Package 'mixOmics'

September 1, 2014

Type Package

Title Omics Data Integration Project

Version 5.0-3

Date 2012-12-11

Depends R (>= 2.10), MASS, lattice

Imports RGCCA, igraph, rgl, pheatmap

Author Sebastien Dejean, Ignacio Gonzalez, Kim-Anh Le Cao with
contributions from Pierre Monget, Jeff Coquery, FangZou Yao,Benoit Liquet, Florian Rohart

Maintainer Kim-Anh Le Cao <k.lecao@uq.edu.au>

Description The package provide statistical integrative techniques and
variants to analyse highly dimensional data sets: regularized
CCA and sparse PLS to unravel relationships between two
heterogeneous data sets of size (nxp) and (nxq) where the p and
q variables are measured on the same samples or individuals n.
These data may come from high throughput technologies, such as
omics data (e.g. transcriptomics, metabolomics or proteomics data) that require an integrative or joint analysis. However,mixOmics can also be applied to any other large data sets where p + q >> n. rCCA is a regularized version of CCA to deal with
the large number of variables. sPLS allows variable selection
in a one step procedure and two frameworks are proposed:
regression and canonical analysis. Numerous graphical outputs
are provided to help interpreting the results. Recent
methodological developments include: sparse PLS-Discriminant
Analysis, Independent Principal Component Analysis and
multilevel analysis using variance decomposition of the data.

License GPL (>= 2)

Repository CRAN

Date/Publication 2014-09-01 15:47:59

NeedsCompilation no

R topics documented:

`breast.tumors` *Human Breast Tumors Data*

Description

This data set contains the expression of 1753 genes in 47 surgical specimens of human breast tumours from 17 different individuals before and after chemotherapy treatment.

Usage

```
data(breast.tumors)
```

Format

A list containing the following components:

`gene.exp` data matrix with 47 rows and 1753 columns. Each row represents an experimental sample, and each column a single gene.

`sample` a list containing two character vector components: `name` the name of the samples, and `treatment` the treatment status.

`genes` a list containing two character vector components: `name` the name of the genes, and `description` the description of each gene.

Details

This data consists of 47 breast cancer samples and 1753 cDNA clones pre-selected by Pérez-Enciso *et al.* (2003) to draw their Fig. 1. The authors selected 47 samples for which there was information at least before or before and after chemotherapy treatment. There were 20 tumours that were microarrayed both before and after treatment.

Source

The Human Breast Tumors dataset is a companion resource for the paper of Perou *et al.* (2000), and was downloaded from the Stanford Genomics Breast Cancer Consortium Portal http://genome-www.stanford.edu/breast_cancer/molecularportraits/download.shtml

References

Pérez-Enciso, M. and Tenenhaus, M. (2003). Prediction of clinical outcome with microarray data: a partial least squares discriminant analysis (PLS-DA) approach. *Human Genetics* **112**, 581-592.

Perou, C. M., Sorlie, T., Eisen, M. B., van de Rijn, M., Jeffrey, S. S., Rees, C. A., Pollack, J. R., Ross, D. T., Johnsen, H., Akslen, L. A., Fluge, O., Pergamenschikov, A., Williams, C., Zhu, S. X., Lonning, P. E., Borresen-Dale, A. L., Brown, P. O. and Botstein, D. (2000). Molecular portraits of human breast tumours. *Nature* **406**, 747-752.

`cim` *Clustered Image Maps (CIMs) ("heat maps")*

Description

This function generates color-coded Clustered Image Maps (CIMs) ("heat maps") to represent "high-dimensional" data sets.

Usage

```
## Default S3 method:
cim(mat, breaks, col = jet.colors,
    distfun = dist, hclustfun = hclust,
    dendrogram = c("both", "row", "column", "none"),
    labRow = NULL, labCol = NULL,
    ColSideColors = NULL, RowSideColors = NULL,
    symkey = TRUE, keysize = 1, zoom = FALSE,
    main = NULL, xlab = NULL, ylab = NULL,
    cexRow = min(1, 0.2 + 1/log10(nr)),
    cexCol = min(1, 0.2 + 1/log10(nc)),
    margins = c(5, 5), lhei = NULL, lwid = NULL, ...)

## S3 method for class 'rcc'
cim(object, comp = 1, X.names = NULL, Y.names = NULL, ...)

## S3 method for class 'spls'
cim(object, comp = 1, X.names = NULL, Y.names = NULL,
    keep.var = TRUE, ...)

## S3 method for class 'pls'
cim(object, comp = 1, X.names = NULL, Y.names = NULL, ...)
```

Arguments

`mat`	numeric matrix of values to be plotted.
`object`	object of class inheriting from `"rcc"`, `"pls"` or `"spls"`.

comp	atomic or vector of positive integers. The components to adequately account for the data association. Defaults to comp = 1.
X.names, Y.names	character vector containing the names of X- and Y-variables.
keep.var	boolean. If TRUE only the variables with loadings not zero are plotted (as selected by spls). Defaults to TRUE.
distfun	function used to compute the distance (dissimilarity) between both rows and columns. Defaults to dist.
breaks	(optional) either a numeric vector indicating the splitting points for binning mat into colors, or a integer number of break points to be used, in which case the break points will be spaced equally between min(mat) and max(mat).
col	a character string specifying the colors function to use: terrain.colors, topo.colors, rainbow or similar functions. Defaults to jet.colors.
hclustfun	function used to compute the hierarchical clustering for both rows and columns. Defaults to hclust. Should take as argument a result of distfun and return an object to which as.dendrogram can be applied.
dendrogram	character string indicating whether to draw "none", "row", "column" or "both" dendrograms. Defaults to "both".
labRow	character vectors with row labels to use. Defaults to rownames(mat).
labCol	character vectors with column labels to use. Defaults to colnames(mat).
ColSideColors	(optional) character vector of length ncol(mat) containing the color names for a horizontal side bar that may be used to annotate the columns of mat.
RowSideColors	(optional) character vector of length nrow(mat) containing the color names for a vertical side bar that may be used to annotate the rows of mat.
symkey	boolean indicating whether the color key should be made symmetric about 0. Defaults to TRUE.
keysize	positive numeric value indicating the size of the color key.
zoom	logical. Whether to use zoom for interactively zooming-out. See Details.
main, xlab, ylab	main, x- and y-axis titles; defaults to none.
cexRow, cexCol	positive numbers, used as cex.axis in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.
margins	numeric vector of length two containing the margins (see par(mar)) for column and row names respectively.
lhei, lwid	arguments passed to layout to divide the device up into two rows and two columns, with the row-heights lhei and the column-widths lwid.
...	arguments passed to cim.default.

Details

One matrix Clustered Image Map (default method) is a 2-dimensional visualization of a real-valued matrix (basically image(t(mat))) with a dendrogram added to the left side and to the top. The rows and columns are reordered according to some hierarchical clustering method to identify interesting

patterns. By default the used clustering method for rows and columns is the *complete linkage* method and the used distance measure is the distance *euclidean*.

In `rcc` method, the matrix `mat` is created where element (j, k) is the scalar product value between every pairs of vectors in dimension `length(comp)` representing the variables X_j and Y_k on the axis defined by Z_i with i in `comp`, where Z_i is the equiangular vector between the i-th X and Y canonical variate.

In `spls`, if `object$mode` is `regression`, the element (j, k) of the similarity matrix `mat` is given by the scalar product value between every pairs of vectors in dimension `length(comp)` representing the variables X_j and Y_k on the axis defined by U_i with i in `comp`, where U_i is the i-th X variate. If `object$mode` is `canonical` then X_j and Y_k are represented on the axis defined by U_i and V_i respectively.

For visualization of "high-dimensional" data sets, a nice zooming tool was created. `zoom=TRUE` open a new device, one for CIM, one for zoom-out region and define an interactive 'zoom' process: click two points at imagen map region by pressing the first mouse button. It then draws a rectangle around the selected region and zoom-out this at new device. The process can be repeated to zoom-out other regions of interest.

The zoom process is terminated by clicking the second button and selecting 'Stop' from the menu, or from the 'Stop' menu on the graphics window.

Value

A list containing the following components:

`simMat`	the similarity matrix used by `cim`.
`rowInd`	row index permutation vectors as returned by `order.dendrogram`.
`colInd`	column index permutation vectors as returned by `order.dendrogram`.
`ddr, ddc`	object of class `"dendrogram"` which describes the row and column trees produced by `cim`.
`labRow, labCol`	character vectors with row and column labels used.

Author(s)

Ignacio González.

References

Eisen, M. B., Spellman, P. T., Brown, P. O. and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceeding of the National Academy of Sciences of the USA* **95**, 14863-14868.

Weinstein, J. N., Myers, T. G., O'Connor, P. M., Friend, S. H., Fornace Jr., A. J., Kohn, K. W., Fojo, T., Bates, S. E., Rubinstein, L. V., Anderson, N. L., Buolamwini, J. K., van Osdol, W. W., Monks, A. P., Scudiero, D. A., Sausville, E. A., Zaharevitz, D. W., Bunow, B., Viswanadhan, V. N., Johnson, G. S., Wittes, R. E. and Paull, K. D. (1997). An information-intensive approach to the molecular pharmacology of cancer. *Science* **275**, 343-349.

See Also

image, heatmap, hclust, plotVar, plot3dVar, network and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## default method
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene

cim(cor(X, Y), dendrogram = "none")

## CIM representation for objects of class 'rcc'
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)

dends <- cim(nutri.res, comp = 1:3, xlab = "genes",
             ylab = "lipids", margins = c(5, 6))

op <- par(mar = c(5, 4, 4, 4), cex = 0.8)
plot(dends$ddr, axes = FALSE, horiz = TRUE)
par(op)

## interactive 'zoom'
## Not run:
cim(nutri.res, comp = 1:3, zoom = TRUE)
## select the region and "see" the zoom-out region

## End(Not run)

## CIM representation for objects of class 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic

toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))

cim(toxicity.spls, comp = 1:3)
```

---

data.simu *Simulation study for multilevel analysis*

Description

Simulation study to illustrate the use of the multilevel analysis for one and two-factor analysis with sPLS-DA. This data set contains the expression measure of 1000 genes.

Usage

```
data(data.simu)
```

Format

A list containing the following components:

`X` data frame with 48 samples and 1000 genes.

`stimu` a factor indicating the conditions on each sample (stimulations)

`sample` a vector indicating the repeated measurements on each unique subject. See details.

Details

In this cross-over design, repeated measurements are performed 12 experiments units (or unique subjects) for each of the 4 stimulations.

The simulation study was based on a mixed effects model (see reference for details). Ten clsuters of 100 genes were generated. Amongt those, 4 clusters of genes discriminate the 4 stimulations (denoted LIPO5, GAG+, GAG- and NS) as follows: \ -2 gene clusters discriminate (LIPO5, GAG+) versus (GAG-, NS) \ -2 gene clusters discriminate LIPO5 versus GAG+, while GAG+ and NS have the same effect \ - gene clusters discriminate GAG- versus NS, while LIPO5 and GAG+ have the same effect \ -the 4 remaining clusters represent noisy signal (no stimulation effect)

References

Liquet, B., Lê Cao, K.-A., Hocini, H. and Thiebaut, R. A novel approach for biomarker selection and the integration of repeated measures experiments from two platforms. *Submitted*.

---

estim.regul *Estimate the parameters of regularization for Regularized CCA*

Description

This function has been renamed `tune.rcc`, see `tune.rcc`.

---

image *Plot the cross-validation score.*

Description

This function provide a image map (checkerboard plot) of the cross-validation score obtained by the `tune.rcc` function.

Usage

```
## S3 method for class 'tune.rcc'
image(x, col = heat.colors, ...)
```

Arguments

x	object returned by estim.regul.
col	a character string specifying the colors function to use: terrain.colors, topo.colors, rainbow or similar functions. Defaults to heat.colors.
...	not used currently.

Details

image.estim.regul creates an image map of the matrix object$mat containing the cross-validation score obtained by the estim.regul function. Also a color scales strip is plotted.

Author(s)

Sébastien Déjean and Ignacio González.

See Also

tune.rcc, image.

Examples

```
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene

## this can take some seconds
cv.score <- tune.rcc(X, Y, validation = "Mfold", plt = FALSE)
image(cv.score)
```

---

image.estim.regul *Plot the cross-validation score.*

Description

This function has been renamed 'image.tune.rcc', see image.tune.rcc.

imgCor *Image Maps of Correlation Matrices between two Data Sets*

Description

Display two-dimensional visualizations (image maps) of the correlation matrices within and between two data sets.

Usage

```
imgCor(X, Y, type = c("combine", "separate"), col = jet.colors,
       X.names = TRUE, Y.names = TRUE,
       XsideColor = "blue", YsideColor = "red",
       symkey = TRUE, keysize = 1, interactive.dev = TRUE,
       cexRow = NULL, cexCol = NULL,
       margins = c(5, 5), lhei = NULL, lwid = NULL)
```

Arguments

X	numeric matrix or data frame (n x p), the observations on the X variables. NAs are allowed.
Y	numeric matrix or data frame (n x q), the observations on the Y variables. NAs are allowed.
type	character string, (partially) maching one of "combine" or "separated", determining the kind of plots to be produced. See Details.
col	vector of colors such as that generated by heat.colors, topo.colors, rainbow or similar functions. Defaults to jet.colors(26).
X.names	logical, should the name of each data on the axis X be shown ? Possible character vector giving the names of the X-variables.
Y.names	logical, should the name of each data on the axis Y be shown ? Possible character vector giving the names of the Y-variables.
XsideColor	character string. The color name for a horizontal side bar that may be used to annotate the columns of X.
YsideColor	character string. The color name for a vertical side bar that may be used to annotate the rows of Y.
symkey	boolean indicating whether the color key should be made symmetric about 0. Defaults to TRUE.
keysize	positive numeric value indicating the size of the color key.
interactive.dev	boolean. The current graphics device that will be opened is interactive?
cexRow, cexCol	positive numbers, used as cex.axis in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.
margins	numeric vector of length two containing the margins (see par(mar)) for column and row names respectively.

`lhei, lwid` arguments passed to `layout` to divide the device up into two rows and two columns, with the row-heights `lhei` and the column-widths `lwid`.

Details

If `type="combine"`, the correlation matrix is computed of the combined matrices `cbind(X, Y)` and then plotted. If `type="separate"`, three correlation matrices are computed, `cor(X)`, `cor(Y)` and `cor(X,Y)` and plotted separately on a device. In both cases, a color correlation scales strip is plotted.

The correlation matrices are pre-processed before calling the `image` function in order to get, as in the numerical representation, the diagonal from upper-left corner to bottom-right one.

Missing values are handled by casewise deletion in the `imgCor` function.

If `X.names = FALSE`, the name of each X-variable is hidden. Default value is `TRUE`.

If `Y.names = FALSE`, the name of each Y-variable is hidden. Default value is `TRUE`.

Author(s)

Sébastien Déjean, Ignacio González and Pierre Monget.

See Also

`cor`, `image`, `jet.colors`.

Examples

```
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene

## 'combine' type plot (default)
imgCor(X, Y)

## 'separate' type plot
## Not run:
imgCor(X, Y, type = "separate")

## 'separate' type plot without the name of datas
imgCor(X, Y, X.names = FALSE, Y.names = FALSE, type = "separate")

## End(Not run)
```

---

internal-functions *Internal Functions*

Description

Internal functions not to be used by the user.

ipca *Independent Principal Component Analysis*

Description

Performs independent principal component analysis on the given data matrix, a combination of Principal Component Analysis and Independent Component Analysis.

Usage

```
ipca(X, ncomp = 3, mode = c("deflation","parallel"),
          fun = c("logcosh", "exp"),
          scale = FALSE, max.iter = 200,
          tol = 1e-04, w.init=NULL)
```

Arguments

X	a numeric matrix (or data frame) which provides the data for the principal component analysis.
ncomp	integer, number of independent component to choose. Set by default to 3.
mode	character string. What type of algorithm to use when estimating the unmixing matrix, choose one of "deflation", "parallel". Default set to deflation.
fun	the function used in approximation to neg-entropy in the FastICA algorithm. Default set to logcosh, see details of FastICA.
scale	a logical value indicating whether the variables (columns) of the data matrix X should be standardized beforehand. By default, X is centered.
max.iter	integer, maximum number of iterations to perform.
tol	a positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged, see fastICA package.
w.init	initial un-mixing matrix (unlike FastICA, this matrix is fixed here).

Details

In PCA, the loading vectors indicate the importance of the variables in the principal components. In large biological data sets, the loading vectors should only assign large weights to important variables (genes, metabolites ...). That means the distribution of any loading vector should be super-Gaussian: most of the weights are very close to zero while only a few have large (absolute) values.

However, due to the existence of noise, the distribution of any loading vector is distorted and tends toward a Gaussian distribtion according to the Central Limit Theroem. By maximizing the non-Gaussianity of the loading vectors using FastICA, we obtain more noiseless loading vectors. We then project the original data matrix on these noiseless loading vectors, to obtain independent principal components, which should be also more noiseless and be able to better cluster the samples according to the biological treatment (note, IPCA is an unsupervised approach).

Algorithm 1. The original data matrix is centered.

2. PCA is used to reduce dimension and generate the loading vectors.

3. ICA (FastICA) is implemented on the loading vectors to generate independent loading vectors.

4. The centered data matrix is projected on the independent loading vectors to obtain the independent principal components.

Value

`ipca` returns a list with class `"ipca"` containing the following components:

`ncomp`	the number of independent principal components used.
`unmixing`	the unmixing matrix of size (ncomp x ncomp)
`mixing`	the mixing matrix of size (ncomp x ncomp)
`X`	the centered data matrix
`x`	the indepenent principal components
`loadings`	the independent loading vectors
`kurtosis`	the kurtosis measure of the independent loading vectors

Author(s)

Fangzhou Yao and Jeff Coquery.

References

Yao, F., Coquery, J. and L\^e Cao, K.-A.(2011) Principal component analysis with independent loadings: a combination of PCA and ICA. (in preparation)

A. Hyvarinen and E. Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*, **13(4-5)**:411-430

J L Marchini, C Heaton and B D Ripley (2010). fastICA: FastICA Algorithms to perform ICA and Projection Pursuit. R package version 1.1-13.

See Also

`sipca`, `pca`, `plotIndiv`, `plotVar`, `plot3dIndiv`, `plot3dVar` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details..

Examples

```
data(liver.toxicity)

# implement IPCA on a microarray dataset
ipca.res <- ipca(liver.toxicity$gene, ncomp = 3, mode="deflation")
ipca.res

# samples representation
plotIndiv(ipca.res, ind.names = liver.toxicity$treatment[, 4], cex = 0.5,
          col = as.numeric(as.factor(liver.toxicity$treatment[, 4])))
## Not run:
plot3dIndiv(ipca.res, cex = 0.01,
```

```
            col = as.numeric(as.factor(liver.toxicity$treatment[, 4])))

## End(Not run)

# variables representation
plotVar(ipca.res, var.label = TRUE, cex = 0.5)

## Not run:
plot3dVar(ipca.res, rad.in = 0.5, cex = 0.5,
          col = as.numeric(as.factor(liver.toxicity$treatment[, 4])))

## End(Not run)
```

`jet.colors` *Jet Colors Palette*

Description

Create a vector of `n` "contiguous" colors.

Usage

```
jet.colors(n)
```

Arguments

`n`	an integer, the number of colors (≥ 1) to be in the palette.

Details

The function `jet.colors(n)` create color scheme, beginning with dark blue, ranging through shades of blue, cyan, green, yellow and red, and ending with dark red. This colors palette is suitable for displaying ordered (symmetric) data, with `n` giving the number of colors desired.

Value

A character vector, `cv`, of color names. This can be used either to create a user-defined color palette for subsequent graphics by `palette(cv)`, a `col=` specification in graphics functions or in `par`.

See Also

`colorRamp`, `palette`, `colors` for the vector of built-in "named" colors; `hsv`, `gray`, `rainbow`, `terrain.colors`, ... to construct colors; and `heat.colors`, `topo.colors` for images.

Examples

```
## Jet Color Scales Strips
def.par <- par(no.readonly = TRUE)
par(mfrow = c(3, 1))
z <- seq(-1, 1, length = 125)
for (n in c(11, 33, 125)) {
    image(matrix(z, ncol = 1), col = jet.colors(n),
          xaxt = "n", yaxt = "n", main = paste("n = ", n))
    box()
    par(usr = c(-1, 1, -1, 1))
    axis(1, at = c(-1, 0, 1))
}
par(def.par)
```

`linnerud` *Linnerud Dataset*

Description

Three physiological and three exercise variables are measured on twenty middle-aged men in a fitness club.

Usage

```
data(linnerud)
```

Format

A list containing the following components:

`exercise` data frame with 20 observations on 3 exercise variables.

`physiological` data frame with 20 observations on 3 physiological variables.

Source

Tenenhaus, M. (1998), Table 1, page 15.

References

Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*. Paris: Editions Technic.

`liver.toxicity` *Liver Toxicity Data*

Description

This data set contains the expression measure of 3116 genes and 10 clinical measurements for 64 subjects (rats) that were exposed to non-toxic, moderately toxic or severely toxic doses of acetaminophen in a controlled experiment.

Usage

```
data(liver.toxicity)
```

Format

A list containing the following components:

`gene` data frame with 64 rows and 3116 columns. The expression measure of 3116 genes for the 64 subjects (rats).

`clinic` data frame with 64 rows and 10 columns, containing 10 clinical variables for the same 64 subjects.

`treatment` data frame with 64 rows and 4 columns, containing the treatment information on the 64 subjects, such as doses of acetaminophen and times of necropsies.

`gene.ID` data frame with 3116 rows and 2 columns, containing geneBank IDs and gene titles of the annotated genes

Details

The data come from a liver toxicity study (Bushel *et al*., 2007) in which 64 male rats of the inbred strain Fisher 344 were exposed to non-toxic (50 or 150 mg/kg), moderately toxic (1500 mg/kg) or severely toxic (2000 mg/kg) doses of acetaminophen (paracetamol) in a controlled experiment. Necropsies were performed at 6, 18, 24 and 48 hours after exposure and the mRNA from the liver was extracted. Ten clinical chemistry measurements of variables containing markers for liver injury are available for each subject and the serum enzymes levels are measured numerically. The data were further normalized and pre-processed by Bushel *et al*. (2007).

Source

The two liver toxicity data sets are a companion resource for the paper of Bushel *et al.* (2007), and was downloaded from:

http://www.biomedcentral.com/1752-0509/1/15/additional/

References

Bushel, P., Wolfinger, R. D. and Gibson, G. (2007). Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes. *BMC Systems Biology* **1**, Number 15.

Lê Cao, K.-A., Rossouw, D., Robert-Granié, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

`mat.rank` *Matrix Rank*

Description

This function estimate the rank of a matrix.

Usage

```
mat.rank(mat, tol)
```

Arguments

`mat`	a numeric matrix or data frame that can contain missing values.
`tol`	positive real, the tolerance for singular values, only those with values larger than `tol` are considered non-zero.

Details

`mat.rank` estimate the rank of a matrix by computing its singular values $d[i]$ (using `nipals`). The rank of the matrix can be defined as the number of singular values $d[i] > 0$.

If `tol` is missing, it is given by `tol=max(dim(mat))*max(d)*.Machine$double.eps`.

Value

The returned value is a list with components:

`rank`	a integer value, the matrix rank.
`tol`	the tolerance used for singular values.

Author(s)

Sébastien Déjean and Ignacio González.

See Also

`nipals`

Examples

```
## Hilbert matrix
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
mat <- hilbert(16)
mat.rank(mat)

## Hilbert matrix with missing data
idx.na <- matrix(sample(c(0, 1, 1, 1, 1), 36, replace = TRUE), ncol = 6)
m.na <- m <- hilbert(9)[, 1:6]
m.na[idx.na == 0] <- NA
mat.rank(m)
mat.rank(m.na)
```

`multidrug` *Multidrug Resistence Data*

Description

This data set contains the expression of 48 known human ABC transporters with patterns of drug activity in 60 diverse cancer cell lines (the NCI-60) used by the National Cancer Institute to screen for anticancer activity.

Usage

```
data(multidrug)
```

Format

A list containing the following components:

`ABC.trans` data matrix with 60 rows and 48 columns. The expression of the 48 human ABC transporters.

`compound` data matrix with 60 rows and 1429 columns. The activity of 1429 drugs for the 60 cell lines.

`comp.name` character vector. The names or the NSC No. of the 1429 compounds.

`cell.line` a list containing two character vector components: `Sample` the names of the 60 cell line which were analysed, and `Class` the phenotypes of the 60 cell lines.

Details

The data come from a pharmacogenomic study (Szakács *et al.*, 2004) in which two kinds of measurements acquired on the NCI-60 cancer cell lines are considered:

- the expression of the 48 human ABC transporters measured by real-time quantitative RT-PCR for each cell line;
- the activity of 1429 drugs expressed as GI_{50} which corresponds to the concentration at which the drug induces 50% inhibition of cellular growth for the cell line tested.

The NCI-60 panel includes cell lines derived from cancers of colorectal (7 cell lines), renal(8), ovarian(6), breast(8), prostate(2), lung(9) and central nervous system origin(6), as well as leukemias(6) and melanomas(8). It was set up by the Developmental Therapeutics Program of the National Cancer Institute (NCI, one of the U.S. National Institutes of Health) to screen the toxicity of chemical compound repositories. The expressions of the 48 human ABC transporters is available as a supplement to the paper of Szakács *et al.* (2004).

The drug dataset consiste of 118 compounds whose mechanisms of action are putatively classifiable (Weinstein *et al.*, 1992) and a larger set of 1400 compounds that have been tested multiple times and whose screening data met quality control criteria described elsewhere (Scherf *et al.*, 2000). The two were combined to form a joint dataset that included 1429 compounds.

Source

The NCI dataset was downloaded from The Genomics and Bioinformatics Group Supplemental Table S1 to the paper of Szakács *et al.* (2004), http://discover.nci.nih.gov/abc/2004_cancercell_abstract.jsp#supplement

The two drug data sets are a companion resource for the paper of Scherf *et al.* (2000), and was downloaded from http://discover.nci.nih.gov/datasetsNature2000.jsp.

References

Scherf, U., Ross, D. T., Waltham, M., Smith, L. H., Lee, J. K., Tanabe, L., Kohn, K. W., Reinhold, W. C., Myers, T. G., Andrews, D. T., Scudiero, D. A., Eisen, M. B., Sausville, E. A., Pommier, Y., Botstein, D., Brown, P. O. and Weinstein, J. N. (2000). A Gene Expression Database for the Molecular Pharmacology of Cancer. *Nature Genetics*, **24**, 236-244.

Szakács, G., Annereau, J.-P., Lababidi, S., Shankavaram, U., Arciello, A., Bussey, K. J., Reinhold, W., Guo, Y., Kruh, G. D., Reimers, M., Weinstein, J. N. and Gottesman, M. M. (2004). Predicting drug sensivity and resistance: Profiling ABC transporter genes in cancer cells. *Cancer Cell* **4**, 147-166.

Weinstein, J.N., Kohn, K.W., Grever, M.R., Viswanadhan, V.N., Rubinstein, L.V., Monks, A.P., Scudiero, D.A., Welch, L., Koutsoukos, A.D., Chiausa, A.J. et al. 1992. Neural computing in cancer drug development: Predicting mechanism of action. *Science* **258**, 447-451.

`multilevel` *Multilevel analysis for repeated measurements (cross-over design)*

Description

The analysis of repeated measurements is performed by combining a multilevel approach with multivariate methods: sPLS-DA (Discriminant Analysis) or sPLS (Integrative analysis). Both approaches embbed variable selection.

Usage

```
multilevel(X, Y = NULL,
           cond = NULL,
           sample = NULL,
           ncomp = 1,
           keepX = rep(ncol(X), ncomp),
           keepY = NULL,
           method = NULL,
           tab.prob.gene=NULL,
           max.iter = 500,
           tol = 1e-06,...)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	if(method = 'spls') numeric vector or matrix of continuous responses (for multi-response models) NAs are allowed.
cond	a factor or a class vector for one-factor discrete outcome, a numeric matrix of 2 columns for two-factor discrete outcome. See Details
sample	a vector indicating the repeated measured on each individual.
ncomp	the number of components to include in the model (see Details).
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
keepY	if(method = 'spls'), numeric vector of length ncomp, the number of variables to keep in Y-loadings. By default all variables are kept in the model.
method	character string. Which multivariate method and type of analysis to choose, matching one of 'splsda' (Discriminant Analysis) or 'spls' (unsupervised integrative analysis). See Details.
tab.prob.gene	matrix of dimension ncol(X)x2 indicating the probes names (column 1) and corresponding gene names (column 2).
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
...	other internal arguments.

Details

multilevel function first decomposes the variance in the data sets X (and Y) and applies either sPLS-DA (method = 'splsda') or sPLS ((method = 'spls')) on the within-subject deviation.

One or two-factor analyses are available for (method = 'splsda').

A sPLS-DA or sPLS model is performed with $1, \ldots,$ncomp components to the factor or class vector cond. The appropriate indicator matrix is created for sPLS-DA.

Multilevel sPLS-DA enables the selection of discriminant variables between the conditions in cond.

Multilevel sPLS enables the integration of data measured on two different platforms on the same individuals. This approach differs from multilevel sPLS-DA as the aim is to select subsets variables

from both data sets that are highly correlated (positively or negatively) across the samples. The approach is unsupervised: no prior knowledge about the samples groups is included.

Value

For sPLS-DA analysis, `multilevel` returns either an object of class `"splsda1fact"` for one-factor analysis and `"splsda2fact"`, a list that contains the following sPLS-DA outputs:

`X`	the centered and standardized original predictor matrix.
`Y`	the centered and standardized indicator response vector or matrix.
`ind.mat`	the indicator matrix.
`ncomp`	the number of components included in the model.
`keepX`	number of X variables kept in the model on each component.
`mat.c`	matrix of coefficients to be used internally by `predict`.
`variates`	list containing the variates.
`loadings`	list containing the estimated loadings for the `X` and `Y` variates.
`names`	list containing the names to be used for individuals and variables.
`nzv`	list containing the zero- or near-zero predictors information.

For sPLS analysis, `multilevel` returns either an object of class `"splslevel"` for one-factor analysis, a list that contains the following sPLS-DA components:

`X`	the centered and standardized original predictor matrix.
`Y`	the centered and standardized original response vector or matrix.
`ncomp`	the number of components included in the model.
`mode`	the algorithm used to fit the model.
`keepX`	number of X variables kept in the model on each component.
`keepY`	number of Y variables kept in the model on each component.
`mat.c`	matrix of coefficients to be used internally by `predict`.
`variates`	list containing the variates.
`loadings`	list containing the estimated loadings for the X and Y variates.
`names`	list containing the names to be used for individuals and variables.
`nzv`	list containing the zero- or near-zero predictors information.

Author(s)

Benoit Liquet, Kim-Anh L\^e Cao.

References

On multilevel analysis: Liquet, B., LÃª Cao, K.-A., Hocini, H. and Thiebaut, R. (2012) A novel approach for biomarker selection and the integration of repeated measures experiments from two platforms. *BMC Bioinformatics* **13**:325.

Westerhuis, J. A., van Velzen, E. J., Hoefsloot, H. C., and Smilde, A. K. (2010). Multivariate paired data analysis: multilevel PLSDA versus OPLSDA. *Metabolomics*, **6**(1), 119-128.

On sPLS-DA: LÃª Cao, K.-A., Boitard, S. and Besse, P. (2011). Sparse PLS Discriminant Analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* **12**:253.

On sPLS: LÃª Cao, K.-A., Martin, P.G.P., Robert-Granie, C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

LÃª Cao, K.-A., Rossouw, D., Robert-Granie, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

See Also

spls, splsda, plotIndiv, plotVar, plot3dIndiv, plot3dVar, cim, network.

Examples

```
## First example: one-factor analysis with sPLS-DA
data(vac18) # vac18 study
X <- vac18$genes
Y <- vac18$stimulation

res <- multilevel(X, cond = Y, ncomp = 3,
  tab.prob.gene = vac18$tab.prob.gene,
  sample = vac18$sample, method = "splsda",
  keepX = c(30, 137, 123))

## Not run:
# color for plot3dIndiv
col_stim <- c("darkblue", "purple", "green4","red3")
cols <- Y
levels(cols) <- col_stim
cols <- as.character(cols)

plot3dIndiv(res, ind.names = Y, col = cols,
cex = 0.3, axes.box = "both")

## End(Not run)

## Second example: two-factor analysis with sPLS-DA
data(data.simu) # simulated data

time = factor(rep(c(rep('t1', 6), rep('t2', 6)), 4))
stimu.time = data.frame(cbind(as.character(data.simu$stimu),
```

```
                        as.character(time)))
repeat.simu2 = rep(c(1:6), 8)

res.2level <- multilevel(data.simu$X, cond = stimu.time,
                         sample = repeat.simu2, ncomp = 2,
                         keepX = c(200, 200), tab.prob.gene = NULL,
                         method = 'splsda')

# color for plotIndiv
col.stimu = as.numeric(data.simu$stimu)
# pch for plots
pch.time = rep(20, 48)
pch.time[time == 't2'] = 4

plotIndiv(res.2level, col = col.stimu, pch = pch.time, ind.names = FALSE)
legend('bottomright', col = unique(col.stimu),
       legend = levels(data.simu$stimu),  pch = 20, cex = 0.8)
legend('topright', col = 'black', legend = levels(time),
       pch = unique(pch.time), cex = 0.8)


## Third example: one-factor integrative analysis with sPLS
## Not run:
data(liver.toxicity)
repeat.indiv = c(1, 2, 1, 2, 1, 2, 1, 2, 3, 3, 4, 3, 4, 3, 4, 4, 5, 6, 5, 5,
                 6, 5, 6, 7, 7, 8, 6, 7, 8, 7, 8, 8, 9, 10, 9, 10, 11, 9, 9,
                 10, 11, 12, 12, 10, 11, 12, 11, 12, 13, 14, 13, 14, 13, 14,
                 13, 14, 15, 16, 15, 16, 15, 16, 15, 16)

result.rat <- multilevel(X = liver.toxicity$gene, Y=liver.toxicity$clinic,
                         cond = liver.toxicity$treatment$Dose.Group,
                         sample = repeat.indiv, ncomp = 2, keepX = c(50, 50),
                         keepY = c(5, 5), method = 'spls')

# variable plots
plotVar(result.rat, comp = 1:2, X.label = TRUE, Y.label = TRUE,
        cex = c(0.5, 0.9))

CIM <- cim(result.rat, comp = 1:2, xlab = "genes", ylab = "clinic var",
           margins = c(5, 6), zoom = FALSE)


network(result.rat, comp = 1:2, threshold = 0.8,
        Y.names = NULL, keep.var = TRUE,
        color.node = c( "lightcyan","mistyrose"),
        shape.node = c("circle", "rectangle"),
        color.edge = c("red", "green"),
        lty.edge = c("solid", "solid"), lwd.edge = c(1, 1),
        show.edge.labels = FALSE, interactive = FALSE)

## End(Not run)
```

`nearZeroVar` *Identification of zero- or near-zero variance predictors*

Description

Borrowed from the **caret** package and, used as internal function. This function diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large.

Usage

```
nearZeroVar(x, freqCut = 95/5, uniqueCut = 10)
```

Arguments

`x`	a numeric vector or matrix, or a data frame with all numeric data.
`freqCut`	the cutoff for the ratio of the most common value to the second most common value.
`uniqueCut`	the cutoff for the percentage of distinct values out of the number of total samples.

Details

For example, an example of near zero variance predictor is one that, for 1000 samples, has two distinct values and 999 of them are a single value.

To be flagged, first the frequency of the most prevalent value over the second most frequent value (called the "frequency ratio") must be above `freqCut`. Secondly, the "percent of unique values," the number of unique values divided by the total number of samples (times 100), must also be below `uniqueCut`.

In the above example, the frequency ratio is 999 and the unique value percentage is 0.0001.

Value

`nearZeroVar` returns a list that contains the following components:

`Position`	a vector of integers corresponding to the column positions of the problematic predictors.
`Metrics`	a data frame containing the zero- or near-zero predictors information with columns: `freqRatio`, the ratio of frequencies for the most common value over the second most common value and, `percentUnique`, the percentage of unique data points out of the total number of data points.

Author(s)

Max Kuhn, with speed improvements to nearZerVar by Allan Engelhardt

network *Relevance Network for (r)CCA and (s)PLS regression*

Description

Display relevance associations network for (regularized) canonical correlation analysis and (sparse) PLS regression.

Usage

```
## Default S3 method:
network(mat, threshold = 0.5, X.names = NULL, Y.names = NULL,
        color.node = c("white", "white"),
        shape.node = c("circle", "rectangle"),
        color.edge = c("blue", "red"),
        lty.edge = c("solid", "solid"),
        lwd.edge = c(1, 1), show.edge.labels = FALSE,
        show.color.key = TRUE, symkey = TRUE, keysize = 1,
        breaks, interactive = FALSE, alpha = 1, ...)

## S3 method for class 'rcc'
network(object, comp = 1, X.names = NULL, Y.names = NULL, ...)

## S3 method for class 'pls'
network(object, comp = 1, X.names = NULL, Y.names = NULL,
        keep.var = TRUE, ...)

## S3 method for class 'spls'
network(object, comp = 1, X.names = NULL, Y.names = NULL,
        keep.var = TRUE, ...)
```

Arguments

mat	numeric matrix of values to be represented.
object	object of class inheriting from "rcc", "pls" or "spls".
comp	atomic or vector of positive integers. The components to adequately account for the data association. Defaults to comp = 1.
threshold	numeric value between 0 and 1. The tuning threshold for the relevant associations network (see Details).
X.names, Y.names	character vector containing the names of X- and Y-variables.
keep.var	boolean. If TRUE only the variables with loadings not zero are plotted (as selected by spls). Defaults to TRUE.
color.node	vector of length two, the colors of the X and Y nodes (see Details).
shape.node	character vector of length two, the shape of the X and Y nodes (see Details).

color.edge	vector of colors or character string specifying the colors function to using to color the edges (see Details).
lty.edge	character vector of length two, the line type for the edges (see Details).
lwd.edge	vector of length two, the line width of the edges (see Details).
show.edge.labels	logical. If TRUE, plot association values as edge labels (defaults to FALSE).
show.color.key	boolean. If TRUE a color key should be plotted.
symkey	boolean indicating whether the color key should be made symmetric about 0. Defaults to TRUE.
keysize	numeric value indicating the size of the color key.
breaks	(optional) either a numeric vector indicating the splitting points for binning mat into colors, or a integer number of break points to be used, in which case the break points will be spaced equally between min(mat) and max(mat).
interactive	logical. If TRUE, a scrollbar is created to change the threshold value interactively (defaults to FALSE). See Details.
alpha	numeric value. Tuning parameter to enhance the differences between edges corresponding to low and high correlations. Defaults to alpha = 1.
...	arguments passed to network.default.

Details

network allows to infer large-scale association networks between the X and Y datasets in rcc or spls. The output is a graph where each X- and Y-variable corresponds to a node and the edges included in the graph portray associations between them.

In rcc, to identify X-Y pairs showing relevant associations, network calculate a similarity measure between X and Y variables in a pair-wise manner: the scalar product value between every pairs of vectors in dimension length(comp) representing the variables X and Y on the axis defined by Z_i with i in comp, where Z_i is the equiangular vector between the i-th X and Y canonical variate.

In spls, if object$mode is regression, the similarity measure between X and Y variables is given by the scalar product value between every pairs of vectors in dimension length(comp) representing the variables X and Y on the axis defined by U_i with i in comp, where U_i is the i-th X variate. If object$mode is canonical then X and Y are represented on the axis defined by U_i and V_i respectively.

Variable pairs with a high similarity measure (in absolute value) are considered as relevant. By changing the threshold, one can tune the relevance of the associations to include or exclude relationships in the network.

interactive=TRUE open two device, one for association network, one for scrollbar, and define an interactive process: by clicking either at each end ('$-$' or '$+$') of the scrollbar or at middle portion of this. The position of the slider indicate which is the 'threshold' value associated to the display network.

The interactive process is terminated by clicking the second button and selecting 'Stop' from the menu, or from the 'Stop' menu on the graphics window.

The color.node is a vector of length two, of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb", or an integer i

meaning `palette()[i]`. `color.node[1]` and `color.node[2]` give the color for filled nodes of the X- and Y-variables respectively. Defaults to `c("white", "white")`.

`color.edge` give the color to edges with colors corresponding to the values in `mat`. Defaults to `c("blue", "red")` for low and high weight respectively.

`shape.node[1]` and `shape.node[2]` provide the shape of the nodes associate to X- and Y-variables respectively. Current acceptable values are `"circle"` and `"rectangle"`. Defaults to `c("circle", "rectangle")`.

`lty.edge[1]` and `lty.egde[2]` give the line type to edges with positive and negative weight respectively. Can be one of `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, `"longdash"` and `"twodash"`. Defaults to `c("solid", "solid")`.

`lwd.edge[1]` and `lwd.edge[2]` provide the line width to edges with positive and negative weight respectively. This attribute is of type double with a default of `c(1, 1)`.

Value

`network` return a list containing the following components:

`simMat`	the similarity (adjacent) matrix used by `network`.
`gR`	a `graph` object (see the **igraph** package).

Warning

If the number of variables is high, the generation of the network can take some seconds.

Author(s)

S?bastien D?jean, Ignacio Gonz?lez and Kim-Anh L? Cao.

References

Butte, A. J., Tamayo, P., Slonim, D., Golub, T. R. and Kohane, I. S. (2000). Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proceedings of the National Academy of Sciences of the USA* **97**, 12182-12186.

Moriyama, M., Hoshida, Y., Otsuka, M., Nishimura, S., Kato, N., Goto, T., Taniguchi, H., Shiratori, Y., Seki, N. and Omata, M. (2003). Relevance Network between Chemosensitivity and Transcriptome in Human Hepatoma Cells. *Molecular Cancer Therapeutics* **2**, 199-205.

See Also

`plotVar`, `plot3dVar`, `cim` and http: //www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## network representation for objects of class 'rcc'
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)
```

```
## Not run:    # may not work on the Linux version, use Windows instead
network(nutri.res, comp = 1:3, threshold = 0.6)

## End(Not run)

## Changing the attributes of the network
## Not run:
network(nutri.res, comp = 1:3, threshold = 0.45,
        color.node = c("mistyrose", "lightcyan"),,
        shape.node = c("circle", "rectangle"),
        color.edge = jet.colors(8),
        lty.edge = c("solid", "solid"), lwd.edge = c(2, 2),
        show.edge.labels = FALSE)

## End(Not run)

## interactive 'threshold'
## Not run:
network(nutri.res, comp = 1:3, threshold = 0.55, interactive = TRUE)
## select the 'threshold' and "see" the new network

## End(Not run)

## network representation for objects of class 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))
## Not run:
network(toxicity.spls, comp = 1:3, threshold = 0.8,
        X.names = NULL, Y.names = NULL, keep.var = TRUE,
        color.node = c("mistyrose", "lightcyan"),
        shape.node = c("rectangle", "circle"),
        color.edge = c("red", "blue"),
        lty.edge = c("solid", "solid"), lwd.edge = c(1, 1),
        show.edge.labels = FALSE, interactive = FALSE)

## End(Not run)
```

`nipals` *Non-linear Iterative Partial Least Squares (NIPALS) algorithm*

Description

This function performs NIPALS algorithm, i.e. the singular-value decomposition (SVD) of a data table that can contain missing values.

Usage

```
nipals(X, ncomp = 1, reconst = FALSE, max.iter = 500, tol = 1e-09)
```

Arguments

X	real matrix or data frame whose SVD decomposition is to be computed. It can contain missing values.
ncomp	integer, the number of components to keep. If missing ncomp=ncol(X).
reconst	logical that specify if nipals must perform the reconstitution of the data using the ncomp components.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.

Details

The NIPALS algorithm (Non-linear Iterative Partial Least Squares) has been developed by H. Wold at first for PCA and later-on for PLS. It is the most commonly used method for calculating the principal components of a data set. It gives more numerically accurate results when compared with the SVD of the covariance matrix, but is slower to calculate.

This algorithm allows to realize SVD with missing data, without having to delete the rows with missing data or to estimate the missing data.

Value

The returned value is a list with components:

eig	vector containing the pseudosingular values of X, of length ncomp.
t	matrix whose columns contain the left singular vectors of X.
p	matrix whose columns contain the right singular vectors of X.
rec	matrix obtained by the reconstitution of the data using the ncomp components.

Author(s)

Sébastien Déjean and Ignacio González.

References

Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

Wold H. (1975). Path models with latent variables: The NIPALS approach. In: Blalock H. M. et al. (editors). *Quantitative Sociology: International perspectives on mathematical and statistical model building*. Academic Press, N.Y., 307-357.

See Also

svd, princomp, prcomp, eigen and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## Hilbert matrix
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
X.na <- X <- hilbert(9)[, 1:6]

## Hilbert matrix with missing data
idx.na <- matrix(sample(c(0, 1, 1, 1, 1), 36, replace = TRUE), ncol = 6)
X.na[idx.na == 0] <- NA
X.rec <- nipals(X.na, reconst = TRUE)$rec
round(X, 2)
round(X.rec, 2)
```

---

nutrimouse *Nutrimouse Dataset*

Description

The nutrimouse dataset contains the expression measure of 120 genes potentially involved in nutritional problems and the concentrations of 21 hepatic fatty acids for forty mice.

Usage

```
data(nutrimouse)
```

Format

A list containing the following components:

gene data frame with 40 observations on 120 numerical variables.

lipid data frame with 40 observations on 21 numerical variables.

diet factor of 5 levels containing 40 labels for the diet factor.

genotype factor of 2 levels containing 40 labels for the diet factor.

Details

The data sets come from a nutrigenomic study in the mouse (Martin *et al.*, 2007) in which the effects of five regimens with contrasted fatty acid compositions on liver lipids and hepatic gene expression in mice were considered. Two sets of variables were acquired on forty mice:

- gene: expressions of 120 genes measured in liver cells, selected (among about 30,000) as potentially relevant in the context of the nutrition study. These expressions come from a nylon macroarray with radioactive labelling;

- lipid: concentrations (in percentages) of 21 hepatic fatty acids measured by gas chromatography.

Biological units (mice) were cross-classified according to two factors experimental design (4 replicates):

- Genotype: 2-levels factor, wild-type (WT) and PPARα -/- (PPAR).
- Diet: 5-levels factor. Oils used for experimental diets preparation were corn and colza oils (50/50) for a reference diet (REF), hydrogenated coconut oil for a saturated fatty acid diet (COC), sunflower oil for an Omega6 fatty acid-rich diet (SUN), linseed oil for an Omega3-rich diet (LIN) and corn/colza/enriched fish oils for the FISH diet (43/43/14).

Source

The `nutrimouse` dataset was provided by Pascal Martin from the Toxicology and Pharmacology Laboratory, National Institute for Agronomic Research, French.

References

Martin, P. G. P., Guillou, H., Lasserre, F., Déjean, S., Lan, A., Pascussi, J.-M., San Cristobal, M., Legrand, P., Besse, P. and Pineau, T. (2007). Novel aspects of PPARα-mediated regulation of lipid and xenobiotic metabolism revealed through a multrigenomic study. *Hepatology* **54**, 767-777.

`pca` *Principal Components Analysis*

Description

Performs a principal components analysis on the given data matrix that can contain missing values. If data are complete 'pca' uses Singular Value Decomposition, if there are some missing values, it uses the NIPALS algorithm.

Usage

```
pca(X, ncomp = 3, center = TRUE, scale = FALSE,
    comp.tol = NULL, max.iter = 500, tol = 1e-09)
```

Arguments

`X`	a numeric matrix (or data frame) which provides the data for the principal components analysis. It can contain missing values.
`ncomp`	integer, if data is complete `ncomp` decides the number of components and associated eigenvalues to display from the `pcasvd` algorithm and if the data has missing values, `ncomp` gives the number of components to keep to perform the reconstitution of the data using the NIPALS algorithm. If `NULL`, function sets `ncomp = min(nrow(X), ncol(X))`

center	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of X can be supplied. The value is passed to scale.
scale	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with prcomp function, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of X can be supplied. The value is passed to scale.
comp.tol	a value indicating the magnitude below which components should be omitted.
max.iter	integer, the maximum number of iterations in the NIPALS algorithm.
tol	a positive real, the tolerance used in the NIPALS algorithm.

Details

The calculation is done either by a singular value decomposition of the (possibly centered and scaled) data matrix, if the data is complete or by using the NIPALS algorithm if there is data missing. Unlike princomp, the print method for these objects prints the results in a nice format and the plot method produces a bar plot of the percentage of variance explaned by the principal components (PCs).

Note that scale= TRUE cannot be used if there are zero or constant (for center = TRUE) variables.

Components are omitted if their standard deviations are less than or equal to comp.tol times the standard deviation of the first component. With the default null setting, no components are omitted. Other settings for comp.tol could be comp.tol = sqrt(.Machine$double.eps), which would omit essentially constant components, or comp.tol = 0.

Value

pca returns a list with class "pca" and "prcomp" containing the following components:

ncomp	the number of principal components used.
sdev	the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix or by using NIPALS.
rotation	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors).
X	if retx is true the value of the rotated data (the centred (and scaled if requested) data multiplied by the rotation matrix) is returned.
center, scale	the centering and scaling used, or FALSE.

Author(s)

Ignacio Gonz?lez.

See Also

nipals, prcomp, biplot, plotIndiv, plotVar, plot3dIndiv, plot3dVar and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details..

Examples

```
data(multidrug)

## this data set contains missing values, therefore
## the 'prcomp' function cannot be applied
pca.res <- pca(multidrug$ABC.trans, ncomp = 4, scale = TRUE)
plot(pca.res)
print(pca.res)
biplot(pca.res, xlabs = multidrug$cell.line$Class, cex = 0.7)

# samples representation
plotIndiv(pca.res, ind.names = multidrug$cell.line$Class, cex = 0.5,
          col = as.numeric(as.factor(multidrug$cell.line$Class)))
## Not run:
plot3dIndiv(pca.res, cex = 0.2,
            col = as.numeric(as.factor(multidrug$cell.line$Class)))

## End(Not run)
# variables representation
plotVar(pca.res, var.label = TRUE)
## Not run:
plot3dVar(pca.res, rad.in = 0.5, var.label = TRUE, cex = 0.5)

## End(Not run)
```

---

pcatune *Tune the number of principal components in PCA*

Description

This function has been renamed `tune.pca`.

---

perf *Compute evaluation criteria for PLS, sPLS, PLS-DA and sPLS-DA*

Description

Function to evaluate the performance of the fitted PLS, sparse PLS, PLS-DA and sparse PLS-DA models using various criteria.

Usage

```
## S3 method for class 'pls'
perf(object,
            criterion = c("all", "MSEP", "R2", "Q2"),
```

```
            validation = c("Mfold", "loo"),
           folds = 10, progressBar = TRUE, ...)

## S3 method for class 'spls'
perf(object,
          criterion = c("all", "MSEP", "R2", "Q2"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)

## S3 method for class 'plsda'
perf(object,
        method.predict = c("all", "max.dist", "centroids.dist", "mahalanobis.dist"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)

## S3 method for class 'splsda'
perf(object,
        method.predict = c("all", "max.dist", "centroids.dist", "mahalanobis.dist"),
          validation = c("Mfold", "loo"),
          folds = 10, progressBar = TRUE, ...)
```

Arguments

`object`	object of class inheriting from `"pls"`, `"plsda"`, `"spls"` or `"splsda"`. The function will retrieve some key parameters stored in that object.
`criterion`	only applies to object inheriting from `"pls"`, and `"spls"`, the criteria measures to be calculated (see details). Can be set to either `"all"`, `"MSEP"`, `"R2"`, `"Q2"`. By default set to `"all"`
`method.predict`	only applies to an object inheriting from `"plsda"` or `"splsda"` to evaluate the classification performance of the model. Should be a subset of `"max.dist"`, `"centroids.dist"`, `"mahalanobis.dist"`. Default is `"all"`. See `predict`.
`validation`	character. What kind of (internal) validation to use, matching one of `"Mfold"` or `"loo"` (see below). Default is `"Mfold"`.
`folds`	the folds in the Mfold cross-validation. See Details.
`progressBar`	by default set to `TRUE` to output the progress bar of the computation.
`...`	arguments to pass to `nearZeroVar`.

Details

For fitted PLS and sPLS regression models, `perf` estimates the mean squared error of prediction (MSEP), R^2, and Q^2 to assess the predictive perfity of the model using M-fold or leave-one-out cross-validation. Note that only the `classic`, `regression` and `invariant` modes can be applied.

If `validation = "Mfold"`, M-fold cross-validation is performed. How many folds to generate is selected by specifying the number of folds in `folds`. The folds also can be supplied as a list of vectors containing the indexes defining each fold as produced by `split`. If `validation = "loo"`, leave-one-out cross-validation is performed.

For fitted PLS-DA and sPLS-DA models, perf estimates the classification error rate using cross-validation. How many folds to generate is selected such that there is at least 1 sample for each class in the test set.

For the sparse approaches (sPLS and sLDA), note that the perf function will retrieve the keepX and keepY inputs from the previously run object. The sPLS or sPLS-DA functions will be run again on several and different subsets of data (the cross-folds) and certainly on different subset of selected features. For sPLS, the MSEP, R^2, and Q^2 criteria are averaged across all folds. For sPLS-DA, the classification erro rate is averaged across all folds. A feature stability measure is output for the user to assess how often the variables are selected across all folds.

Value

For PLS and sPLS models, perf produces a list with the following components:

MSEP	Mean Square Error Prediction for each Y variable, only applies to object inherited from "pls", and "spls".
R2	a matrix of R^2 values of the Y-variables for models with $1, \ldots,$ncomp components, only applies to object inherited from "pls", and "spls".
Q2	if Y containts one variable, a vector of Q^2 values else a list with a matrix of Q^2 values for each Y-variable. Note that in the specific case of an sPLS model, it is better to have a look at the Q2.total criterion, only applies to object inherited from "pls", and "spls"
Q2.total	a vector of Q^2-total values for models with $1, \ldots,$ncomp components, only applies to object inherited from "pls", and "spls"
features	a list of features selected across the folds ($stable.X and $stable.Y) or on the whole data set ($final) for the keepX and keepY parameters from the input object.
error.rate	For PLS-DA and sPLS-DA models, perf produces a matrix of classification error rate estimation. The dimensions correspond to the components in the model and to the prediction method used, respectively. Note that error rates reported in any component include the performance of the model in earlier components for the specified keepX parameters (e.g. error rate reported for component 3 for keepX = 20 already includes the fitted model on components 1 and 2 for keepX = 20). For more advanced usage of the perf function, see http://perso.math.univ-toulouse.fr/mixomics/methods/spls-da/ and consider using the predict function.

Author(s)

S\'ebastien D\'ejean, Ignacio Gonz'alez, Amrit Singh, Kim-Anh L\^e Cao.

References

Tenenhaus, M. (1998). *La r\'egression PLS: th\'eorie et pratique*. Paris: Editions Technic.

L\^e Cao, K. A., Rossouw D., Robert-Grani\'e, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Mevik, B.-H., Cederkvist, H. R. (2004). Mean Squared Error of Prediction (MSEP) Estimates for Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR). *Journal of Chemometrics* **18**(9), 422-429.

See Also

predict, nipals, plot.perf and http://perso.math.univ-toulouse.fr/mixomics/ for more details.

Examples

```
## validation for objects of class 'pls' (regression)
# ----------------------------------------
## Not run:
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic


# try tune the number of component to choose
# ---------------------
# first learn the full model
liver.pls <- pls(X, Y, ncomp = 10)

# with 5-fold cross validation: we use the same parameters as in model above
# but we perform cross validation to compute the MSEP, Q2 and R2 criteria
# --------------------------
liver.val <- perf(liver.pls, validation = "Mfold", folds = 5)

# Q2 total should decrease until it reaches a threshold
liver.val$Q2.total

# ncomp = 3 is enough
plot(liver.val$Q2.total, type = 'l', col = 'red', ylim = c(-0.1, 0.5),
xlab = 'PLS components', ylab = 'Q2 total')
abline(h = 0.0975, col = 'darkgreen')
legend('topright', col = c('red', 'darkgreen'), legend = c('Q2 total', 'threshold 0.0975')
, lty = 1)
title('Liver toxicity PLS 5-fold, Q2 values')

#have a look at the other criteria
# ----------------------
# R2
liver.val$R2
matplot(t(liver.val$R2), type = 'l', xlab = 'PLS components', ylab = 'R2 for each variable')
title('Liver toxicity PLS 5-fold, R2 values')

# MSEP
liver.val$MSEP
matplot(t(liver.val$MSEP), type = 'l', xlab = 'PLS components', ylab = 'MSEP for each variable')
title('Liver toxicity PLS 5-fold, MSEP values')
```

```
## validation for objects of class 'spls' (regression)
# ----------------------------------------
ncomp = 7
# first, learn the model on the whole data set
model.spls = spls(X, Y, ncomp = ncomp, mode = 'regression',
 keepX = c(rep(5, ncomp)), keepY = c(rep(2,ncomp)))


# with leave-one-out cross validation
set.seed(45)
model.spls.loo.val <- perf(model.spls, validation = "loo")

#Q2 total
model.spls.loo.val$Q2.total

# R2:we can see how the performance degrades when ncomp increases
# results are similar to 5-fold
model.spls.loo.val$R2


## validation for objects of class 'splsda' (classification)
# ----------------------------------------
data(srbct)
X <- srbct$gene
Y <- srbct$class

ncomp = 5

srbct.splsda <- splsda(X, Y, ncomp = ncomp, keepX = rep(10, ncomp))

# with Mfold
# ---------
set.seed(45)
error <- perf(srbct.splsda, validation = "Mfold", folds = 8,
               method.predict = "all")

plot(error, type = "l")

## End(Not run)
```

`pheatmap.multilevel` *Clustered heatmap*

Description

A function to draw clustered heatmaps from the pheatmap package.

Usage

```
## S3 method for class 'splsda1fact'
pheatmap.multilevel(result, cluster = NULL,
                    color=colorRampPalette(rev(c("#D73027",
                    "#FC8D59", "#FEE090", "#FFFFBF",
                    "#E0F3F8", "#91BFDB", "#4575B4")))(100),
                    col_sample=NULL,
                    col_stimulation=NULL,
                    label_annotation=NULL,
                    breaks = NA,
                    border_color = "grey60",
                    cellwidth = NA,
                    cellheight = NA,
                    scale = "none",
                    cluster_rows = TRUE,
                    cluster_cols = TRUE,
                    clustering_distance_rows = "euclidean",
                    clustering_distance_cols = "euclidean",
                    clustering_method = "complete",
                    treeheight_row = ifelse(cluster_rows, 50, 0),
                    treeheight_col = ifelse(cluster_cols, 50, 0),
                    legend = TRUE, annotation = NA,
                    annotation_colors = NA,
                    annotation_legend = TRUE,
                    show_rownames = TRUE,
                    show_colnames = TRUE,
                    fontsize = 10,
                    fontsize_row = fontsize,
                    fontsize_col = fontsize,
                    filename = NA,
                    width = NA,
                    height = NA,
                    order_sample=NULL,
                    ...)
## S3 method for class 'splsda2fact'
pheatmap.multilevel(result,  cluster = NULL,
                    color=colorRampPalette(rev(c("#D73027",
                    "#FC8D59", "#FEE090", "#FFFFBF",
                    "#E0F3F8", "#91BFDB", "#4575B4")))(100),
                    col_sample=NULL,
                    col_stimulation=NULL,
                    col_time=NULL,
                    label_color_stimulation=NULL,
                    label_color_time=NULL,
                    label_annotation=NULL,
                    breaks = NA,
                    border_color = "grey60",
                    cellwidth = NA,
```

```
                    cellheight = NA,
                    scale = "none",
                    cluster_rows = TRUE,
                    cluster_cols = TRUE,
                    clustering_distance_rows = "euclidean",
                    clustering_distance_cols = "euclidean",
                    clustering_method = "complete",
                    treeheight_row = ifelse(cluster_rows, 50, 0),
                    treeheight_col = ifelse(cluster_cols, 50, 0),
                    legend = TRUE,
                    annotation = NA,
                    annotation_colors = NA,
                    annotation_legend = TRUE,
                    show_rownames = TRUE,
                    show_colnames = TRUE,
                    fontsize = 10,
                    fontsize_row = fontsize,
                    fontsize_col = fontsize,
                    filename = NA,
                    width = NA,
                    height = NA,
                    order_sample=NULL,
                    ...)
```

Arguments

result	a mode result from function multilevel
cluster	by default set to NULL
col_sample	vector of colors indicating the color of each individual
col_stimulation	vector of colors indicating the color of each condition
col_time	if two-factor analysis, vector of colors indicating the color of the factors of the second condition
label_color_stimulation	character vector indicating the label of the first factor, see details
label_color_time	character vector indicating the label of the second factor, see details
label_annotation	set to NULL by default.
order_sample	vector indicatin the reordering of the samples, set to NULL by default.
color	vector of colors used in heatmap.
breaks	a sequence of numbers that covers the range of values in mat and is one element longer than color vector. Used for mapping values to colors. Useful, if needed to map certain values to certain colors, to certain values. If value is NA then the breaks are calculated automatically.
border_color	color of cell borders on heatmap, use NA if no border should be drawn.

cellwidth	individual cell width in points. If left as NA, then the values depend on the size of plotting window.
cellheight	individual cell height in points. If left as NA, then the values depend on the size of plotting window.
scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. Corresponding values are "row", "column" and "none"
cluster_rows	boolean values determining if rows should be clustered,
cluster_cols	boolean values determining if columns should be clustered.
clustering_distance_rows	distance measure used in clustering rows. Possible values are "correlation" for Pearson correlation and all the distances supported by dist, such as "euclidean", etc. If the value is none of the above it is assumed that a distance matrix is provided.
clustering_distance_cols	distance measure used in clustering columns. Possible values the same as for clustering_distance_rows.
clustering_method	clustering method used. Accepts the same values as hclust.
treeheight_row	the height of a tree for rows, if these are clustered. Default value 50 points.
treeheight_col	the height of a tree for columns, if these are clustered. Default value 50 points.
legend	boolean value that determines if legend should be drawn or not.
annotation	data frame that specifies the annotations shown on top of the columns. Each row defines the features for a specific column. The columns in the data and rows in the annotation are matched using corresponding row and column names. Note that color schemes takes into account if variable is continuous or discrete.
annotation_colors	list for specifying annotation track colors manually. It is possible to define the colors for only some of the features. Check examples for details.
annotation_legend	boolean value showing if the legend for annotation tracks should be drawn.
show_rownames	boolean specifying if column names are be shown.
show_colnames	boolean specifying if column names are be shown.
fontsize	base fontsize for the plot
fontsize_row	fontsize for rownames (Default: fontsize)
fontsize_col	fontsize for colnames (Default: fontsize)
filename	file path where to save the picture. Filetype is decided by the extension in the path. Currently following formats are supported: png, pdf, tiff, bmp, jpeg. Even if the plot does not fit into the plotting window, the file size is calculated so that the plot would fit there, unless specified otherwise.
width	manual option for determining the output file width in
height	manual option for determining the output file height in inches.
...	graphical parameters for the text used in plot. Parameters passed to grid.text, see gpar.

Details

This function has been borrowed from the `pheatmap` function of the pheatmap package. See `help(pheatmap)` for more details about the arguments of the function.

In the `multilevel` function, the factors indicated in the vector or the data frame cond must match the arguments `label_color_stimulation` and `label_color_time`, see example below.

Author(s)

Raivo Kolde <rkolde@gmail.com> (pheatmap), Benoit Liquet, Kim-Anh LÃª Cao.

References

On multilevel analysis: Liquet, B., LÃª Cao, K.-A., Hocini, H. and Thiebaut, R. (2012) A novel approach for biomarker selection and the integration of repeated measures experiments from two platforms. *BMC Bioinformatics* **13**:325.

Westerhuis, J. A., van Velzen, E. J., Hoefsloot, H. C., and Smilde, A. K. (2010). Multivariate paired data analysis: multilevel PLSDA versus OPLSDA. *Metabolomics*, **6**(1), 119-128.

See Also

`multilevel`, `pheatmap` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details..

Examples

```
## First example: one-factor analysis with sPLS-DA
data(data.simu) # simulated data
res.1level <- multilevel(data.simu$X, cond = data.simu$stimu, sample=data.simu$sample, ncomp=3,
                         keepX=c(200,200,200),tab.prob.gene=NULL, method = 'splsda')

#color for heatmap
col.stimu = as.numeric(data.simu$stimu)
col.stimu <- c("darkblue","purple","green4","red3")
col.sample <- c("lightgreen", "red","lightblue","darkorange",
  "purple","maroon","blue","chocolate","turquoise","tomato1",
  "pink2","aquamarine")

pheatmap.multilevel(res.1level, col_sample=col.sample, col_stimulation=col.stimu,
              label_annotation=NULL,border=FALSE,clustering_method="ward",
              show_colnames = FALSE,show_rownames = TRUE,fontsize_row=2)


## Second example: two-factor analysis with sPLS-DA
## Not run:
data(data.simu) # simulated data

time = factor(rep(c(rep('t1', 6), rep('t2', 6)), 4))
stimu.time = data.frame(cbind(as.character(data.simu$stimu), as.character(time)))
repeat.simu2 = rep(c(1:6), 8)

res.2level = multilevel(data.simu$X, cond = stimu.time,sample=repeat.simu2,
```

```
                        ncomp=2,keepX=c(200, 200),tab.prob.gene=NULL, method = 'splsda')

# color for plotIndiv
col.stimu = as.numeric(data.simu$stimu)
col.sample = c("lightgreen", "red","lightblue","darkorange","purple","maroon") # 6 samples
col.time= c("pink","lightblue1") # two time points
col.stimu = c('green', 'black', 'red', 'blue') # 4 stimulations
label.stimu = unique(data.simu$stimu)
label.time = unique(time)

pheatmap.multilevel(res.2level,
              col_sample=col.sample,
              col_stimulation=col.stimu,
              col_time=col.time,
              label_color_stimulation=label.stimu,
              label_color_time=label.time,
              label_annotation=NULL,border=FALSE,
              clustering_method="ward",
              show_colnames = FALSE,
              show_rownames = TRUE,
              fontsize_row=2)

## End(Not run)
```

`plot.perf` *Plot for model performance*

Description

Function to plot performance criteria, such as MSEP, RMSEP, R^2 or Q^2, as a function of the number of components.

Usage

```
## S3 method for class 'perf'
plot(x, criterion = c("MSEP", "RMSEP", "R2", "Q2"),
     pred.method = "all",
     xlab = "number of components", ylab = NULL,
     LimQ2 = 0.0975, LimQ2.col = "darkgrey",
     cTicks = NULL, layout = NULL, ...)
```

Arguments

`x`	an `perf` object.
`criterion`	character string. What type of validation criterion to plot for `pls` or `spls`. One of `"MSEP"`, `"RMSEP"`, `"R2"` or `"Q2"`. See `perf`.
`pred.method`	prediction method applied in `perf` for `plsda` or `splsda`. See `perf`.

xlab, ylab	titles for x and y axes. Typically character strings, but can be expressions (e.g., expression(R^2)).
LimQ2	numeric value. Signification limit for the components in the model. Default is LimQ2 = 0.0975.
LimQ2.col	character string specifying the color for the LimQ2 line to be plotted. If "none" the line will not be plotted.
cTicks	integer vector. Axis tickmark locations for the used number of components. Default is 1:ncomp (see perf).
layout	numeric vector of length two giving the number of rows and columns in a multi panel display. If not specified, plot.perf tries to be intelligent.
...	Further arguments sent to xyplot function.

Details

plot.perf creates one plot for each response variable in the model, laid out in a multi panel display. It uses xyplot for performing the actual plotting.

Author(s)

Ignacio Gonz?lez.

See Also

pls, spls, plsda, splsda, perf.

Examples

```
require(lattice)

## validation for objects of class 'pls' or 'spls'
## Not run:
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic

liver.pls <- pls(X, Y, ncomp = 3)
liver.perf <- perf(liver.pls, validation = "Mfold")

plot(liver.perf, criterion = "R2", type = "l", layout = c(2, 2))

## End(Not run)
```

plot.rcc *Canonical Correlations Plot*

Description

This function provides scree plot of the canonical correlations.

Usage

```
## S3 method for class 'rcc'
plot(x, scree.type = c("pointplot", "barplot"), ...)
```

Arguments

x	object of class inheriting from "rcc".
scree.type	character string, (partially) matching one of "pointplot" or "barplot", determining the kind of scree plots to be produced.
...	arguments to be passed to other methods. For the "pointplot" type see points, for "barplot" type see barplot.

Author(s)

Sébastien Déjean and Ignacio González.

See Also

points, barplot, par.

Examples

```
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, lambda1 = 0.064, lambda2 = 0.008)

## 'pointplot' type scree
plot(nutri.res) #(default)

plot(nutri.res, pch = 19, cex = 1.2,
     col = c(rep("red", 3), rep("darkblue", 18)))

## 'barplot' type scree
plot(nutri.res, scree.type = "barplot")

plot(nutri.res, scree.type = "barplot", density = 20, col = "black")
```

plot3dIndiv *Plot of Individuals (Experimental Units) in three dimensions*

Description

This function provides scatter plots of individuals (experimental units) in three dimensions for (r)CCA, (s)PLS regression and PCA.

Usage

```
## S3 method for class 'rcc'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            rep.space = "XY-variate",
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'pls'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            rep.space = "X-variate",
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'splsda'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            rep.space = "X-variate",
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'plsda'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            rep.space = "X-variate",
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'spls'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            rep.space = "X-variate",
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'pca'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
```

```
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)

## S3 method for class 'spca'
plot3dIndiv(object, comp = 1:3, ind.names = FALSE,
            xlab = NULL, ylab = NULL, zlab = NULL,
            col = "blue", cex = 1, pch = "s", font = 1,
            axes.box = "box", ...)
```

Arguments

`object`	object of class inheriting from `"rcc"`, `"pls"`, `"plsda"`, `"spls"`, `"plsda"` or `"pca"`.
`comp`	an integer vector of length 3, the components that will be used on the x-axis, the y-axis and z-axis respectively to project the individuals.
`ind.names`	either a character vector of names for the individuals to be plotted, or `FALSE` for no names. If `TRUE`, the row names of the first (or second) data matrix is used as names (see Details).
`rep.space`	character string, (partially) matching one of `"X-variate"`, `"Y-variate"` or `"XY-variate"`, determining the subspace to project the individuals. Defaults to `"XY-variate"` and `"X-variate"` for (r)CCA and (s)PLS respectively.
`xlab, ylab, zlab`	x-, y- and z-axis titles.
`col`	character (or item) color to be used, possibly vector (see Details).
`cex`	numeric character (or item) expansion, possibly vector (see Details).
`pch`	character indicating the type of item to plot, possibly vector (see Details).
`font`	character font to be used, possibly vector (see Details).
`axes.box`	character string specifying the axes box type. Should be a of `c("box", "bbox", "both")`.
`...`	further title parameters are passed to `title3d`.

Details

`plot3dIndiv` method makes scatter plot for individuals representation in three dimensions. Each item corresponds to an individual.

If `ind.names=TRUE` and row names is `NULL`, then `ind.names=1:n`, where `n` is the number of individuals.

The arguments `col`, `cex`, `pch` and `font` can be atomic vectors or vectors of length `n`. If atomic, this argument value determines the graphical attribute for all the individuals. In the last case, multiple arguments values can be specified so that each item (individual) can be given its own graphic attributes. Default values exist for this arguments. Supported types for `pch` are: `"s"` for spheres, `"t"` for tetrahedrons, `"c"` for cubes, `"o"` for octahedrons, `"i"` for icosahedrons and `"d"` dodecahedrons.

The pointing device of your graphics user-interface can also be used to set the viewpoint interactively. With the pointing device the buttons are by default set as follows: - left adjust viewpoint position - middle adjust field of view angle - right or wheel adjust zoom factor

Author(s)

Ignacio Gonzalez

See Also

plotIndiv, plot3dVar, text3d, title3d and rgl.postscript to save the screen shot to a file in PostScript or other vector graphics format.

Examples

```
require(rgl)

## Not run:
## plot3d of individuals for objects of class 'rcc'
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)

col = nutrimouse$diet
font = c(rep(1, 20), rep(3, 20))
plot3dIndiv(nutri.res, ind.names = nutrimouse$diet,
                axes.box = "box", font = font, col = col)

pch = c(rep("s", 20), rep("t", 20))
plot3dIndiv(nutri.res, ind.names = FALSE, axes.box = "both",
                col = col, cex = 1.5, pch = pch)

## plot3d of individuals for objects of class 'pls' or 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))

Time.Group = liver.toxicity$treatment[, "Time.Group"]
col <- rep(c("blue", "red", "darkgreen", "darkviolet"), rep(16, 4))
plot3dIndiv(toxicity.spls, ind.names = Time.Group,
                 col = col, cex = 0.8)

col <- rainbow(48)[Time.Group]
plot3dIndiv(toxicity.spls, ind.names = FALSE,
                 col = col, cex = 0.3, axes.box = "both")

## plot3d of individuals for objects of class 'pca'
data(multidrug)
pca.res <- pca(multidrug$ABC.trans, ncomp = 4, scale = TRUE)

palette(rainbow(9))
col = as.numeric(as.factor(multidrug$cell.line$Class))
plot3dIndiv(pca.res, cex = 0.25, col = col)
palette("default")
```

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

---

plot3dVar *Plot of Variables in three dimensions*

Description

This function provides variables representation in three dimensions for (regularized) CCA, (sparse) PLS regression and PCA.

Usage

```
## S3 method for class 'rcc'
plot3dVar(object, comp = 1:3, rad.in = 1,
          cutoff = NULL, X.label = FALSE, Y.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)

## S3 method for class 'pls'
plot3dVar(object, comp = 1:3, rad.in = 1,
          X.label = FALSE, Y.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)

## S3 method for class 'spls'
plot3dVar(object, comp = 1:3, rad.in = 1,
          X.label = FALSE, Y.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)

## S3 method for class 'plsda'
plot3dVar(object, comp = 1:3, rad.in = 1,
          var.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)

## S3 method for class 'splsda'
plot3dVar(object, comp = 1:3, rad.in = 1,
          var.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)
```

```
## S3 method for class 'pca'
plot3dVar(object, comp = 1:3, rad.in = 1,
      var.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)

## S3 method for class 'spca'
plot3dVar(object, comp = 1:3, rad.in = 1,
          var.label = FALSE,
          pch = NULL, cex = NULL, col = NULL, font = NULL,
          axes.box = "all", label.axes.box = "both",
          xlab = NULL, ylab = NULL, zlab = NULL, ...)
```

Arguments

`object`	object of class inheriting from `"rcc"`, `"pls"`, `"plsda"`, `"spls"`, `"plsda"`, `"pca"` or `"spca"`.
`comp`	an integer vector of length 3, the components that will be used on the x-axis, the y-axis and z-axis respectively to project the variables.
`rad.in`	numeric between 0 and 1, the radius of the small sphere. Defaults to 1.
`cutoff`	numeric between 0 and 1. Variables with correlations below this cutoff in absolute value are not plotted (see Details).
`X.label, Y.label, var.label`	either a character vector of names for the variables or `FALSE` for no names. If `TRUE`, the columns names of the matrice are used as labels.
`col`	character or integer vector of colors for plotted character and items. See Details.
`pch`	character indicating the type of item to plot, possibly vector (see Details).
`cex`	numeric character (or item) expansion (see Details).
`font`	numeric font to be used. See `text3d` for details.
`axes.box`	character string or vector specifying the axes box type. Should be a subset of `c("axes", "box", "bbox", "all")`.
`label.axes.box`	character string indicating whether to label the axes and/or boxes. Should be a of `c("axes", "box", "both")`
`xlab, ylab, zlab`	x-, y- and z-axis titles.
`...`	further title parameters are passed to `title3d`.

Details

`plot3dVar` gives an extension of the "correlation circle" to three dimensions, i.e. the correlations between each variable and the selected components are plotted as scatter plot in three dimensions.

For the objects of class rcc, pls and spls a sphere of radius given by rad.in centred in the origin is displayed. If cutoff is not NULL the rad.in is ignored and the radius of the sphere is equal to cutoff.

For plsda and splsda objects, only the X variables are represented.

For spls and splsda objects, only the X and Y variables selected on dimensions comp are represented.

The arguments cex, pch and font are vectors of length two for the objects of class rcc, pls and spls. The firts and second component value determines the graphical attribute for X- and Y-variables respectively. Default values exist for this arguments. Supported types for pch are: "s" for spheres, "t" for tetrahedrons, "c" for cubes, "o" for octahedrons, "i" for icosahedrons and "d" dodecahedrons.

The argument col can be either a vector of length two or a list with two vector components of length p and q respectively, where p is the number of X-variables and q is the number of Y-variables. In the first case, the first and second component of the vector determine the color for the X- and Y-variables respectively. Otherwise, multiple colors can be specified so that each item (variable) can be given its own color. In this case, the first component of the list correspond to the X-color attribut and the second component correspond to the Y-color attribute.

The pointing device of your graphics user-interface can also be used to set the viewpoint interactively. With the pointing device the buttons are by default set as follows: - left adjust viewpoint position - middle adjust field of view angle - right or wheel adjust zoom factor

Value

A list containing the following components:

coord.X	matrix of X-variables coordinates.
coord.Y	matrix of Y-variables coordinates.

Author(s)

Ignacio González.

See Also

plotVar, plot3dIndiv, text3d, title3d and rgl.postscript to save the screen shot to a file in PostScript or other vector graphics format.

Examples

```
require(rgl)

## Not run:
## 3D variable representation for objects of class 'rcc'
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)

\dontrun{
```

```
# default
plot3dVar(nutri.res)

# cutoff active, labeling the variables
plot3dVar(nutri.res, cutoff = 0.7, X.label = TRUE, cex = c(0.8, 0.8))
}
## 3D variable representation for objects of class 'pls' or 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxi.spls.1 <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                    keepY = c(10, 10, 10))
\dontrun{
plot3dVar(toxi.spls.1, rad.in = 0.5, keep.var = TRUE, cex = c(1, 0.8),
          main = "Variables 3D representation")
}
toxi.spls.2 <- spls(X, Y, ncomp = 3, keepX = c(10, 10, 10),
                    keepY = c(10, 10, 10))

plot3dVar(toxi.spls.2, rad.in = 0.5, Y.label = TRUE,
          main = "Variables 3D representation",
          label.axes.box = "axes")

## 3D variable representation for objects of class 'pca'
data(multidrug)
pca.res <- pca(multidrug$ABC.trans, ncomp = 4, scale = TRUE)

plot3dVar(pca.res)

## variable representation for objects of class 'splsda'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- as.factor(liver.toxicity$treatment[, 4])

ncomp <- 3
keepX <- rep(20, ncomp)

splsda.liver <- splsda(X, Y, ncomp = ncomp, keepX = keepX)
\dontrun{
plot3dVar(splsda.liver, var.label = FALSE, Y.label = TRUE, keep.var = TRUE)
}

## End(Not run)
```

---

plotIndiv *Plot of Individuals (Experimental Units)*

Description

This function provides scatter plots for individuals (experimental units) representation in (regularized) CCA, (sparse) PLS regression, (sparse)RGCCA and PCA.

Usage

```
## S3 method for class 'rcc'
plotIndiv(object, comp = 1:2, ind.names = TRUE,
          rep.space = "XY-variate",
          X.label = NULL, Y.label = NULL,
          col = "black", cex = 1, pch = 1,
          abline.line = FALSE, ...)

## S3 method for class 'pls'
plotIndiv(object, comp = 1:2,
ind.names = TRUE,
          rep.space = "X-variate",
          X.label = NULL, Y.label = NULL,
          col = "black", cex = 1, pch = 1,
          abline.line = FALSE,...)

## S3 method for class 'spls'
plotIndiv(object, comp = 1:2, ind.names = TRUE,
          rep.space = "X-variate",
          X.label = NULL, Y.label = NULL,
          col = "black", cex = 1, pch = 1,
          abline.line = FALSE,...)

## S3 method for class 'pca'
plotIndiv(object, comp = 1:2, ind.names = TRUE,
          X.label = NULL, Y.label = NULL,
          abline.line = FALSE,...)

## S3 method for class 'spca'
plotIndiv(object, comp = 1:2, ind.names = TRUE,
          X.label = NULL, Y.label = NULL,
          abline.line = FALSE,...)

## S3 method for class 'rgcca'
plotIndiv(object, comp = 1:2, ind.names = TRUE, rep.space = 1,
    X.label = NULL, Y.label = NULL,  col = "black", cex = 1, pch = 1,
    abline.line = FALSE,...)

## S3 method for class 'sgcca'
plotIndiv(object, comp = 1:2, ind.names = TRUE, rep.space = 1,
    X.label = NULL, Y.label = NULL,  col = "black", cex = 1, pch = 1,
    abline.line = FALSE, ...)
```

Arguments

object	object of class inheriting from "rcc", "pls", "plsda", "spls", "plsda", "pca" or "spca".
comp	integer vector of length two. The components that will be used on the horizontal and the vertical axis respectively to project the individuals.
ind.names	either a character vector of names for the individuals to be plotted, or FALSE for no names. If TRUE, the row names of the first (or second) data matrix is used as names (see Details).
rep.space	For objects of class "rcc", "pls", "spls", character string, (partially) matching one of "X-variate", "Y-variate" or "XY-variate", determining the subspace to project the individuals. Defaults to "XY-variate" "rcc" object for and "X-variate" "pls" and "spls" objects. For objects of class "rgcca" and "sgcca", numerical value indicating the block data set form which to represent the individuals.
X.label, Y.label	x- and y-axis titles.
col	character (or symbol) color to be used, possibly vector.
cex	numeric character (or symbol) expansion, possibly vector.
pch	plot character. A character string or a vector of single characters or integers. See points for all alternatives.
abline.line	should the vertical and horizontal line through the center be plotted?default set to FALSE
...	further graphical parameters are passed to text.

Details

plotIndiv method makes scatter plot for individuals representation depending on the subspace of projection. Each point corresponds to an individual.

If ind.names=TRUE and row names is NULL, then ind.names=1:n, where n is the number of individuals.

The arguments col, cex and pch can be atomic vectors or vectors of length n. If atomic, this argument value determines the graphical attribute for all the individuals. In the last case, multiple arguments values can be specified so that each point (individual) can be given its own graphic attributes (see par). Default values exist for this arguments.

Author(s)

S?bastien D?jean and Ignacio Gonz?lez.

See Also

plot3dIndiv, text, points and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## plot of individuals for objects of class 'rcc'
# ------------------------------------------------------
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)

plotIndiv(nutri.res) #(default)

col <- rep(c("blue", "red"), c(20, 20))
plotIndiv(nutri.res, ind.names = nutrimouse$diet, col = col)
legend(-2.2, -1.1, c("WT", "PPAR"), pch = c(16, 16),
       col = c("blue", "red"), text.col = c("blue", "red"),
       cex = 1, pt.cex = c(1.2, 1.2))

## plot of individuals for objects of class 'pls' or 'spls'
# ------------------------------------------------------
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))

col <- rep(c("blue", "red", "darkgreen", "darkviolet"), rep(16, 4))
cex <- rep(c(1, 1.2, 1, 1.4), c(16, 16, 16, 16))
pch <- rep(c(15, 16, 17, 18), c(16, 16, 16, 16))
plotIndiv(toxicity.spls, comp = 1:2, ind.names = FALSE,
          rep.space = "X-variate", col = col, cex = cex, pch = pch)
legend("topright", c("50 mg/kg", "150 mg/kg", "1500 mg/kg", "2000 mg/kg"),
       col = c("blue", "red", "darkgreen", "darkviolet"),
       pch = c(15, 16, 17, 18), pt.cex = c(1, 1.2, 1, 1.4),
       title = "Treatment")

## variable representation for objects of class 'sgcca' (or 'rgcca')
# ------------------------------------------------------
data(nutrimouse)

# need to unmap the Y factor diet
Y = unmap(nutrimouse$diet)
data = list(nutrimouse$gene, nutrimouse$lipid,Y)
# with this design, gene expression and lipids are connected to the diet factor
# and gene expression and lipids are also connected
design = matrix(c(0,1,1,
                  1,0,1,
                  1,1,0), ncol = 3, nrow = 3, byrow = TRUE)

#note: the penalty parameters will need to be tuned
wrap.result.sgcca = wrapper.sgcca(data = data, design = design, penalty = c(.3,.5, 1),
                                  ncomp = c(2, 2, 1),
                                  scheme = "centroid", verbose = FALSE)
```

```
# on the first data set
plotIndiv(wrap.result.sgcca, rep.space = 1, ind.names = TRUE,
          col = as.numeric(nutrimouse$diet), cex = .6)
# on the second data set
plotIndiv(wrap.result.sgcca, rep.space = 2, ind.names = TRUE,
          col = as.numeric(nutrimouse$diet), cex = .6)
```

---

plotVar *Plot of Variables*

Description

This function provides variables representation for (regularized) CCA, (sparse) PLS regression, PCA nd (sparse) Regularized generalised CCA.

Usage

```
## S3 method for class 'rcc'
plotVar(object, comp = 1:2, rad.in = 0.5, cutoff = NULL,
        X.label = FALSE, Y.label = FALSE,
        pch = NULL, cex = NULL, col = NULL, font = NULL, ...)

## S3 method for class 'pls'
plotVar(object, comp = 1:2, rad.in = 0.5,
        X.label = FALSE, Y.label = FALSE, pch = NULL, cex = NULL,
        col = NULL, font = NULL, ...)

## S3 method for class 'plsda'
plotVar(object, comp = 1:2, rad.in = 0.5,
        var.label = FALSE, pch = NULL, cex = NULL, col = NULL,
        font = NULL, ...)

## S3 method for class 'spls'
plotVar(object, comp = 1:2, rad.in = 0.5,
        X.label = FALSE, Y.label = FALSE, pch = NULL, cex = NULL,
        col = NULL, font = NULL, ...)

## S3 method for class 'splsda'
plotVar(object, comp = 1:2, rad.in = 0.5,
        var.label = FALSE, pch = NULL, cex = NULL, col = NULL,
        font = NULL, ...)

## S3 method for class 'pca'
plotVar(object, comp = 1:2, rad.in = 0.5,
        var.label = FALSE, ...)
```

```
## S3 method for class 'spca'
plotVar(object, comp = 1:2, rad.in = 0.5,
        var.label = FALSE, pch = NULL, cex = NULL, col = NULL,
        font = NULL, ...)

## S3 method for class 'rgcca'
plotVar(object, comp = c(1,2), block = c(1,2),
           labels = FALSE, pch = c(16,17), cex =  c(0.5, 0.5),
           col = c('green', 'blue'), font = c(2,3), rad.in = 0.5,
           ...)

## S3 method for class 'sgcca'
plotVar(object,
comp = c(1,2),
block = c(1,2), ncomp.select = c(1,2),
labels = FALSE, pch = c(16,17),
cex =  c(0.5, 0.5),
col = c('green', 'blue'),
font = c(2,3),
rad.in = 0.5,...)
```

Arguments

object	object of class inheriting from "rcc", "pls", "plsda", "spls", "splsda", "pca" or "spca".
comp	integer vector of length two. The components that will be used on the horizontal and the vertical axis respectively to project the variables.
rad.in	numeric between 0 and 1, the radius of the inner circle. Defaults to 0.5.
cutoff	numeric between 0 and 1. Variables with correlations below this cutoff in absolute value are not plotted (see Details).
X.label, Y.label, var.label, labels	either a character vector of names for the variables or FALSE for no names. If TRUE, the columns names of the matrice are used as labels.
col	character or integer vector of colors for plotted character and symbols, can be of length 2 (one for each data set) or of length (p+q) (i.e. the total number of variables). See Details.
pch	plot character. A vector of single characters or integers, can be of length 2 (one for each data set) or of length (p+q) (i.e. the total number of variables). See points for all alternatives.
cex	numeric vector of character expansion sizes for the plotted character and symbols, can be of length 2 (one for each data set) or of length (p+q) (i.e. the total number of variables).
font	numeric vector of font to be used, can be of length 2 (one for each data set) or of length (p+q) (i.e. the total number of variables). See par for details.

block	for an object of class "rgcca" or "sgcca", a numerical vector indicating the block variables to display.
ncomp.select	for the sparse versions, an input vector indicating the components on which the variables were selected. Only those selected variables are displayed.
...	not used currently.

Details

plotVar produce a "correlation circle", i.e. the correlations between each variable and the selected components are plotted as scatter plot, with concentric circles of radius one et radius given by rad.in. Each point corresponds to a variable. For (regularized) CCA the components correspond to the equiangular vector between X- and Y-variates. For (sparse) PLS regression mode the components correspond to the X-variates. If mode is canonical, the components for X and Y variables correspond to the X- and Y-variates respectively.

For plsda and splsda objects, only the X variables are represented.

For spls and splsda objects, only the X and Y variables selected on dimensions comp are represented.

The arguments col, pch, cex and font can be either vectors of length two or a list with two vector components of length p and q respectively, where p is the number of X-variables and q is the number of Y-variables. In the first case, the first and second component of the vector determine the graphics attributes for the X- and Y-variables respectively. Otherwise, multiple arguments values can be specified so that each point (variable) can be given its own graphic attributes. In this case, the first component of the list correspond to the X attributs and the second component correspond to the Y attributs. Default values exist for this arguments.

Value

A list containing the following components:

coord.X	matrix of X-variables coordinates.
coord.Y	matrix of Y-variables coordinates.

Author(s)

S\'ebastien D\'ejean, Ignacio Gonz\'alez, Kim-Anh L\^e Cao.

References

Gonz\'alez I., L\^e Cao K-A., Davis, M.J. and D\'ejean, S. (2012). Visualising associations between paired ?omics? data sets. J. Data Mining 5:19. http://www.biodatamining.org/content/5/1/19/abstract

See Also

plot3dVar, cim, network, par and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## variable representation for objects of class 'rcc'
# ----------------------------------------------------
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)

plotVar(nutri.res) #(default)

plotVar(nutri.res, comp = 1:2, cutoff = 0.5,
        X.label = TRUE, Y.label = TRUE)

## variable representation for objects of class 'pls' or 'spls'
# ----------------------------------------------------
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))

plotVar(toxicity.spls, keep.var = TRUE, Y.label = TRUE, cex = c(1,0.8))

## variable representation for objects of class 'splsda'
# ----------------------------------------------------
## Not run:
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- as.factor(liver.toxicity$treatment[, 4])

ncomp <- 2
keepX <- rep(20, ncomp)

splsda.liver <- splsda(X, Y, ncomp = ncomp, keepX = keepX)
plotVar(splsda.liver, var.label = FALSE)

## End(Not run)

## variable representation for objects of class 'sgcca' (or 'rgcca')
# ----------------------------------------------------
data(nutrimouse)

# need to unmap the Y factor diet
Y = unmap(nutrimouse$diet)
data = list(nutrimouse$gene, nutrimouse$lipid,Y)
# with this design, gene expression and lipids are connected to the diet factor
# and gene expression and lipids are also connected
design = matrix(c(0,1,1,
                  1,0,1,
                  1,1,0), ncol = 3, nrow = 3, byrow = TRUE)

#note: the penalty parameters will need to be tuned
```

```
wrap.result.sgcca = wrapper.sgcca(data = data, design = design, penalty = c(.3,.5, 1),
                                  ncomp = c(2, 2, 1),
                                  scheme = "centroid", verbose = FALSE)
par(mfrow=c(2,2))
plotVar(wrap.result.sgcca, comp = c(1,2), block = c(1,2), ncomp.select = c(1,2))
plotVar(wrap.result.sgcca, comp = c(1,2), block = c(1,2), ncomp.select = c(1,2), labels = TRUE)
plotVar(wrap.result.sgcca, comp = c(1,2), block = c(1,2), ncomp.select = 1, labels = TRUE)
plotVar(wrap.result.sgcca, comp = c(1,2), block = 1, ncomp.select = 1, labels = TRUE)
par(mfrow=c(1,1))
```

`pls` *Partial Least Squares (PLS) Regression*

Description

Function to perform Partial Least Squares (PLS) regression.

Usage

```
pls(X, Y, ncomp = 2,
    mode = c("regression", "canonical", "invariant", "classic"),
    max.iter = 500, tol = 1e-06, near.zero.var = TRUE, ...)
```

Arguments

`X`	numeric matrix of predictors. `NA`s are allowed.
`Y`	numeric vector or matrix of responses (for multi-response models). `NA`s are allowed.
`ncomp`	the number of components to include in the model. Default to 2.
`mode`	character string. What type of algorithm to use, (partially) matching one of `"regression"`, `"canonical"`, `"invariant"` or `"classic"`. See Details.
`max.iter`	integer, the maximum number of iterations.
`tol`	a not negative real, the tolerance used in the iterative algorithm.
`near.zero.var`	boolean, see the internal `nearZeroVar` function (should be set to TRUE in particular for data with many zero values). Setting this argument to FALSE (when appropriate) will speed up the computations.
`...`	arguments to pass to `nearZeroVar`.

Details

`pls` function fit PLS models with $1, \ldots,$`ncomp` components. Multi-response models are fully supported. The `X` and `Y` datasets can contain missing values.

The type of algorithm to use is specified with the `mode` argument. Four PLS algorithms are available: PLS regression (`"regression"`), PLS canonical analysis (`"canonical"`), redundancy analysis (`"invariant"`) and the classical PLS algorithm (`"classic"`) (see References).

The estimation of the missing values can be performed by the reconstitution of the data matrix using the `nipals` function. Otherwise, missing values are handled by casewise deletion in the `pls` function without having to delete the rows with missing data.

Value

`pls` returns an object of class `"pls"`, a list that contains the following components:

`X`	the centered and standardized original predictor matrix.
`Y`	the centered and standardized original response vector or matrix.
`ncomp`	the number of components included in the model.
`mode`	the algoritthm used to fit the model.
`mat.c`	matrix of coefficients to be used internally by `predict`.
`variates`	list containing the X and Y variates.
`loadings`	list containing the estimated loadings for the variates.
`names`	list containing the names to be used for individuals and variables.
`nzv`	list containing the zero- or near-zero predictors information.
`tol`	the tolerance used in the iterative algorithm, used for subsequent S3 methods
`max.iter`	the maximum number of iterations, used for subsequent S3 methods

Author(s)

Sébastien Déjean and Ignacio González and Kim-Anh Lê Cao.

References

Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

See Also

`spls`, `summary`, `plotIndiv`, `plotVar`, `predict`, `perf` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.pls <- pls(X, Y, mode = "classic")

data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic

toxicity.pls <- pls(X, Y, ncomp = 3)
```

plsda *Partial Least Squares Discriminate Analysis (PLS-DA).*

Description

Function to perform standard Partial Least Squares regression to classify samples.

Usage

```
plsda(X, Y, ncomp = 2, max.iter = 500, tol = 1e-06, near.zero.var = TRUE, ...)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	a factor or a class vector for the discrete outcome.
ncomp	the number of components to include in the model. Default to 2.
max.iter	integer, the maximum number of iterations.
tol	a not negative real, the tolerance used in the iterative algorithm.
near.zero.var	boolean, see the internal nearZeroVar function (should be set to TRUE in particular for data with many zero values). Setting this argument to FALSE (when appropriate) will speed up the computations.
...	arguments to pass to nearZeroVar.

Details

plsda function fit PLS models with $1, ...,$ncomp components to the factor or class vector Y. The appropriate indicator matrix is created.

Value

plsda returns an object of class "plsda", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized indicator response vector or matrix.
ind.mat	the indicator matrix.
ncomp	the number of components included in the model.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the X and Y variates.
loadings	list containing the estimated loadings for the variates.
names	list containing the names to be used for individuals and variables.
nzv	list containing the zero- or near-zero predictors information.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods

Author(s)

Ignacio GonzÃ¡lez, Kim-Anh LÃª Cao and Pierre Monget.

References

PÃ©rez-Enciso, M. and Tenenhaus, M. (2003). Prediction of clinical outcome with microarray data: a partial least squares discriminant analysis (PLS-DA) approach. *Human Genetics* **112**, 581-592.

Nguyen, D. V. and Rocke, D. M. (2002). Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics* **18**, 39-50.

Tenenhaus, M. (1998). *La rÃ©gression PLS: thÃ©orie et pratique*. Paris: Editions Technic.

See Also

splsda, summary, plotIndiv, plotVar, plot3dIndiv, plot3dVar, predict, perf and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## First example
data(breast.tumors)
X <- breast.tumors$gene.exp
Y <- breast.tumors$sample$treatment

plsda.breast <- plsda(X, Y, ncomp = 2)
palette(c("red", "blue"))
col.breast <- as.numeric(as.factor(Y))
plotIndiv(plsda.breast, ind.names = TRUE, col = col.breast)
legend('bottomleft', c("After", "Before"), pch = c(16, 16),
       col = unique(col.breast), cex = 1, pt.cex = c(1.2, 1.2),
       title = "Treatment")
palette("default")

## Not run:
## Second example
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$treatment[, 4]

plsda.liver <- plsda(X, Y, ncomp = 2)
col.rat <- as.numeric(as.factor(Y))
plotIndiv(plsda.liver, col = col.rat, ind.names = Y)

## End(Not run)
```

predict *Predict Method for PLS, sPLS, PLS-DA or sPLS-DA*

Description

Predicted values based on PLS, sparse PLS, PLS-DA or sparse PLS-DA models. New responses and variates are predicted using a fitted model and a new matrix of observations.

Usage

```
## S3 method for class 'pls'
predict(object, newdata, ...)

## S3 method for class 'spls'
predict(object, newdata, ...)

## S3 method for class 'plsda'
predict(object, newdata, method = c("all", "max.dist",
        "centroids.dist", "mahalanobis.dist"), ...)

## S3 method for class 'splsda'
predict(object, newdata, method = c("all", "max.dist",
        "centroids.dist", "mahalanobis.dist"), ...)
```

Arguments

object	object of class inheriting from "pls", "spls", "plsda" or "splsda".
newdata	data matrix in which to look for for explanatory variables to be used for prediction.
method	method to be applied for plsda or splsda to predict the class of new data, should be a subset of "centroids.dist", "mahalanobis.dist" or "max.dist" (see Details). Defaults to "all".
...	not used currently.

Details

predict produces predicted values, obtained by evaluating the PLS, sparse PLS, PLSDA or sparse PLSDA model returned by pls, spls, plsda or splsda in the frame newdata. Variates for newdata are also returned. The prediction values are calculated based on the regression coefficients of object$Y onto object$variates$X.

Different class prediction methods are proposed for plsda or splsda: "max.dist" is the naive method to predict the class. It is based on the predicted matrix (object$predict) which can be seen as a probability matrix to assign each test data to a class. The class with the largest class value is the predicted class. "centroids.dist" allocates the individual x to the class of Y minimizing $dist(\texttt{x-variate}, G_l)$, where G_l, $l = 1, ..., L$ are the centroids of the classes calculated on the X-variates of the model. "mahalanobis.dist" allocates the individual x to the class of Y as in "centroids.dist" but by using the Mahalanobis metric in the calculation of the distance.

Value

`predict` produces a list with the following components:

`predict`	a three dimensional array of predicted response values. The dimensions correspond to the observations, the response variables and the model dimension, respectively.
`variates`	matrix of predicted variates.
`B.hat`	matrix of regression coefficients (without the intercept).
`class`	vector or matrix of predicted class by using $1, \ldots,$`ncomp` (sparse)PLS-DA components.
`centroids`	matrix of coordinates for centroids.

Author(s)

S\'ebastien D\'ejean, Ignacio Gonz'alez, Kim-Anh L\^e Cao and Pierre Monget

References

Tenenhaus, M. (1998). *La r?gression PLS: th?orie et pratique*. Paris: Editions Technic.

See Also

`pls`, `spls`, `plsda`, `splsda` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.pls <- pls(X, Y, ncomp = 2, mode = "classic")

indiv1 <- c(200, 40, 60)
indiv2 <- c(190, 45, 45)
newdata <- rbind(indiv1, indiv2)
colnames(newdata) <- colnames(X)
newdata

pred <- predict(linn.pls, newdata)

plotIndiv(linn.pls, comp = 1:2, rep.space = "X-variate")
points(pred$variates[, 1], pred$variates[, 2], pch = 19, cex = 1.2)
text(pred$variates[, 1], pred$variates[, 2],
     c("new ind.1", "new ind.2"), pos = 3)

## First example with plsda
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- as.factor(liver.toxicity$treatment[, 4])
```

```
## if training is perfomed on 4/5th of the original data
samp <- sample(1:5, nrow(X), replace = TRUE)
test <- which(samp == 1)   # testing on the first fold
train <- setdiff(1:nrow(X), test)

plsda.train <- plsda(X[train, ], Y[train], ncomp = 2)
test.predict <- predict(plsda.train, X[test, ], method = "max.dist")
Prediction <- levels(Y)[test.predict$class$max.dist[, 2]]
cbind(Y = as.character(Y[test]), Prediction)

## Not run:
## Second example with splsda
splsda.train <- splsda(X[train, ], Y[train], ncomp = 2, keepX = c(30, 30))
test.predict <- predict(splsda.train, X[test, ], method = "max.dist")
Prediction <- levels(Y)[test.predict$class$max.dist[, 2]]
cbind(Y = as.character(Y[test]), Prediction)

## End(Not run)
```

---

print *Print Methods for CCA, (s)PLS, PCA and Summary objects*

Description

Produce print methods for class "rcc", "pls", "spls", "pca", "rgcca", "sgcca" and "summary".

Usage

```
## S3 method for class 'rcc'
print(x, ...)

## S3 method for class 'pls'
print(x, ...)

## S3 method for class 'spls'
print(x, ...)

## S3 method for class 'pca'
print(x, ...)

## S3 method for class 'spca'
print(x, ...)

## S3 method for class 'rgcca'
print(x, ...)

## S3 method for class 'sgcca'
print(x, ...)
```

```
## S3 method for class 'summary'
print(x, ...)
```

Arguments

x	object of class inheriting from "rcc", "pls", "spls", "pca", "spca", "rgcca", "sgcca"or "summary".
...	not used currently.

Details

print method for "rcc", "pls", "spls" "pca", "rgcca", "sgcca" class, returns a description of the x object including: the function used, the regularization parameters (if x of class "rcc"), the (s)PLS algorithm used (if x of class "pls" or "spls"), the samples size, the number of variables selected on each of the sPLS components (if x of class "spls") and the available components of the object.

print method for "summary" class, gives the (s)PLS algorithm used (if x of class "pls" or "spls"), the number of variates considered, the canonical correlations (if x of class "rcc"), the number of variables selected on each of the sPLS components (if x of class "spls") and the available components for Communalities Analysis, Redundancy Analysis and Variable Importance in the Projection (VIP).

Author(s)

Sébastien Déjean, Ignacio González and Kim-Anh Lê Cao.

See Also

rcc, pls, spls, vip.

Examples

```
## print for objects of class 'rcc'
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)
print(nutri.res)

## print for objects of class 'summary'
more <- summary(nutri.res, cutoff = 0.65)
print(more)

## print for objects of class 'pls'
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.pls <- pls(X, Y)
print(linn.pls)
```

```
## print for objects of class 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))
print(toxicity.spls)
```

`prostate` *Human Prostate Tumors Data*

Description

This data set contains the expression of 6,033 genes derived from 52 prostate tumors and from 50 non tumor prostate samples (referred to as normal) using oligonucleotide microarrays.

Usage

```
data(prostate)
```

Format

A list containing the following components:

`data` data matrix with 102 rows and 6033 columns. Each row represents an experimental sample, and each column a single gene.

`type` a vector containing the clinical status (normal or tumor).

Details

This study investigated whether gene expression differences could distinguished common clinical and pathological features of prostate cancer. Expression profiles were derived from 52 prostate tumors and from 50 non tumour prostate samples (referred to as normal) using oligonucleotide microarrays containing probes for approximately 12,600 genes and ESTs. After preprocessing remains the expression of 6033 genes.

References

Singh D, Febbo P, Ross K, Jackson D, Manola J, Ladd C, Tamayo P, Renshaw A, D'Amico A, Richie J, et al. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, **1**(2):203–209.

rcc *Regularized Canonical Correlation Analysis*

Description

The function performs the regularized extension of the Canonical Correlation Analysis to seek correlations between two data matrices.

Usage

```
rcc(X, Y, ncomp = 2, lambda1 = 0, lambda2 = 0)
```

Arguments

X	numeric matrix or data frame $(n \times p)$, the observations on the X variables. NAs are allowed.
Y	numeric matrix or data frame $(n \times q)$, the observations on the Y variables. NAs are allowed.
ncomp	the number of components to include in the model. Default to 2.
lambda1, lambda2	a not negative real. The regularization parameter for the *X* and *Y* data. Defaults to lambda1=lambda2=0.

Details

The main purpose of Canonical Correlations Analysis (CCA) is the exploration of sample correlations between two sets of variables X and Y observed on the same individuals (experimental units) whose roles in the analysis are strictly symmetric.

The cancor function performs the core of computations but additional tools are required to deal with data sets highly correlated (nearly collinear), data sets with more variables than units by example.

The rcc function, the regularized version of CCA, is one way to deal with this problem by including a regularization step in the computations of CCA. Such a regularization in this context was first proposed by Vinod (1976), then developped by Leurgans *et al.* (1993). It consists in the regularization of the empirical covariances matrices of X and Y by adding a multiple of the matrix identity, that is, $\text{Cov}(X) + \lambda_1 I$ and $\text{Cov}(Y) + \lambda_2 I$.

When lambda1=0 and lambda2=0, rcc perform classic CCA, if posible.

The estimation of the missing values can be performed by the reconstitution of the data matrix using the nipals function. Otherwise, missing values are handled by casewise deletion in the rcc function.

Value

rcc returns a object of class "rcc", a list that contains the following components:

X	the original X data.

`Y`	the original Y data.
`lambda`	a vector containing the regularization parameters.
`cor`	a vector containing the canonical correlations.
`loadings`	list containing the estimated loadings for the X and Y canonical variates.
`variates`	list containing the canonical variates.
`names`	list containing the names to be used for individuals and variables.

Author(s)

Sébastien Déjean and Ignacio González.

References

Leurgans, S. E., Moyeed, R. A. and Silverman, B. W. (1993). Canonical correlation analysis when the data are curves. *Journal of the Royal Statistical Society. Series B* **55**, 725-740.

Vinod, H. D. (1976). Canonical ridge and econometrics of joint production. *Journal of Econometrics* **6**, 129-137.

See Also

`summary`, `tune.rcc`, `plot.rcc`, `plotIndiv`, `plot3dIndiv`, `plotVar`, `plot3dVar`, `cim`, `network` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## Classic CCA
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.res <- rcc(X, Y)

## Regularized CCA
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)
```

---

s.match *Plot of Paired Coordinates*

Description

performs the scatter diagram for a paired coordinates.

Usage

```
s.match(df1xy, df2xy, xax = 1, yax = 2, pch = 20, cpoint = 1,
        label = row.names(df1xy), clabel = 1, edge = TRUE,
        xlim = NULL, ylim = NULL, grid = FALSE, addaxes = TRUE,
        cgrid = 1, include.origin = TRUE, origin = c(0, 0),
        sub = "", csub = 1.25, possub = "bottomleft",
        pixmap = NULL, contour = NULL, area = NULL,
        add.plot = FALSE, col, lty = 1)
```

Arguments

df1xy	a data frame containing two columns from the first system.
df2xy	a data frame containing two columns from the second system.
xax	the column number for the x-axis of both the two systems.
yax	the column number for the y-axis of both the two systems.
pch	if cpoint > 0, an integer specifying the symbol or the single character to be used in plotting points.
cpoint	a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn.
label	a vector of strings of characters for the couple labels.
clabel	if not NULL, a character size for the labels, used with par("cex")*clabel.
edge	if TRUE the arrows are plotted, otherwise only the segments are drawn.
xlim	the ranges to be encompassed by the x axis, if NULL they are computed.
ylim	the ranges to be encompassed by the y axis, if NULL they are computed.
grid	a logical value indicating whether a grid in the background of the plot should be drawn.
addaxes	a logical value indicating whether the axes should be plotted.
cgrid	a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid.
include.origin	a logical value indicating whether the point "origin" should be belonged to the graph space.
origin	the fixed point in the graph space, for example c(0,0) the origin axes.
sub	a string of characters to be inserted as legend.
csub	a character size for the legend, used with par("cex")*csub.
possub	character string indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomri
pixmap	an object pixmap displayed in the map background.
contour	a data frame with 4 columns to plot the contour of the map : each row gives a segment (x1, y1, x2, y2).
area	a data frame of class 'area' to plot a set of surface units in contour.
add.plot	if TRUE uses the current graphics window.
col	vector of size n (the number of samples) to indicate the biological conditions of each sample.
lty	set by default to 1.

Details

Graphical of the samples (individuals) is displayed in a superimposed manner where each sample will be indicated using an arrow. The start of the arrow indicates the location of the sample in X in one plot, and the tip the location of the sample in Y in the other plot. Short arrows indicate if both data sets strongly agree and long arrows a disagreement between the two data sets.

Value

The matched call.

Author(s)

Daniel Chessel from **ade4 R** package, modified by Kim-Anh Lê Cao.

References

Lê Cao, K.-A., Martin, P.G.P., Robert-Granié, C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

Examples

```
## relevant only for canonical mode
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, mode = "canonical", ncomp = 3,
                      keepX = c(50, 50, 50), keepY = c(10, 10, 10))

color.toxicity <- as.numeric(liver.toxicity$treatment[, 2])
label.toxicity <- liver.toxicity$treatment[, 1]
s.match(toxicity.spls$variates$X[, c(1, 2)],
        toxicity.spls$variates$Y[, c(1, 2)],
        clabel = 0.5, label = label.toxicity, col = color.toxicity)
```

---

scatterutil *Graphical utility functions from ade4*

Description

These are utilities used in graphical functions.

Details

The functions scatter use some utilities functions :

scatterutil.base defines the layer of the plot for all scatters.

scatterutil.eti puts labels centred on the points.

scatterutil.grid plots a grid and adds a legend.

Author(s)

Daniel Chessel, Stephane Dray <dray@biomserv.univ-lyon1.fr>

See Also

s.match

`select.var` *Output of selected variables*

Description

This function outputs the selected variables on each component for the sparse versions of the approaches.

Usage

```
## S3 method for class 'spls'
select.var(object, comp = 1, ...)
## S3 method for class 'splsda'
select.var(object, comp = 1, ...)
## S3 method for class 'spca'
select.var(object, comp = 1, ...)
## S3 method for class 'sipca'
select.var(object, comp = 1, ...)
## S3 method for class 'sgcca'
select.var(object, block = NULL, comp = 1,  ...)
```

Arguments

`object`	object of class inheriting from `"spls"`, `"splsda"`, `"spca"`, `"sipca"`.
`comp`	integer value indicating the component of interest.
`block`	for an object of class `"sgcca"`, the block data sets can be specified as an input vector, for example `c(1,2)` for the first two blocks. Default to NULL (all block data sets)
`...`	other arguments.

Details

`select.var` provides the variables selected on a given component. \

`name` outputs the name of the selected variables (provided that the input data have colnames) ranked in decreasing order of importance.

`value` outputs the loading value for each selected variable, the loadings are ranked according to their absolute value.

These functions are only implemented for the sparse versions.

Author(s)

Kim-Anh L\^e Cao.

Examples

```
data(liver.toxicity)
X = liver.toxicity$gene
Y = liver.toxicity$clinic

# example with sPCA
# ------------------
liver.spca <- spca(X, ncomp = 1, keepX = 10)
select.var(liver.spca, comp = 1)$name
select.var(liver.spca, comp = 1)$value

#example with sIPCA
# -----------------
## Not run:
liver.sipca <- sipca(X, ncomp = 3, keepX = rep(10, 3))
select.var(liver.sipca, comp = 1)

## End(Not run)

# example with sPLS
# -----------------
## Not run:
liver.spls = spls(X, Y, ncomp = 2, keepX = c(20, 40),keepY = c(5, 5))
select.var(liver.spls, comp = 2)

# example with sPLS-DA
data(srbct)   # an example with no gene name in the data
X = srbct$gene
Y = srbct$class

srbct.splsda = splsda(X, Y, ncomp = 2, keepX = c(5, 10))
select = select.var(srbct.splsda, comp = 2)
select
# this is a very specific case where a data set has no rownames.
srbct$gene.name[substr(select$select, 2,5),]

## End(Not run)
```

```
# example with sGCCA
# -----------------
## Not run:
data(nutrimouse)

# ! need to unmap the Y factor
Y = unmap(nutrimouse$diet)
data = list(nutrimouse$gene, nutrimouse$lipid,Y)
# in this design, gene expression and lipids are connected to the diet factor
# and gene expression and lipids are also connected
design = matrix(c(0,1,1,
                  1,0,1,
                  1,1,0), ncol = 3, nrow = 3, byrow = T)
#note: the penalty parameters need to be tuned
wrap.result.sgcca = wrapper.sgcca(data = data, design = design, penalty = c(.3,.5, 1),
                                  ncomp = c(2, 2, 1),
                                  scheme = "centroid", verbose = FALSE)

select = select.var(wrap.result.sgcca, comp = 1)
# loading value of the variables selected on the first block
select$value.var[[1]]

## End(Not run)
```

---

sipca *Independent Principal Component Analysis*

Description

Performs sparse independent principal component analysis on the given data matrix to enable variable selection.

Usage

```
sipca(X, ncomp, mode = c("deflation","parallel"),
          fun = c("logcosh", "exp"),
          scale = FALSE, max.iter = 200,
          tol = 1e-04, keepX = rep(50,ncomp),
          w.init=NULL)
```

Arguments

X	a numeric matrix (or data frame) which provides the data for the principal component analysis.
ncomp	integer, number of independent component to choose. Set by default to 3.

mode	character string. What type of algorithm to use when estimating the unmixing matrix, (partially) matching one of "deflation", "parallel". Default set to deflation.
fun	the function used in approximation to neg-entropy in the FastICA algorithm. Default set to logcosh, see details of FastICA.
scale	a logical value indicating whether rows of the data matrix X should be standardized beforehand.
max.iter	integer, maximum number of iterations to perform.
tol	a positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged, see fastICA package.
keepX	the number of variable to keep on each dimensions.
w.init	initial un-mixing matrix (unlike FastICA, this matrix is fixed here).

Details

See Details of ipca.

Soft thresholding is implemented on the independent loading vectors to obtain sparse loading vectors and enable variable selection.

Value

pca returns a list with class "ipca" containing the following components:

ncomp	the number of principal components used.
unmixing	the unmixing matrix of size (ncomp x ncomp)
mixing	the mixing matrix of size (ncomp x ncomp
X	the centered data matrix
x	the principal components (with sparse independent loadings)
loadings	the sparse independent loading vectors
kurtosis	the kurtosis measure of the independent loading vectors

Author(s)

Fangzhou Yao and Jeff Coquery.

References

Yao, F., Coquery, J. and Lê Cao, K.-A.(2011) Principal component analysis with independent loadings: a combination of PCA and ICA. (in preparation)

A. Hyvarinen and E. Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*, **13(4-5)**:411-430

J L Marchini, C Heaton and B D Ripley (2010). fastICA: FastICA Algorithms to perform ICA and Projection Pursuit. R package version 1.1-13.

See Also

ipca, pca, plotIndiv, plotVar, plot3dIndiv, plot3dVar and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(liver.toxicity)

# implement IPCA on a microarray dataset
sipca.res <- sipca(liver.toxicity$gene, ncomp = 3, mode="deflation", keepX=c(50,50,50))
sipca.res

# samples representation
plotIndiv(sipca.res, ind.names = liver.toxicity$treatment[, 4], cex = 0.5,
          col = as.numeric(as.factor(liver.toxicity$treatment[, 4])))
## Not run:
plot3dIndiv(sipca.res, cex = 0.01,
            col = as.numeric(as.factor(liver.toxicity$treatment[, 4])))

## End(Not run)
# variables representation
plotVar(sipca.res, var.label = TRUE, cex = 0.5)
## Not run:
plot3dVar(sipca.res, rad.in = 0.5, var.label = TRUE, cex = 0.5)

## End(Not run)
```

---

spca *Sparse Principal Components Analysis*

Description

Performs a sparse principal components analysis to perform variable selection by using singular value decomposition.

Usage

```
spca(X, ncomp = 3, center = TRUE, scale = TRUE,
     keepX = rep(ncol(X),ncomp), max.iter = 500,
     tol = 1e-06)
```

Arguments

X	a numeric matrix (or data frame) which provides the data for the sparse principal components analysis.
ncomp	integer, the number of components to keep.
center	a logical value indicating whether the variables should be shifted to be zero centered. Alternatively, a vector of length equal the number of columns of X can be supplied. The value is passed to scale.

scale	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is TRUE. See details.
max.iter	integer, the maximum number of iterations to check convergence in each component.
tol	a positive real, the tolerance used in the iterative algorithm.
keepX	numeric vector of length ncomp, the number of variables to keep in loading vectors. By default all variables are kept in the model. See details.

Details

The calculation employs singular value decomposition of the (centered and scaled) data matrix and LASSO to generate sparsity on the loading vectors.

scale= TRUE is highly recommended as it will help obtaining orthogonal sparse loading vectors.

keepX is the number of variables to keep in loading vectors. The difference between number of columns of X and keepX is the degree of sparsity, which refers to the number of zeros in each loading vector.

Note that spca does not apply to the data matrix with missing values. The biplot function for spca is not available.

Value

spca returns a list with class "spca" containing the following components:

ncomp	the number of components to keep in the calculation.
varX	the adjusted cumulative percentage of variances explained.
keepX	the number of variables kept in each loading vector.
iter	the number of iterations needed to reach convergence for each component.
rotation	the matrix containing the sparse loading vectors.
x	the matrix containing the principal components.

Author(s)

Kim-Anh L? Cao, Fangzhou Yao, Leigh Coonan

References

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis* **99**, 1015-1034.

See Also

pca and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(liver.toxicity)
spca.rat <- spca(liver.toxicity$gene, ncomp = 3, keepX = rep(50, 3))
spca.rat

## variable representation
plotVar(spca.rat, X.label = TRUE, cex = 0.5)
## Not run: plot3dVar(spca.rat)

## samples representation
plotIndiv(spca.rat, ind.names = liver.toxicity$treatment[, 3], cex = 0.5,
          col = as.numeric(liver.toxicity$treatment[, 3]))
## Not run: plot3dIndiv(spca.rat, cex = 0.01,
            col = as.numeric(liver.toxicity$treatment[, 3]))
## End(Not run)
```

---

spls *Sparse Partial Least Squares (sPLS)*

Description

Function to perform sparse Partial Least Squares (sPLS). The sPLS approach combines both integration and variable selection simultaneously on two data sets in a one-step strategy.

Usage

```
spls(X, Y, ncomp = 2, mode = c("regression", "canonical"),
     max.iter = 500, tol = 1e-06, keepX = rep(ncol(X), ncomp),
     keepY = rep(ncol(Y), ncomp), near.zero.var = TRUE,...)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	numeric vector or matrix of responses (for multi-response models). NAs are allowed.
ncomp	the number of components to include in the model (see Details). Default is set to from one to the rank of X.
mode	character string. What type of algorithm to use, (partially) matching one of "regression" or "canonical". See Details.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
keepY	numeric vector of length ncomp, the number of variables to keep in Y-loadings. By default all variables are kept in the model.

near.zero.var	boolean, see the internal nearZeroVar function (should be set to TRUE in particular for data with many zero values). Setting this argument to FALSE (when appropriate) will speed up the computations.
...	arguments to pass to nearZeroVar.

Details

spls function fit sPLS models with $1, \ldots,$ncomp components. Multi-response models are fully supported. The X and Y datasets can contain missing values.

The type of algorithm to use is specified with the mode argument. Two sPLS algorithms are available: sPLS regression ("regression") and sPLS canonical analysis ("canonical") (see References).

The estimation of the missing values can be performed by the reconstitution of the data matrix using the nipals function. Otherwise, missing values are handled by casewise deletion in the spls function without having to delete the rows with missing data.

Value

spls returns an object of class "spls", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized original response vector or matrix.
ncomp	the number of components included in the model.
mode	the algorithm used to fit the model.
keepX	number of X variables kept in the model on each component.
keepY	number of Y variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.
nzv	list containing the zero- or near-zero predictors information.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods

Author(s)

Sébastien Déjean, Ignacio González and Kim-Anh Lê Cao.

References

Lê Cao, K.-A., Martin, P.G.P., Robert-Granié, C. and Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics* **10**:34.

Lê Cao, K.-A., Rossouw, D., Robert-Granié, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis* **99**, 1015-1034.

Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*. Paris: Editions Technic.

Wold H. (1966). Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P. R. (editors), *Multivariate Analysis*. Academic Press, N.Y., 391-420.

See Also

pls, summary, plotIndiv, plotVar, plot3dIndiv, plot3dVar, cim, network, predict, perf and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic

toxicity.spls <- spls(X, Y, ncomp = 2, keepX = c(50, 50),
                      keepY = c(10, 10))
```

---

splsda *Sparse Partial Least Squares Discriminate Analysis (sPLS-DA)*

Description

Function to perform sparse Partial Least Squares to classify samples. The sPLS-DA approach embeds variable selection for this purpose.

Usage

```
splsda(X, Y, ncomp = 2, keepX = rep(ncol(X), ncomp),
       max.iter = 500, tol = 1e-06, near.zero.var = TRUE, ...)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	a factor or a class vector for the discrete outcome.
ncomp	the number of components to include in the model (see Details).
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
max.iter	integer, the maximum number of iterations.
tol	a positive real, the tolerance used in the iterative algorithm.
near.zero.var	boolean, see the internal nearZeroVar function (should be set to TRUE in particular for data with many zero values). Setting this argument to FALSE (when appropriate) will speed up the computations.
...	arguments to pass to nearZeroVar.

Details

splsda function fit sPLS models with $1, \ldots,$ncomp components to the factor or class vector Y. The appropriate indicator matrix is created.

Value

splsda returns an object of class "splsda", a list that contains the following components:

X	the centered and standardized original predictor matrix.
Y	the centered and standardized indicator response vector or matrix.
ind.mat	the indicator matrix.
ncomp	the number of components included in the model.
keepX	number of X variables kept in the model on each component.
mat.c	matrix of coefficients to be used internally by predict.
variates	list containing the variates.
loadings	list containing the estimated loadings for the X and Y variates.
names	list containing the names to be used for individuals and variables.
nzv	list containing the zero- or near-zero predictors information.
tol	the tolerance used in the iterative algorithm, used for subsequent S3 methods
max.iter	the maximum number of iterations, used for subsequent S3 methods

Author(s)

Ignacio González, Kim-Anh Lê Cao and Pierre Monget.

References

On sPLS-DA: Lê Cao, K.-A., Boitard, S. and Besse, P. (2011). Sparse PLS Discriminant Analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* **12**:253.

See Also

spls, summary, plotIndiv, plotVar, plot3dIndiv, plot3dVar, cim, network, predict, perf and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
## First example
data(breast.tumors)
X <- breast.tumors$gene.exp
Y <- breast.tumors$sample$treatment

res <- splsda(X, Y, ncomp = 2, keepX = c(25, 25))
palette(c("red", "blue"))
col.breast <- as.numeric(as.factor(Y))
plotIndiv(res, ind.names = TRUE, col = col.breast)
```

```
legend('bottomleft', c("After", "Before"), pch = c(16, 16),
       col = unique(col.breast), cex = 1, pt.cex = c(1.2, 1.2),
       title = "Treatment")
palette("default")

## Second example
## Not run:
data(liver.toxicity)
X <- as.matrix(liver.toxicity$gene)
Y <- liver.toxicity$treatment[, 4]

splsda.liver = splsda(X, Y, ncomp = 2, keepX = c(20, 20))
col.rat <- as.numeric(as.factor(Y))
plotIndiv(splsda.liver, col = col.rat, ind.names = Y)

## End(Not run)
```

`srbct` *Small version of the small round blue cell tumors of childhood data*

Description

This data set from Khan *et al.*, (2001) gives the expression measure of 2308 genes measured on 63 samples.

Usage

```
data(srbct)
```

Format

A list containing the following components:

`gene` data frame with 63 rows and 2308 columns. The expression measure of 2308 genes for the 63 subjects.

`class` A class vector containing the class tumour of each case (4 classes in total).

`gene.name` data frame with 2308 rows and 2 columns containing further information on the genes.

Source

http://research.nhgri.nih.gov/microarray/Supplement

References

Khan et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine* **7**, Number 6, June.

summary *Summary Methods for CCA and PLS objects*

Description

Produce summary methods for class "rcc", "pls" and "spls".

Usage

```
## S3 method for class 'rcc'
summary(object, what = c("all", "communalities", "redundancy"),
        cutoff = NULL, digits = 4, ...)

## S3 method for class 'pls'
summary(object, what = c("all", "communalities", "redundancy",
        "VIP"), digits = 4, keep.var = FALSE, ...)

## S3 method for class 'spls'
summary(object, what = c("all", "communalities", "redundancy",
        "VIP"), digits = 4, keep.var = FALSE, ...)
```

Arguments

object	object of class inheriting from "rcc", "pls" or "spls".
cutoff	real between 0 and 1. Variables with all correlations components below this cutoff in absolute value are not showed (see Details).
digits	integer, the number of significant digits to use when printing. Defaults to 4.
what	character string or vector. Should be a subset of c("all", "summarised", "communalities", "redundancy", "VIP"). "VIP" is only available for (s)PLS. See Details.
keep.var	boolean. If TRUE only the variables with loadings not zero (as selected by spls) are showed. Defaults to FALSE.
...	not used currently.

Details

The information in the rcc, pls or spls object is summarised, it includes: the dimensions of X and Y data, the number of variates considered, the canonical correlations (if object of class "rcc") and the (s)PLS algorithm used (if object of class "pls" or "spls") and the number of variables selected on each of the sPLS components (if x of class "spls").

"communalities" in what gives Communalities Analysis. "redundancy" display Redundancy Analysis. "VIP" gives the Variable Importance in the Projection (VIP) coefficients fit by pls or spls. If what is "all", all are given.

For class "rcc", when a value to cutoff is specified, the correlations between each variable and the equiangular vector between X- and Y-variates are computed. Variables with at least one correlation

componente bigger than cutoff are showed. The defaults is cutoff=NULL all the variables are given.

Value

The function summary returns a list with components:

ncomp	the number of components in the model.
cor	the canonical correlations.
cutoff	the cutoff used.
keep.var	list containing the name of the variables selected.
mode	the algoritm used in pls or spls.
Cm	list containing the communalities.
Rd	list containing the redundancy.
VIP	matrix of VIP coefficients.
what	subset of c("all", "communalities", "redundancy", "VIP").
digits	the number of significant digits to use when printing.
method	method used: rcc, pls or spls.

Author(s)

Sébastien Déjean Ignacio González and Kim-Anh Lê Cao.

See Also

rcc, pls, spls, vip.

Examples

```
## summary for objects of class 'rcc'
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene
nutri.res <- rcc(X, Y, ncomp = 3, lambda1 = 0.064, lambda2 = 0.008)
more <- summary(nutri.res, cutoff = 0.65)

## summary for objects of class 'pls'
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.pls <- pls(X, Y)
more <- summary(linn.pls)

## summary for objects of class 'spls'
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
toxicity.spls <- spls(X, Y, ncomp = 3, keepX = c(50, 50, 50),
                      keepY = c(10, 10, 10))
more <- summary(toxicity.spls, what = "redundancy", keep.var = TRUE)
```

tune.multilevel *Tuning functions for multilevel analyses*

Description

These functions were implemented to help tuning the variable selection parameters in the multilevel analyses.

Usage

```
tune.multilevel(X, Y = NULL,
                    cond = NULL,
                    sample,
                    ncomp=1,
                    test.keepX=c(5, 10, 15),
                    test.keepY=NULL,
                    already.tested.X = NULL,
                    already.tested.Y = NULL,
                    method = NULL,
                    dist,
                    validation = c("Mfold", "loo"),
                    folds = 10)
```

Arguments

X	numeric matrix of predictors. NAs are allowed.
Y	if(method = 'spls') numeric vector or matrix of continuous responses (for multi-response models) NAs are allowed.
cond	a factor or a class vector for one-factor discrete outcome, a numeric matrix of 2 columns for two-factor discrete outcome. See Details
sample	a vector indicating the repeated measured on each individual.
ncomp	the number of components to include in the model.
test.keepX	numeric vector for the different number of variables to test from the X data set
test.keepY	If method = 'spls', numeric vector for the different number of variables to test from the Y data set
already.tested.X	if ncomp > 1 numeric vector indicating the number of variables to select rom the X data set on the previous ncomp-1 components
already.tested.Y	if method = 'spls' and if(ncomp > 1) numeric vector indicating the number of variables to select rom the Y data set on the previous ncomp-1 components
method	character string. Which multivariate method and type of analysis to choose, matching one of 'splsda' (Discriminant Analysis) or 'spls' (unsupervised integrative analysis). See Details.

`dist`	distance metric to use for `splsda` to estimate the classification error rate, should be a subset of `"centroids.dist"`, `"mahalanobis.dist"` or `"max.dist"` (see Details).
`validation`	character. What kind of (internal) validation to use, matching one of `"Mfold"` or `"loo"` (see below). Default is `"Mfold"`.
`folds`	the folds in the Mfold cross-validation. See Details.

Details

This tuning function should be used to tune the parameters in the `multilevel` function.

If `method = 'splsda'`, a distance metric must be used, see `help(predict.splsda)` for details about the distances.

For a sPLS-DA one-factor analysis, leave-one-out cross-validation is performed, internally the training data is decomposed into within-subject variation.

For a sPLS-DA two-factor analysis, the correlation between components from the within-subject variation of X and the `cond` matrix is computed on the whole data set.

For a sPLS-DA two-factor analysis, the correlation between components from the within-subject variation of X and Y is computed on the whole data set.

If `validation = "Mfold"`, M-fold cross-validation is performed. How many folds to generate is selected by specifying the number of folds in `folds`. The folds also can be supplied as a list of vectors containing the indexes defining each fold as produced by `split`.

If `validation = "loo"`, leave-one-out cross-validation is performed. By default `folds` is set to the number of unique individuals.

Value

Depending on the type of analysis performed, a list that contains:

`error`	leave-one-out cross-validation is performed for one-factor sPLS-DA analysis.
`cor.value`	compute the correlation between latent variables for two-factor sPLS-DA analysis or sPLS.

Author(s)

Benoit Liquet, Kim-Anh LÃª Cao.

References

On multilevel analysis: Liquet, B., LÃª Cao, K.-A., Hocini, H. and Thiebaut, R. A novel approach for biomarker selection and the integration of repeated measures experiments from two platforms. *Submitted*.

Westerhuis, J. A., van Velzen, E. J., Hoefsloot, H. C., and Smilde, A. K. (2010). Multivariate paired data analysis: multilevel PLSDA versus OPLSDA. *Metabolomics*, **6**(1), 119-128.

See Also

`multilevel` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details..

Examples

```
## First example: one-factor analysis with sPLS-DA
## Not run:
data(data.simu) # simulated data
result.ex1 = tune.multilevel(data.simu$X,
                             cond = data.simu$stimu,
                             sample = data.simu$sample,
                             ncomp=2,
                             test.keepX=c(5, 10, 15),
                             already.tested.X = c(50),
                             method = 'splsda',
                             dist = 'mahalanobis.dist',
                             validation = 'loo')

result.ex1

## End(Not run)

## Second example: two-factor analysis with sPLS-DA
## Not run:
data(liver.toxicity)
dose = liver.toxicity$treatment$Dose.Group
time = liver.toxicity$treatment$Time.Group
dose.time = cbind(dose, time)
repeat.indiv = c(1,2, 1,  2,  1,  2,  1,  2,  3,  3,  4,
  3,  4,  3,  4,  4,  5,  6,  5,  5,  6,  5,  6,  7,  7,
  8,  6,  7,  8,  7,  8,  8,  9, 10,  9, 10, 11, 9,  9,
  10, 11, 12, 12, 10, 11, 12, 11, 12, 13, 14, 13, 14, 13,
  14, 13, 14, 15, 16, 15, 16, 15, 16, 15, 16)


result.ex2 = tune.multilevel (liver.toxicity$gene,
                             cond = dose.time,
                             sample = repeat.indiv,
                             ncomp=2,
                             test.keepX=c(5, 10, 15),
                             already.tested.X = c(50),
                             method = 'splsda',
       dist = 'mahalanobis.dist')
result.ex2

## End(Not run)

## Third example: one-factor integrative analysis with sPLS
## Not run:
result.ex3 = tune.multilevel (liver.toxicity$gene, liver.toxicity$clinic,
                             cond = dose,
                             sample = repeat.indiv,
                             ncomp=2,
                             test.keepX=c(5, 10, 15),
                             test.keepY=c(2,3),
                             already.tested.X = c(50), already.tested.Y = c(5),
```

```
                                method = 'spls')

result.ex3

## End(Not run)
```

---

tune.pca *Tune the number of principal components in PCA*

Description

tune.pca can be used to quickly visualise the proportion of explained variance for a large number of principal components in PCA.

Usage

```
tune.pca(X, ncomp = NULL, center = TRUE, scale = FALSE,
        max.iter = 500, tol = 1e-09)
```

Arguments

X	a numeric matrix (or data frame) which provides the data for the principal components analysis. It can contain missing values.
ncomp	integer, the number of components to initially analyse in tune.pca to choose a final ncomp for pca. If NULL, function sets ncomp = min(nrow(X), ncol(X))
center	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of X can be supplied. The value is passed to scale.
scale	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with prcomp function, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of X can be supplied. The value is passed to scale.
max.iter	integer, the maximum number of iterations for the NIPALS algorithm.
tol	a positive real, the tolerance used for the NIPALS algorithm.

Details

The calculation is done either by a singular value decomposition of the (possibly centered and scaled) data matrix, if the data is complete or by using the NIPALS algorithm if there is data missing. Unlike princomp, the print method for these objects prints the results in a nice format and the plot method produces a bar plot of the percentage of variance explained by the principal components (PCs).

When using NIPALS (missing values), we make the assumption that the first (min(ncol(X), nrow(X)) principal components will account for 100 % of the explained variance.

Note that scale= TRUE cannot be used if there are zero or constant (for center = TRUE) variables.

Value

tune.pca returns a list with class "tune.pca" containing the following components:

var	the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix).
prop.var	the proportion of explained variance accounted for by each principal component is calculated using the eigenvalues
cum.var	the cumulative proportion of explained variance accounted for by the sequential accumulation of principal components is calculated using the sum of the proportion of explained variance

Author(s)

Ignacio González and Leigh Coonan

See Also

nipals, biplot, plotIndiv, plotVar, plot3dIndiv, plot3dVar and http://www.math. univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(liver.toxicity)
tune <- tune.pca(liver.toxicity$gene, center = TRUE, scale = TRUE)
```

---

tune.rcc *Estimate the parameters of regularization for Regularized CCA*

Description

Computes leave-one-out or M-fold cross-validation scores on a two-dimensional grid to determine optimal values for the parameters of regularization in rcc.

Usage

```
tune.rcc(X, Y, grid1 = seq(0.001, 1, length = 5),
            grid2 = seq(0.001, 1, length = 5),
            validation = c("loo", "Mfold"), folds,
            M = 10, plt = TRUE)
```

Arguments

X	numeric matrix or data frame $(n \times p)$, the observations on the X variables. NAs are allowed.
Y	numeric matrix or data frame $(n \times q)$, the observations on the Y variables. NAs are allowed.

grid1, grid2	vector numeric defining the values of lambda1 and lambda2 at which cross-validation score should be computed. Defaults to grid1=grid2=seq(0.001, 1, length=5).
validation	character string. What kind of (internal) cross-validation method to use, (partially) matching one of "loo" (leave-one-out) or "Mfolds" (M-folds). See Details.
folds	list of vectors (as returned by split) containing the indices for the validation sample (see Details).
M	positive integer. Number of folds to use if validation="Mfold". Defaults to M=10.
plt	logical argument indicating whether a image map should be plotted by calling the imgCV function.

Details

If validation="Mfolds", M-fold cross-validation is performed by calling Mfold. When folds is given, the elements of folds should be integer vectors specifying the indices of the validation sample and the argument M is ignored. Otherwise, the folds are generated. The number of cross-validation folds is specified with the argument M.

If validation="loo", leave-one-out cross-validation is performed by calling the loo function. In this case the arguments folds and M are ignored.

The estimation of the missing values can be performed by the reconstitution of the data matrix using the nipals function. Otherwise, missing values are handled by casewise deletion in the rcc function.

Value

The returned value is a list with components:

opt.lambda1,	
opt.lambda2	value of the parameters of regularization on which the cross-validation method reached it optimal.
opt.score	the optimal cross-validation score reached on the grid.
grid1, grid2	original vectors grid1 and grid2.
mat	matrix containing the cross-validation score computed on the grid.

Author(s)

Sébastien Déjean and Ignacio González.

See Also

loo, Mfold, image.tune.rcc and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(nutrimouse)
X <- nutrimouse$lipid
Y <- nutrimouse$gene

## this can take some seconds
## Not run:
tune.rcc(X, Y, validation = "Mfold")

## End(Not run)
```

---

vac18 *Vaccine study Data*

Description

The data come from a trial evaluating a vaccine based on HIV-1 lipopeptides in HIV-negative volunteers. The vaccine (HIV-1 LIPO-5 ANRS vaccine) contains five HIV-1 amino acid sequences coding for Gag, Pol and Nef proteins. This data set contains the expression measure of a subset of 2500 genes from purified in vitro stimulated Peripheral Blood Mononuclear Cells from 42 repeated samples (12 unique vacicnated participants) 14 weeks after vaccination, , 6 hours after in vitro stimulation by either (1) all the peptides included in the vaccine (LIPO-5), or (2) the Gag peptides included in the vaccine (GAG+) or (3) the Gag peptides not included in the vaccine (GAG-) or (4) without any stimulation (NS).

Usage

```
data(vac18)
```

Format

A list containing the following components:

`gene` data frame with 42 rows and 2500 columns. The expression measure of 2500 genes for the 42 samples (PBMC cells from 12 unique subjects).

`stimulation` is a fctor of 42 elements indicating the type of in vitro simulation for each sample.

`sample` is a vector of 42 elements indicating the unique subjects (for example the value '1' correspond to the first patient PBMC cells). Note that the design of this study is unbalanced.

`tab.prob.gene` is a data frame with 2500 rows and 2 columns, indicating the Illumina probe ID and the gene name of the annotated genes.

Details

This is a subset of the original study for illustrative purposes.

References

Salmon-Ceron D, Durier C, Desaint C, Cuzin L, Surenaud M, Hamouda N, Lelievre J, Bonnet B, Pialoux G, Poizot-Martin I, Aboulker J, Levy Y, Launay O, trial group AV: Immunogenicity and safety of an HIV-1 lipopeptide vaccine in healthy adults: a phase 2 placebo-controlled ANRS trial. AIDS 2010, 24(14):2211-2223.

`valid` *Compute validation criterion for PLS, sPLS, PLS-DA and sPLS-DA*

Description

The valid function has been supersided by the perf function to evaluate the performance of the PLS, sPLS, PLS-DA and sPLS-DA methods. See ?perf for more details

Usage

```
valid(object, ...)
```

Arguments

`object`	object of class inheriting from `"pls"`, `"plsda"`, `"spls"` or `"splsda"`.
`...`	arguments to pass to `nearZeroVar`.

Author(s)

Sebastien Dejean, Ignacio Gonzalez and Kim-Anh Le Cao.

References

Tenenhaus, M. (1998). *La r?gression PLS: th?orie et pratique*. Paris: Editions Technic.

L? Cao, K. A., Rossouw D., Robert-Grani?, C. and Besse, P. (2008). A sparse PLS for variable selection when integrating Omics data. *Statistical Applications in Genetics and Molecular Biology* **7**, article 35.

Mevik, B.-H., Cederkvist, H. R. (2004). Mean Squared Error of Prediction (MSEP) Estimates for Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR). *Journal of Chemometrics* **18**(9), 422-429.

See Also

`perf`, `predict` and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

`vip` *Variable Importance in the Projection (VIP)*

Description

The function `vip` computes the influence on the Y-responses of every predictor X in the model.

Usage

```
vip(object)
```

Arguments

`object`	object of class inheriting from `"pls"` or `"spls"`.

Details

Variable importance in projection (VIP) coefficients reflect the relative importance of each X variable for each X variate in the prediction model. VIP coefficients thus represent the importance of each X variable in fitting both the X- and Y-variates, since the Y-variates are predicted from the X-variates.

VIP allows to classify the X-variables according to their explanatory power of Y. Predictors with large VIP, larger than 1, are the most relevant for explaining Y.

Value

`vip` produces a matrix of VIP coefficients for each X variable (rows) on each variate component (columns).

Author(s)

Sébastien Déjean and Ignacio González.

References

Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*. Paris: Editions Technic.

See Also

`pls`, `spls`, `summary`.

Examples

```
data(linnerud)
X <- linnerud$exercise
Y <- linnerud$physiological
linn.pls <- pls(X, Y)

linn.vip <- vip(linn.pls)

barplot(linn.vip,
        beside = TRUE, col = c("lightblue", "mistyrose", "lightcyan"),
        ylim = c(0, 1.7), legend = rownames(linn.vip),
        main = "Variable Importance in the Projection", font.main = 4)
```

`wrapper.rgcca` *mixOmics wrapper for Regularised Generalised Canonical Correlation Analysis (rgcca)*

Description

Wrapper function to perform Regularized Generalised Canonical Correlation Analysis (rGCCA), a generalised approach for the integration of multiple datasets. For more details, see the `help(rgcca)` from the **RGCCA** package.

Usage

```
wrapper.rgcca(data, design = 1 - diag(length(data)), tau = rep(1, length(data)),
  ncomp = rep(1, length(data)), scheme = "centroid", scale = TRUE, init = "svd",
  bias = TRUE, tol = .Machine$double.eps, verbose = FALSE)
```

Arguments

`data`	a list of data sets (called 'blocks') matching on the same samples. Data in the list should be arranged in samples x variables. `NA`s are not allowed.
`design`	numeric matrix of size (number of blocks) x (number of blocks) with only 0 or 1 values. A value of 1 (0) indicates a relationship (no relationship) between the blocks to be modelled using sGCCA.
`tau`	numeric vector of length the number of blocks in `data`. Each regularization parameter will be applied on each block and takes the value between 0 (no regularisation) and 1. If tau = "optimal" the shrinkage paramaters are estimated for each block and each dimension using the Schafer and Strimmer (2005) analytical formula.
`ncomp`	numeric vector of length the number of blocks in `data`. The number of components to include in the model for each block (does not necessarily takes the same value for each block).
`scheme`	Either "horst", "factorial" or "centroid" (Default: "centroid").

scale	If scale = TRUE, each block is standardized to zero means and unit variances (default: TRUE)
init	Mode of initialization use in the SGCCA algorithm, either by Singular Value Decompostion ("svd") or random ("random") (default : "svd").
bias	A logical value for biaised or unbiaised estimator of the var/cov (defaults to TRUE).
tol	Convergence stopping value.
verbose	if set to TRUE, reports progress on computing.

Details

This wrapper function performs rGCCA (see **RGCCA**) with $1, \ldots,$ncomp components on each block data set. A supervised or unsupervised model can be run. For a supervised model, the unmap function should be used as an input data set. More details can be found on the package **RGCCA**.

Value

wrapper.rgcca returns an object of class "rgcca", a list that contains the following components:

data	the input data set (as a list).
design	the input design.
variates	the sgcca components.
loadings	the loadings for each block data set (outer wieght vector).
loadings.star	the laodings, standardised.
tau	the input tau parameter.
scheme	the input schme.
ncomp	the number of components on each block.
crit	the convergence criterion.
AVE	Indicators of model quality based on the Average Variance Explained (AVE): AVE(for one block), AVE(outer model), AVE(inner model)..
names	list containing the names to be used for individuals and variables.

More details can be found in the references.

Author(s)

Arthur Tenenhaus, Vincent Guillemot and Kim-Anh Lê Cao.

References

Tenenhaus A. and Tenenhaus M., (2011), Regularized Generalized Canonical Correlation Analysis, Psychometrika, Vol. 76, Nr 2, pp 257-284.

Schafer J. and Strimmer K., (2005), A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statist. Appl. Genet. Mol. Biol. 4:32.

See Also

rgcca, plotIndiv, plotVar, wrapper.sgcca. sgcca and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(nutrimouse)
# need to unmap the Y factor diet
Y = unmap(nutrimouse$diet)
data = list(nutrimouse$gene, nutrimouse$lipid,Y)
# with this design, gene expression and lipids are connected to the diet factor
# design = matrix(c(0,0,1,
#                   0,0,1,
#                   1,1,0), ncol = 3, nrow = 3, byrow = TRUE)

# with this design, gene expression and lipids are connected to the diet factor
# and gene expression and lipids are also connected
design = matrix(c(0,1,1,
                  1,0,1,
                  1,1,0), ncol = 3, nrow = 3, byrow = TRUE)
#note: the tau parameter is the regularization parameter
wrap.result.rgcca = wrapper.rgcca(data = data, design = design, tau = c(1, 1, 0),
                                  ncomp = c(2, 2, 1),
                                  scheme = "centroid", verbose = FALSE)
#wrap.result.rgcca
```

---

wrapper.sgcca *mixOmics wrapper for Sparse Generalised Canonical Correlation Analysis (sgcca)*

Description

Wrapper function to perform Sparse Generalised Canonical Correlation Analysis (sGCCA), a generalised approach for the integration of multiple datasets. For more details, see the help(sgcca) from the **RGCCA** package.

Usage

```
wrapper.sgcca(data, design = 1 - diag(length(data)), penalty = rep(1, length(data)),
  ncomp = rep(1, length(data)), scheme = "centroid", scale = TRUE, init = "svd",
  bias = TRUE, tol = .Machine$double.eps, verbose = FALSE)
```

Arguments

data	a list of data sets (called 'blocks') matching on the same samples. Data in the list should be arranged in samples x variables. NAs are not allowed.

design	numeric matrix of size (number of blocks) x (number of blocks) with only 0 or 1 values. A value of 1 (0) indicates a relationship (no relationship) between the blocks to be modelled using sGCCA.
penalty	numeric vector of length the number of blocks in data. Each penalty parameter will be applied on each block and takes the value between 0 (no variable selected) and 1 (all variables included).
ncomp	numeric vector of length the number of blocks in data. The number of components to include in the model for each block (does not necessarily takes the same value for each block).
scheme	Either "horst", "factorial" or "centroid" (Default: "centroid").
scale	If scale = TRUE, each block is standardized to zero means and unit variances (default: TRUE)
init	Mode of initialization use in the SGCCA algorithm, either by Singular Value Decompostion ("svd") or random ("random") (default : "svd").
bias	A logical value for biaised or unbiaised estimator of the var/cov (defaults to TRUE).
tol	Convergence stopping value.
verbose	if set to TRUE, reports progress on computing.

Details

This wrapper function performs sGCCA (see **RGCCA**) with $1, \ldots,$ncomp components on each block data set. A supervised or unsupervised model can be run. For a supervised model, the unmap function should be used as an input data set. More details can be found on the package **RGCCA**.

Value

wrapper.sgcca returns an object of class "sgcca", a list that contains the following components:

data	the input data set (as a list).
design	the input design.
variates	the sgcca components.
loadings	the loadings for each block data set (outer wieght vector).
loadings.star	the laodings, standardised.
penalty	the input penalty parameter.
scheme	the input schme.
ncomp	the number of components on each block.
crit	the convergence criterion.
AVE	Indicators of model quality based on the Average Variance Explained (AVE): AVE(for one block), AVE(outer model), AVE(inner model)..
names	list containing the names to be used for individuals and variables.

More details can be found in the references.

Author(s)

Arthur Tenenhaus, Vincent Guillemot and Kim-Anh Lê Cao.

References

Tenenhaus A. and Tenenhaus M., (2011), Regularized Generalized Canonical Correlation Analysis, Psychometrika, Vol. 76, Nr 2, pp 257-284.

Tenenhaus A., Phillipe C., Guillemot, V., Le Cao K-A., Grill J., Frouin, V. Variable Selection For Generalized Canonical Correlation Analysis. 2013. (in revision)

See Also

sgcca, plotIndiv, plotVar, wrapper.rgcca, rgcca and http://www.math.univ-toulouse.fr/~biostat/mixOmics/ for more details.

Examples

```
data(nutrimouse)
# need to unmap the Y factor diet
Y = unmap(nutrimouse$diet)
data = list(nutrimouse$gene, nutrimouse$lipid,Y)
# with this design, gene expression and lipids are connected to the diet factor
# design = matrix(c(0,0,1,
#                   0,0,1,
#                   1,1,0), ncol = 3, nrow = 3, byrow = TRUE)

# with this design, gene expression and lipids are connected to the diet factor
# and gene expression and lipids are also connected
design = matrix(c(0,1,1,
                  1,0,1,
                  1,1,0), ncol = 3, nrow = 3, byrow = TRUE)

#note: the penalty parameters will need to be tuned
wrap.result.sgcca = wrapper.sgcca(data = data, design = design, penalty = c(.3,.5, 1),
                                  ncomp = c(2, 2, 1),
                                  scheme = "centroid", verbose = FALSE)
wrap.result.sgcca
#did the algo converge?
wrap.result.sgcca$crit  # yes
```

---

yeast *Yeast metabolomic study*

Description

Two Saccharomyces Cerevisiae strains were compared under two different environmental conditions, 37 metabolites expression are measured.

Usage

```
data(yeast)
```

Format

A list containing the following components:

`data` data matrix with 55 rows and 37 columns. Each row represents an experimental sample, and each column a single metabolite.

`strain` a factor containing the type of strain (MT or WT).

`condition` a factor containing the type of environmental condition (AER or ANA).

`strain.condition` a crossed factor between `strain` and `condition`.

Details

In this study, two Saccharomyces cerevisiae strains were used - wild-type (WT) and mutant (MT), and were carried out in batch cultures under two different environmental conditions, aerobic (AER) and anaerobic (ANA) in standard mineral media with glucose as the sole carbon source. After normalization and pre processing, the metabolomic data results in 37 metabolites and 55 samples which include 13 MT-AER, 14 MT-ANA, 15 WT-AER and 13 WT-ANA samples

References

Villas-Boas S, Moxley J, Akesson M, Stephanopoulos G, Nielsen J: High-throughput metabolic state analysis (2005). The missing link in integrated functional genomics. *Biochemical Journal*, bold388:669–677.

Index