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

See Also

pcaCV for the underlying function. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "pls" for this example
if (requireNamespace("pls", quietly = TRUE)) {
  data(SrE.IR)
  pca <- cv_pcaSpectra(SrE.IR, pcs = 5)
}
```

---

c_pcaSpectra *Classical PCA of Spectra Objects*

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto"). "autoscale" is called "standard normal variate" or "correlation matrix PCA" in some literature.
cent	Logical: whether or not to center the data. Always center the data unless you know it to be already centered.

Details

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

Value

An object of class prcomp, modified to include a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (used to annotate plots).

Author(s)

Bryan A. Hanson (DePauw University).

References

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

See Also

prcomp for the underlying function, s_pcaSpectra for sparse PCA calculations, r_pcaSpectra for robust PCA calculations, irlba_pcaSpectra for PCA via the IRLBA algorithm. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

For displaying the results, ChemoSpecUtils::plotScree(), ChemoSpecUtils::plotScores(), plotLoadings(), plot2Loadings(), sPlotSpectra().

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(metMUD1)
pca <- c_pcaSpectra(metMUD1)

p1 <- plotScree(pca)
p1

p2 <- plotScores(metMUD1, pca, pcs = c(1, 2), ellipse = "cls", tol = 0.05)
p2 <- p2 + ggtitle("Scores: metMUD1 NMR Data")
p2

p3 <- plotLoadings(metMUD1, pca, loads = 1:2, ref = 1)
p3 <- p3 + ggtitle("Loadings: metMUD1 NMR Data")
p3

## End(Not run)
```

`evalClusters` *Evaluate or Compare the Quality of Clusters Quantitatively*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

References

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

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. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
## Not run:
data(metMUD2)

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

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

## End(Not run)
```

---

files2SpectraObject *Import Data into a Spectra Object*

Description

These functions import data into a Spectra object. For "csv-like" files they use read.table, so they are very flexible in regard to file formatting. **Be sure to see the ...argument below for important details you need to provide.** files2SpectraObject can also read JCAMP-DX files and will do so if fileExt is any of "dx", "DX", "jdx" or "JDX".

Usage

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

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

Arguments

`gr.crit`	Group Criteria. A vector of character strings which will be searched for among the file/sample names in order to assign an individual spectrum to group membership. This is done using grep, so characters like "." (period/dot) do not have their literal meaning (see below). Warnings are issued if there are file/sample names that don't match entries in `gr.crit` or there are entries in `gr.crit` that don't match any file names.
`gr.cols`	Group Colors. See `colorSymbol` for some options. One of the following: • Legacy behavior and the default: The word `"auto"`, in which case up to 8 colors will be automatically assigned from package `RColorBrewer` `Set1`. • `"Col7"`. A unique set of up to 7 colorblind-friendly colors is used. • `"Col8"`. A unique set of up to 8 colors is used. • `"Col12"`. A mostly paired set of up to 12 colors is used. • A vector of acceptable color designations with the same length as `gr.crit`. 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. For `Col12` you should pay careful attention to the order of `gr.crit` in order to match up colors.
`freq.unit`	A character string giving the units of the x-axis (frequency or wavelength).
`int.unit`	A character string giving the units of the y-axis (some sort of intensity).

descrip	A character string describing the data set that will be stored. This string is used in some plots so it is recommended that its length be less than about 40 characters.
fileExt	A character string giving the extension of the files to be processed. regex strings can be used. For instance, the default finds files with either ".csv" or ".CSV" as the extension. Matching is done via a grep process, which is greedy. See also the "Advanced Tricks" section.
out.file	A file name. The completed object of S3 class Spectra will be written to this file.
debug	Logical. Applies to files2SpectraObject only. Set to TRUE for troubleshooting when an error is thrown during import. In addition, values of 1-5 will work when importing a JCAMP-DX file via fileExt = "\.jdx" etc. These will be passed through to the readJDX function. See there for much more info on importing JCAMP-DX files.
...	Arguments to be passed to read.table, list.files or readJDX; see the "Advanced Tricks" section. For read.table, **You MUST supply values for** sep**,** dec **and** header **consistent with your file structure, unless they are the same as the defaults for** read.table.
in.file	Character. Applies to matrix2SpectraObject only. Input file name, including extension. Can be a vector of file names.
chk	Logical. Applies to matrix2SpectraObject only. Should the Spectra object be checked for integrity? If you are having trouble importing your data, set this to FALSE and do str(your object) to troubleshoot.

Value

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

Functions

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

files2SpectraObject

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

If fileExt contains any of "dx", "DX", "jdx" or "JDX", then the files will be processed by readJDX. Consider setting debug = TRUE, or debug = 1 etc for this format, as there are many options for JCAMP, and many are untested. See readJDX for options and known limitations.

matrix2SpectraObject

This function takes one or more csv-like files, containing frequencies in the first column, and samples in additional columns, and processes it into a Spectra object. The file MUST have a header row which includes the sample names. There need not be a header for the first (frequency) column. If more than one file given, they must all have the same frequency entries.

gr.crit and Sample Name Gotchas

The matching of gr.crit against the sample file names (in files2SpectraObject) or column headers/sample names (in matrix2SpectraObject) is done one at a time, in order, using grep. While powerful, this has the potential to lead to some "gotchas" in certain cases, noted below.

Your file system may allow file/sample names which R will not like, and will cause confusing behavior. File/sample names become variables in ChemoSpec, and R does not like things like "-" (minus sign or hyphen) in file/sample names. A hyphen is converted to a period (".") if found, which is fine for a variable name. However, a period in gr.crit is interpreted from the grep point of view, namely a period matches any single character. At this point, things may behave very differently than one might hope. See make.names for allowed characters in R variables and make sure your file/sample names comply.

The entries in gr.crit must be mutually exclusive. For example, if you have files with names like "Control_1" and "Sample_1" and use gr.crit = c("Control", "Sample") groups will be assigned as you would expect. But, if you have file names like "Control_1_Shade" and "Sample_1_Sun" you can't use gr.crit = c("Control", "Sample", "Sun", "Shade") because each criteria is grepped in order, and the "Sun/Shade" phrases, being last, will form the basis for your groups. Because this is a grep process, you can get around this by using regular expressions in your gr.crit argument to specify the desired groups in a mutually exclusive manner. In this second example, you could use gr.crit = c("Control(.*)Sun", "Control(.*)Shade", "Sample(.*)Sun", "Sample(.*)Shade") to have your groups assigned based upon both phrases in the file names.

To summarize, gr.crit is used as a grep pattern, and the file/sample names are the target. Make sure your file/sample names comply with make.names.

Finally, samples whose names are not matched using gr.crit are still incorporated into the Spectra object, but they are not assigned a group or color. Therefore they don't plot, but they do take up space in a plot! A warning is issued in these cases, since one wouldn't normally want a spectrum to be orphaned this way.

All these problems can generally be identified by running sumSpectra once the data is imported.

Advanced Tricks

The ... argument can be used to pass any argument to read.table or list.files. This includes the possibility of passing arguments that will cause trouble later, for instance na.strings in read.table. While one might successfully read in data with NA, it will eventually cause problems. The intent of this feature is to allow one to recurse a directory tree containing the data, and/or to specify a starting point other than the current working directory. So for instance if the current working directory is not the directory containing the data files, you can use path = "my_path" to point to the desired top-level directory, and recursive = TRUE to work your way through a set of subdirectories. In addition, if you are reading in JCAMP-DX files, you can pass arguments to readJDX via ..., e.g. SOFC = FALSE. Finally, while argument fileExt appears to be a file extension (from its

name and the description elsewhere), it's actually just a grep pattern that you can apply to any part of the file name if you know how to construct the proper pattern.

Author(s)

Bryan A. Hanson (DePauw University).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/, as well as updateGroups.

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")

wd <- getwd() # save current location
setwd(tempdir())

# Grab an included file & move to a temporary directory
tf <- system.file("extdata/PCRF.jdx", package = "ChemoSpec")
chk <- file.copy(from = tf, to = basename(tf))

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

setwd(wd) # restore working directory

## End(Not run)
```

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via hcaScores.

hcaSpectra *Plot HCA Results of a Spectra Object*

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
c.method	A character string describing the clustering method; must be acceptable to hclust.
d.method	A character string describing the distance calculation method; must be acceptable as a method in rowDist.
use.sym	A logical; if true, use no color and use lower-case letters to indicate group membership.
leg.loc	Character; if "none" no legend will be drawn. Otherwise, any string acceptable to legend.
...	Other parameters to be passed to the plotting functions.

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

hclust for the underlying function. hcaScores for similar analysis of PCA scores from a Spectra object. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "amap" for this example
if (requireNamespace("amap", quietly = TRUE)) {
  data(SrE.IR)
  myt <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(IR ~ Spectra))
  res <- hcaSpectra(SrE.IR, main = myt)
}
```

`hmapSpectra` *Seriated Heat Map for a Spectra Object*

Description

Creates a heat map with marginal dendrograms using seriation procedures. Heirchical cluster analysis is followed by re-ordering the clusters in a coordinated way across each dimension (controlled by argument `method`, see `hmap`). The vignette for package **seriation** has more details.

Usage

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

Arguments

`spectra`	An object of S3 class `Spectra()`.
`...`	Additional arguments to be passed downstream. A great deal of control is available - check `hmap` for details. Most of the control actually derives from the `heatmap` function in package **stats**. See the examples.

Value

A list as described in `hmap`. Side effect is a plot.

Note

The underlying `stats::heatmap` function does certain things automatically for the user. For instance, if you pass a vector of names to argument `row_labels` it is automatically reordered using the `rowInd` vector that is computed. The user does not need to do the reordering. Another example is the labeling of the columns. The labels are automatically turned 90 degrees, and not every column is labeled to keep things readible.

Interpretation

Looking at the 2nd `hmapSpectra` example, and keeping in mind the nature of the sample (see `SrE.IR`), the most similar samples based on the ester peaks (~1740), are in the lower right corner of the heatmap. These are the two outlier samples, composed of triglycerides which contain the ester functional group (and no detectable carboxylic acid). The most similar samples based on the

carboxylic acid peaks (~1710) are in the upper left corner. These samples are mostly from the "pure" extract group, according to the manufacturer's label. These samples have a modest to low amount of the ester functional group, which indicates dilution (or adulteration if you like). In fact, the first two samples (NP_adSrE and NR_pSrE) are the two samples with the smallest ester peaks (see first plot in the examples). This suggests that NP_adSrE was diluted only a little with added olive oil.

Author(s)

Bryan A. Hanson (DePauw University).

See Also

hmap which will get you to the package (there is no package index page); the vignette is a good place to begin (browseVignettes("seriation")). Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "seriation" for this example
if (requireNamespace("seriation", quietly = TRUE)) {
  data(SrE.IR)

  # Let's look just at the carbonyl region
  IR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 1775 | SrE.IR$freq < 1660)
  p <- plotSpectra(IR, which = 1:16, lab.pos = 1800)

  # Defaults, except for color scheme:
  res <- hmapSpectra(IR, col = heat.colors(5))

  # Label samples and frequencies by passing arguments to stats:heatmap
  # Also make a few other nice plot adjustments
  res <- hmapSpectra(IR,
    col = heat.colors(5),
    row_labels = IR$names, col_labels = as.character(round(IR$freq)),
    margins = c(4, 6)
    )
}
```

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

splitSpectraGroups which can be used to create additional factor elements in the Spectra object, which can then be used with this function. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
data(metMUD2)

# Original factor encoding:
levels(metMUD2$groups)

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

# Now do the PCA and anova, with 3 ways to see the results
pca <- c_pcaSpectra(mM3)
res <- hypTestScores(mM3, pca, fac = c("geneBb", "geneCc"))
res
summary(res)
```

```
summary.aov(res)

# You can also call this function on the existing groups:
res <- hypTestScores(metMUD2, pca, fac = "groups")
```

`irlba_pcaSpectra` *IRLBA PCA of Spectra Objects*

Description

A wrapper which carries out IRLBA 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`. The data can be supplied already centered if desired.

Usage

```
irlba_pcaSpectra(spectra, choice = "noscale", n = 3, center = TRUE, ...)
```

Arguments

`spectra`	An object of S3 class `Spectra()`.
`choice`	A character string indicating the choice of scaling. One of `c("noscale", "autoscale", "Pareto")`. `"autoscale"` is called "standard normal variate" or "correlation matrix PCA" in some literature.
`n`	Integer. The number of components desired.
`center`	Logical. Should the data be centered? Data must be centered for PCA, either before arriving here or via this argument.
`...`	Other parameters to be passed to `irlba`.

Details

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

Value

A modified object of class `prcomp` and `computed_via_irlba`, which includes a list element called `$method`, a character string describing the pre-processing carried out and the type of PCA performed (used to annotate plots).

Author(s)

Bryan A. Hanson (DePauw University).

References

J. Baglama and L. Reichel, "Augmented Implicitly Restarted Lanczos Bidiagonalization Methods" *SIAM J. Sci. Comput.* (2005).

See Also

prcomp_irlba for the underlying function, c_pcaSpectra for classical PCA calculations, r_pcaSpectra for robust PCA calculations, s_pcaSpectra for sparse PCA calculations. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

For displaying the results, ChemoSpecUtils::plotScree(), ChemoSpecUtils::plotScores(), plotLoadings(), plot2Loadings(), sPlotSpectra().

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.NMR)
pca <- irlba_pcaSpectra(SrE.NMR)

p1 <- plotScree(pca)
p1

p2 <- plotScores(SrE.NMR, pca, pcs = c(1, 2), ellipse = "cls", tol = 0.05)
p2 <- p2 + ggtitle("Scores: SrE NMR Data")
p2

p3 <- plotLoadings(SrE.NMR, pca, loads = 1:2, ref = 1)
p3 <- p3 + ggtitle("Loadings: SrE NMR Data")
p3

## End(Not run)
```

---

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

Description

This function conducts an mclust analysis of the PCA results of a Spectra object and displays the results in 3D. Classical or robust confidence ellipses can be added if desired. Improperly classified data points can be marked. The interactive plot is made via plotly and appears in a browser window. 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.

Usage

```
mclust3dSpectra(
  spectra,
  pca,
  pcs = 1:3,
  ellipse = TRUE,
  rob = FALSE,
  cl = 0.95,
  frac.pts.used = 0.8,
  truth = NULL,
  ...
)
```

Arguments

spectra	An object of S3 class Spectra().
pca	An object of class prcomp.
pcs	An integer vector describing which PCs to use.
ellipse	Logical indicating if confidence ellipses should be drawn.
rob	Logical; if ellipse = TRUE, indicates that robust confidence ellipses should be drawn. If FALSE, classical confidence ellipses are drawn.
cl	A number indicating the confidence interval for the ellipse.
frac.pts.used	If ellipse = TRUE and rob = TRUE, a number indicating the fraction of the data points to be considered "good" and thus used to compute the robust confidence ellipse.
truth	A character vector indicating the known group membership for reach row of the PC scores. Generally this would be spectra$groups.
...	Arguments to be passed to mclust.

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

Mclust for background on the method. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
## Not run:
require(mclust)
data(metMUD1)
pca <- c_pcaSpectra(metMUD1)
mclust3dSpectra(metMUD1, pca)

# show mis-classified points
mclust3dSpectra(metMUD1, pca, truth = metMUD1$groups)

## End(Not run)
```

`mclustSpectra` *mclust Analysis of a Spectra Object PCA Results*

Description

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

Usage

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

See Also

Mclust for background on the method. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
## Not run:
require("mclust")
data(metMUD1)
cls <- c_pcaSpectra(metMUD1, choice = "autoscale")

p <- plotScores(metMUD1, cls)

mclustSpectra(metMUD1, cls, plot = "BIC")
mclustSpectra(metMUD1, cls, plot = "proj")
mclustSpectra(metMUD1, cls, plot = "errors", truth = metMUD1$groups)

## End(Not run)
```

---

metMUD1 *Made Up NMR Data Sets*

Description

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

Format

The data is stored as a Spectra object.

Details

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

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

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

Author(s)

Bryan A. Hanson (DePauw University).

Source

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

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

---

normSpectra *Normalize a Spectra Object*

Description

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

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
method	One of c("PQN", "TotInt", "Range", "zero2one") giving the method for normalization.
RangeExpress	A vector of logicals (must be of length(Spectra$freq)). This vector should be TRUE for the frequency range you want to serve as the basis for norming, and FALSE otherwise. The entire spectrum will be divided by the sum of the TRUE range. See the examples.

Value

An object of S3 class Spectra().

Author(s)

Bryan A. Hanson (DePauw University).

References

- Probabalistic Quotient Normalization is reported in F. Dieterle et al. Analytical Chemistry vol. 78 pages 4281-4290 (2006).
- The exact same mathematics are called "median fold change normalization" by Nicholson's group, reported in K. A. Veselkov et. al. Analytical Chemistry vol. 83 pages 5864-5872 (2011).
- Corriea et al. "1H NMR Signals from Urine Excreted Protein are a Source of Bias in Probablistic Quotient Normalization" Analytical Chemistry vol. 94 pages 6919-6923 (2022).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.IR)

# Reference spectrum before normalization
p1 <- plotSpectra(SrE.IR) + ggtitle("Original Spectrum")
p1

# Default PQN normalization
res1 <- normSpectra(SrE.IR)
p2 <- plotSpectra(res1) + ggtitle("PQN Normalization")
p2
```

```
# Norm over carbonyl region
RE <- SrE.IR$freq > 1650 & SrE.IR$freq < 1800
res2 <- normSpectra(SrE.IR, method = "Range", RangeExpress = RE)
p3 <- plotSpectra(res2) + ggtitle("Normalized to Carbonyl Peaks")
p3

# Check numerically
rowSums(res2$data[, RE]) # compare to rowSums(SrE.IR$data[,RE])
```

---

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

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
pca	An object of class prcomp 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.
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

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

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

- **base** A data frame or list containing the data plotted. Assign the value and run `str()` or `names()` on it to see what it contains. Side effect is a plot.
- **ggplot2** The plot is displayed, and a `ggplot2` plot object is returned if the value is assigned. The plot can be modified in the usual `ggplot2` manner. If you want the plotted values, you can access them via the base graphics mode.

Author(s)

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

References

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

See Also

`pcaDiagplot` in package `chemometrics` for the underlying function. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR, choice = "noscale")
p1 <- pcaDiag(SrE.IR, pca, pcs = 2, plot = "OD") + ggtitle("OD Plot")
p1
p2 <- pcaDiag(SrE.IR, pca, pcs = 2, plot = "SD") + ggtitle("SD Plot")
p2
```

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

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
pca	An object of class prcomp, modified to include a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (it appears on the plot). This is automatically provided if ChemoSpec functions c_pcaSpectra or r_pcaSpectra were used to create pca.
loads	A vector of two integers specifying which loading vectors to plot.
tol	A number describing the fraction of points to be labeled. tol = 1.0 labels all the points; tol = 0.05 labels *approximately* the most extreme 5 percent. Set to 'none' to completely suppress labels. Note that a simple approach based upon quantiles is used, assumes that both x and y are each normally distributed, and treats x and y separately. Thus, this is not a formal treatment of outliers, just a means of labeling points. Groups are lumped together for the computation.
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

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

Author(s)

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

See Also

See plotLoadings to plot one loading against the original variable (frequency) axis. See sPlotSpectra for a different approach. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR)
myt <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(IR ~ Spectra))
p <- res <- plot2Loadings(SrE.IR, pca, loads = c(1, 2), tol = 0.001)
p <- p + ggtitle(myt)
p
```

`plot3dScores` *3D PCA Score Plot for a Spectra Object*

Description

Creates an interactive 3D plot of PCA scores from the analysis of a `Spectra` object, color coded according the to scheme stored in the object. The plot is created by `plotly` and appears in a browser window.

Usage

```
plot3dScores(
  spectra,
  pca,
  pcs = c(1:3),
  ellipse = TRUE,
  rob = FALSE,
  cl = 0.95,
  frac.pts.used = 0.8
)
```

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.

Value

None. Side effect is a plot in a browser window.

Author(s)

Bryan A. Hanson (DePauw University).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
if (interactive()) {
  data(metMUD1)
  pca <- c_pcaSpectra(metMUD1, choice = "noscale")
  p <- plot3dScores(metMUD1, pca)
  p
}
```

---

plotLoadings *Plot PCA Loadings for a Spectra Object*

Description

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

Usage

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

Arguments

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

Value

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

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

Author(s)

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

See Also

c_pcaSpectra for an example. See plot2Loadings to plot two loadings against each other, and sPlotSpectra for an alternative approach. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via plotScores.

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via plotScree.

plotSpectra *Plot Spectra Object*

Description

Plots the spectra stored in a Spectra object. Spectra may be plotted offset or stacked. The vertical scale is controlled by a combination of several parameters.

Usage

```
plotSpectra(
  spectra,
  which = c(1),
  yrange = range(spectra$data),
  offset = 0,
  amplify = 1,
  lab.pos = mean(spectra$freq),
  showGrid = TRUE,
  leg.loc = "none",
  ...
)
```

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 to overlay the spectra.
amplify	A number specifying an amplification factor to be applied to all spectra. Useful for magnifying spectra so small features show up (though large peaks will then be clipped, unless you zoom on the x axis).
lab.pos	A number (in frequency units) giving the location of a label for each spectrum. Generally, pick an area that is clear in all spectra plotted. If no label is desired, set lab.pos = "none".
showGrid	Logical. Places light gray vertical lines at each tick mark if TRUE.
leg.loc	Either a list with elements x and y, or a string like 'topright'. Values in a list should be on [0,1], i.e. the lower left of the plot area is 0,0 and the upper right is 1,1. Allowed string values are those described in graphics::legend() under 'Details'. A value of 'none' is acceptable as well.
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

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

Author(s)

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

See Also

plotSpectraJS for the interactive version. See GraphicsOptions for more information about the graphics options. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(metMUD1)

p1 <- plotSpectra(metMUD1, which = c(10, 11), yrange = c(0, 1.5),
  offset = 0.06, amplify = 10, lab.pos = 0.5)
p1 <- p1 + ggtitle("metMUD1 NMR Data")
p1

# Add a legend at x, y coords
p2 <- plotSpectra(metMUD1, which = c(10, 11), yrange = c(0, 1.5),
  offset = 0.06, amplify = 10, lab.pos = 0.5, leg.loc = list(x = 0.8, y = 0.8))
p2 <- p2 + ggtitle("metMUD1 NMR Data")
p2
```

---

plotSpectraDist *Plot the Distance Between Spectra and a Reference Spectrum in a Spectra Object*

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
method	Character. Any method acceptable to rowDist.
ref	Integer. The spectrum to be used as a reference.
labels	Logical. Shall the points be labeled?
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

- **base** A data frame or list containing the data plotted. Assign the value and run str() or names() on it to see what it contains. Side effect is a plot.
- **ggplot2** The plot is displayed, and a ggplot2 plot object is returned if the value is assigned. The plot can be modified in the usual ggplot2 manner. If you want the plotted values, you can access them via the base graphics mode.

Author(s)

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

See Also

To compare all spectra simultaneously in a heatmap, see sampleDist. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "amap" to use this function
if (requireNamespace("amap", quietly = TRUE)) {
  # This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
  library("ggplot2")
  data(SrE.NMR)
  txt1 <- paste("Distance from", SrE.NMR$names[1])
  txt2 <- paste("Rank Distance from", SrE.NMR$names[1])
  p <- plotSpectraDist(SrE.NMR)
  p <- p + labs(title = txt1, xlab = txt2, ylab = txt2) +
           coord_cartesian(ylim = c(0, 1.1), xlim = c(0, 16))
p
}
```

plotSpectraJS *Plot a Spectra Object Interactively*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Browser Choice

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

RStudio Viewer

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

Browser Choice (Mac)

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

Browser Choice & Performance

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

`plotSpectra` for non-interactive plotting. Details about `d3.js` are at `https://d3js.org`. Additional documentation at `https://bryanhanson.github.io/ChemoSpec/`

Examples

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

---

removeFreq *Remove Frequencies from a Spectra or Spectra2D Object*

Description

This function is used by `ChemoSpec` and `ChemoSpec2D`, but is formally part of `ChemoSpecUtils`. You can access full documentation via `removeFreq`.

---

removeGroup *Remove Groups from a Spectra or Spectra2D Object*

Description

This function is used by `ChemoSpec` and `ChemoSpec2D`, but is formally part of `ChemoSpecUtils`. You can access full documentation via `removeGroup`.

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via removeSample.

reviewAllSpectra — *Review All the Spectra in a Spectra Object*

Description

Utility to review all spectra in a Spectra object. Output depends upon the graphics output choice.

base: Plots each spectrum 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.

ggplot2: All the spectra are plotted in a single column.

Usage

```
reviewAllSpectra(spectra, ...)
```

Arguments

spectra	An object of S3 class Spectra().
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

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

Author(s)

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

See Also

See GraphicsOptions for more information about the graphics options. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# Because there are 16 spectra in this data set, you probably want to
# expand the height of the graphics device to see the spectra clearly.
# This example assumes the graphics output is set to ggplot2 or plotly (see ?GraphicsOptions).
# If you do options(ChemoSpecGraphics == "plotly") you'll get the results
# in a web page, which is particularly convenient.
library("ggplot2")
data(metMUD1)
p <- reviewAllSpectra(metMUD1)
p
```

---

rowDist *Compute Distance Between Rows of a Matrix*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via rowDist.

---

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 converted_from_princomp and prcomp. It includes a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (used to annotate plots).

Author(s)

Bryan A. Hanson (DePauw University).

References

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

See Also

PCAgrid for the underlying function, c_pcaSpectra for classical PCA calculations, s_pcaSpectra for sparse PCA calculations, irlba_pcaSpectra for PCA via the IRLBA algorithm. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

For displaying the results, ChemoSpecUtils::plotScree(), ChemoSpecUtils::plotScores(), plotLoadings(), plot2Loadings(), sPlotSpectra().

https://bryanhanson.github.io/ChemoSpec/

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(metMUD1)
pca <- r_pcaSpectra(metMUD1)

p1 <- plotScree(pca)
p1

p2 <- plotScores(metMUD1, pca, pcs = c(1, 2), ellipse = "cls", tol = 0.05)
p2 <- p2 + ggtitle("Scores: metMUD1 NMR Data")
p2

p3 <- plotLoadings(metMUD1, pca, loads = 1:2, ref = 1)
p3 <- p3 + ggtitle("Loadings: metMUD1 NMR Data")
p3

## End(Not run)
```

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via sampleDist.

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

Description

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

Usage

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

Arguments

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

Value

A object of class Spectra.

Author(s)

Bryan A. Hanson (DePauw University).

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# You need to install package "signal" for this example
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
if (requireNamespace("signal", quietly = TRUE)) {
  library("ggplot2")
  library("patchwork")
  data(SrE.IR)
  myt1 <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(IR ~ Spectra))
 myt2 <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(IR ~ Spectra ~ (Smoothed)))

  p1 <- plotSpectra(SrE.IR)
  p1 <- p1 + ggtitle(myt1) + coord_cartesian(xlim = c(1900, 2100), ylim = c(0.0, 0.03))

  sgf <- sgfSpectra(SrE.IR)

  p2 <- plotSpectra(sgf)
```

```
  p2 <- p2 + ggtitle(myt2) + coord_cartesian(xlim = c(1900, 2100), ylim = c(0.0, 0.03))

  p3 <- p1/p2
  p3
}
```

Spectra *Spectra Objects*

Description

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

Structure

The structure of a Spectra object is a list of 9 elements and an attribute as follows:

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

sumSpectra to summarize a Spectra object. sumGroups to summarize group membership of a Spectra object. chkSpectra to verify the integrity of a Spectra object. colorSymbol for a discussion of color options. Finally, additional documentation at https://bryanhanson.github.io/ChemoSpec/

splitSpectraGroups *Create New Groups from an Existing Spectra Object*

Description

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

Usage

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

Arguments

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

Details

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

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

Value

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

Author(s)

Bryan A. Hanson (DePauw University).

See Also

conColScheme Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

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

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

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

---

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

Description

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

Usage

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

Arguments

spectra	An object of S3 class Spectra().
pca	The result of a pca calculation on Spectra (i.e. the output from c_pcaSpectra or r_pcaSpectra).
pc	An integer specifying the desired pc plot.
tol	A number describing the fraction of points to be labeled. tol = 1.0 labels all the points; tol = 0.05 labels *approximately* the most extreme 5 percent. Set to 'none' to completely suppress labels. Note that a simple approach based upon quantiles is used, assumes that both x and y are each normally distributed, and treats x and y separately. Thus, this is not a formal treatment of outliers, just a means of labeling points. Groups are lumped together for the computation.
...	Parameters to be passed to the plotting routines. *Applies to base graphics only*.

Value

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

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

Author(s)

Bryan A. Hanson (DePauw University), Tejasvi Gupta, Matthew J. Keinsley.

References

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.IR)
pca <- c_pcaSpectra(SrE.IR)
myt <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(IR ~ Spectra))
p <- sPlotSpectra(spectra = SrE.IR, pca = pca, pc = 1, tol = 0.001)
p <- p + ggtitle(myt)
p
```

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

Description

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

Format

The data are stored as a Spectra object.

Source

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

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

---

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

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via sumGroups.

---

sumSpectra *Summarize a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via sumSpectra.

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

Description

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

Usage

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

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

Arguments

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

Details

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

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

Value

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

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

Functions

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

Author(s)

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

See Also

Additional documentation at https://bryanhanson.github.io/ChemoSpec/

Examples

```
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.IR)
myt <- expression(bolditalic(Serenoa) ~ bolditalic(repens) ~ bold(Extract ~ IR ~ Spectra))

p1 <- surveySpectra(SrE.IR, method = "iqr")
p1 <- p1 + ggtitle(myt)
p1

p2 <- surveySpectra2(SrE.IR, method = "iqr")
p2 <- p2 + ggtitle(myt)
p2
```

---

s_pcaSpectra *Sparse PCA of Spectra Objects*

Description

A wrapper which carries out sparse 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. The data will be centered, as is required by PCA.

Usage

```
s_pcaSpectra(spectra, choice = "noscale", K = 3, para = rep(0.5, K), ...)
```

Arguments

spectra	An object of S3 class Spectra().
choice	A character string indicating the choice of scaling. One of c("noscale", "autoscale", "Pareto"). "autoscale" is called "standard normal variate" or "correlation matrix PCA" in some literature.
K	Integer. The number of components desired.
para	A vector of length(K) giving the tuning parameters.
...	Other parameters to be passed to arrayspc.

Details

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

Value

An object of class prcomp and converted_from_arrayspc, which includes a list element called $method, a character string describing the pre-processing carried out and the type of PCA performed (used to annotate plots). A check is carried out to see if the computation was successful and a warning issued if it failed.

Author(s)

Bryan A. Hanson (DePauw University).

References

H. Zou, T. Hastie and R. Tibshirani "Sparse Principal Components Analysis" *J. Comp. Stat. Graphics* vol. 15 no. 2 pgs. 265-286 (2006).

See Also

arrayspc for the underlying function, c_pcaSpectra for classical PCA calculations, r_pcaSpectra for robust PCA calculations, irlba_pcaSpectra for PCA via the IRLBA algorithm. Additional documentation at https://bryanhanson.github.io/ChemoSpec/

For displaying the results, ChemoSpecUtils::plotScree(), ChemoSpecUtils::plotScores(), plotLoadings(), plot2Loadings(), sPlotSpectra().

Examples

```
## Not run:
# This example assumes the graphics output is set to ggplot2 (see ?GraphicsOptions).
library("ggplot2")
data(SrE.NMR)
pca <- s_pcaSpectra(SrE.NMR)

p1 <- plotScree(pca)
p1
```

```
p2 <- plotScores(SrE.NMR, pca, pcs = c(1, 2), ellipse = "cls", tol = 0.05)
p2 <- p2 + ggtitle("Scores: SrE NMR Data")
p2

p3 <- plotLoadings(SrE.NMR, pca, loads = 1:2, ref = 1)
p3 <- p3 + ggtitle("Loadings: SrE NMR Data")
p3

## End(Not run)
```

---

updateGroups *Update Group Names in a Spectra or Spectra2D Object*

Description

This function is used by ChemoSpec and ChemoSpec2D, but is formally part of ChemoSpecUtils. You can access full documentation via updateGroups.

Index