Package 'SciViews'

July 26, 2011

Type Package

Title SciViews GUI API - Main package

Imports ellipse

Depends R (>= 2.6.0), stats, grDevices, graphics, MASS

Enhances base, stats

Description Functions to install SciViews additions to R, and more (various) tools

Version 0.9-4

Date 2011-07-26

Author Philippe Grosjean

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

License GPL-2

LazyLoad yes

URL http://www.sciviews.org/SciViews-R

BugReports https://r-forge.r-project.org/tracker/?group_id=194

R topics documented:

SciViews-package *SciViews GUI API - Main package*

Description

Functions to install SciViews additions to R, and miscellaneous

Details

Package:	SciViews
Type:	Package
Version:	0.9-2
Date:	2010-09-26
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Philippe Grosjean

Maintainer: Philippe Grosjean <phgrosjean@sciviews.org>

References

SciViews: http://www.sciviews.org/

colors *Various color palettes*

Description

Create vectors of `n` contiguous colors.

Usage

```
cwm.colors(n, alpha = 1, s = 0.9, v = 0.9)
rwb.colors(n, alpha = 1, s = 0.9, v = 0.9)
ryg.colors(n, alpha = 1, s = 0.9, v = 0.9)
```

Arguments

`n`	the number of colors (>= 1) to be in the palette.
`alpha`	the alpha transparency, a number in [0, 1], see argument `alpha` in `hsv`.
`s`	the 'saturation' to be used to complete the HSV color descriptions.
`v`	the 'value' to use for the HSV color descriptions.

Details

`cwm.colors(s = 0.5, v = 1)` gives very similar colors to `cm.colors()`.

`ryg.colors()` is similar to `rainbow(start = 0, end = 2/6)`.

Value

A character vector, `cv` of color names. This can be used for user-defined color palette, using `palette(cv)`, or a `col = cv` specification in a graphic function or in `par`.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

`cm.colors`, `colorRampPalette`

Examples

```
## Draw color wheels with various palettes
opar <- par(mfrow = c(2, 2))
pie(rep(1, 11), col = cwm.colors(11), main = "Cyan - white - magenta")
pie(rep(1, 11), col = rwb.colors(11), main = "Red - white - blue")
pie(rep(1, 11), col = ryg.colors(11), main = "Red - yellow - green (1)")
pie(rep(1, 11), col = ryg.colors(11, s = 0.5, v = 1), main = "Red - yellow - green (2)")
par(opar)
```

---

correlation *Correlation matrices*

Description

Compute the correlation matrix between two variables, or more (between all columns of a matrix or data frame).

Usage

```
correlation(x, ...)
## S3 method for class 'formula'
correlation(formula, data = NULL, subset, na.action, ...)
## Default S3 method:
correlation(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"), ...)

is.correlation(x)
as.correlation(x)

## S3 method for class 'correlation'
print(x, digits = 3, cutoff = 0, ...)
## S3 method for class 'correlation'
summary(object, cutpoints = c(0.3, 0.6, 0.8, 0.9, 0.95),
    symbols = c(" ", ".", ",", "+", "*", "B"), ...)
```

```
## S3 method for class 'summary.correlation'
print(x, ...)
## S3 method for class 'correlation'
plot(x, y = NULL, outline = TRUE,
    cutpoints = c(0.3, 0.6, 0.8, 0.9, 0.95), palette = rwb.colors, col = NULL,
    numbers = TRUE, digits = 2, type = c("full", "lower", "upper"),
    diag = (type == "full"), cex.lab = par("cex.lab"), cex = 0.75 * par("cex"),
    ...)
```

Arguments

x	a numeric vector, matrix or data frame (or any object for `is.correlation()`, or `as.correlation()`).
formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
subset	an optional vector used to select rows (observations) of the data matrix `x`.
na.action	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`.
method	a character string indicating which correlation coefficient is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`, can be abbreviated.
y	`NULL` (default), or a vector, matrix or data frame with compatible dimensions to `x` for `correlation()`. The default is equivalent to `x = y`, but more efficient. For `plot.correlation()`, if a second 'correlation' object is provided in `y`, then a visual comparison of two correlation matrices is performed (not implemented yet)!
use	an optional character string giving a method for computing correlations in the presence of missing values. This must be (an abbreviation of) one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"`.
digits	digits to print after the decimal separator.
cutoff	correlation coefficients lower than this (in absolute value) are suppressed.
object	a 'correlation' object.
cutpoints	the cut points to use for categories. Specify only positive values (absolute value of correlation coefficients are summarized, or negative equivalents are automatically computed for the graph. Do not include 0 or 1 in the cutpoints).
symbols	the symbols to use to summarize the correlation matrix.
outline	do we draw the outline of the ellipse?
palette	a function that can produce a palette of colors.
col	color of the ellipse. If `NULL` (default), the colors will be computed using `cutpoints` and `palette`.
numbers	do we print correlation values in the center of the ellipses?
type	do we plot a complete matrix, or only lower or upper triangle?
diag	do we plot items on the diagonal? They have always a correlation of one.
cex.lab	the expansion factor for labels.
cex	the expansion factor for text.
...	further arguments passed to functions.

Value

`correlation()` and `as.correlation()` create a 'correlation' object, while `is.correlation()` tests for it.

There are `print()` and `summary()` methods for the 'correlation' object that differ in the symbolic encoding of the correlations in `summary()`, using `symnum`, which makes large correlation matrices more readable.

The method `plot` returns nothing, but it draws ellipses on a graph that represent the correlation matrix visually. This is essentially the `plotcorr()` function from package ellipse, with slightly different default arguments and with default `cutpoints` equivalent to those used in the `summary` method.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>, wrapping code in package ellipse, function `plotcorr()` for the `plot.correlation()` method.

See Also

`cov`, `cov2cor`, `cov.wt`, `symnum`, `plotcorr` and look also at `panel.cor`

Examples

```
## This is a simple correlation coefficient
cor(rnorm(10), runif(10))
## but this is a 'correlation' object containing a correlation matrix
correlation(rnorm(10), runif(10))

## 'correlation' objects allow better inspection of the correlation matrices
## than the output of default R cor() function
(longley.cor <- correlation(longley))
summary(longley.cor) # Synthetic view of the correlation matrix
plot(longley.cor)    # Graphical representation

## Use of the formula interface
(mtcars.cor <- correlation(~ mpg + cyl + disp + hp, data = mtcars,
    method = "spearman", na.action = "na.omit"))

mtcars.cor2 <- correlation(mtcars, method = "spearman")
print(mtcars.cor2, cutoff = 0.6)
summary(mtcars.cor2)
plot(mtcars.cor2, type = "lower")

mtcars.cor2["mpg", "cyl"] # Extract one correlation from the correlation matrix
## TODO: a plot comparing two correlation matrices
```

`ln` *Logarithms*

Description

To avoid confusion using the default `log()` function, which is natural logarithm, but spells out like base 10 logarithm in the mind of some beginneRs, we define `ln()` and `ln1p()` as wrappers for `log()` with default `base = exp(1)` argument and for `log1p()`, respectively. For similar reasons, `lg()` is a wrapper of `log10()` (there is no possible confusion here, but 'lg' is another common notation for base 10 logarithm). `lg1p()` is a convenient way to use the optimized code to calculate the logarithm of x + 1, but returning the result in base 10 logarithm. `e` is the euler constant and is provided for convenience as `exp(1)`. Finally `lb()` is a synonym of `log2()`.

Usage

```
ln(x)
ln1p(x)
lg(x)
lg1p(x)
e
lb(x)
```

Arguments

`x`	a numeric or complex vector.

Value

A vector of the same length as `x` containing the transformed values. `ln(0)` gives `-Inf`, and negative values give `NaN`.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>, but these are just convenient wrappers around standard R logarithm functions in the base package.

See Also

`log`

Examples

```
ln(exp(3))              # Same as log(exp(3))
ln1p(c(0, 1, 10, 100))  # Wrapper for log1p()
lg(10^3)                # Same as log10(10^3)
lg1p(c(0, 1, 10, 100))  # log10(x + 1), but optimized for x << 1
e^4                     # Similar to exp(4), but different calculation!
## Note: exp(4) is to be preferred to e^4, if possible!
lb(1:3)                 # Wrapper for log2()
```

panels *More panel plots*

Description

Several panel plots that can be used with functions like coplot and pairs.

Usage

```
panel.reg(x, y, col = par("col"), bg = par("bg"), pch = par("pch"),
    cex = par("cex"), lwd = par("lwd"), line.reg = lm, line.col = "red",
    line.lwd = lwd, untf = TRUE, ...)
panel.ellipse(x, y, col = par("col"), bg = par("bg"), pch = par("pch"),
    cex = par("cex"), el.level = 0.7, el.col = "cornsilk", el.border = "red",
    major = TRUE, ...)
panel.cor(x, y, use = "everything", method = c("pearson", "kendall", "spearman")
    alternative = c("two.sided", "less", "greater"), digits = 2, prefix = "",
    cex = par("cex"), cor.cex = cex, stars.col = "red", ...)
```

Arguments

x	a numeric vector.
y	a numeric vector of same length as x
col	the color of the points.
bg	the background color for symbol used for the points.
pch	the symbol used for the points.
cex	the expansion factor used for the points.
lwd	the line width.
line.reg	a function that calculates coefficients of a straight line, for instance, lm, or rlm for robust linear regression.
line.col	the color of the line.
line.lwd	the width of the line.
untf	logical asking whether to untransform the straight line in case one or both axis are in log scale.
el.level	the confidence level for the bivariate normal ellipse around data; the default value of 0.7 draws an ellipse of roughly +/-1 sd.
el.col	the color used to fill the ellipse.
el.border	the color used to draw the border of the ellipse and the standardized major axis.
major	if TRUE, the standardized major axis is also drawn.
use	one of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (can be abbreviated). Defines how the cor() function behaves with missing observations.
method	one of the three correlation coefficients "pearson", (default), "kendall", or "spearman" (can be abbreviated).
alternative	the alternative hypothesis in correlation test, see cor.test.

digits	the number of decimal digits to print when the correlation coefficient is printed in the graph.
prefix	a prefix (character string) to use before the correlation coefficient printed in the graph.
cor.cex	expansion coefficient for text in printing correlation coefficients.
stars.col	the color used for significance stars (with: *** p < 0.001, ** p < 0.1, * p < 0.05, . p < 0.1.
...	further arguments to plot functions.

Details

Theses functions should be used outside of the diagonal in `pairs()`, or with `coplot()`, as they are bivariate plots.

Value

These functions return nothing and are used for their side effect of plotting in panels of composite plots.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>, but code inspired from `panel.smooth()` in graphics and `panel.car()` in package car.

See Also

`coplot`, `pairs`, `panel.smooth`, `lm`, `ellipse`, `cor` and `cor.test`

Examples

```
## Smooth lines in lower graphs and straight lines in upper graphs
pairs(trees, lower.panel = panel.smooth, upper.panel = panel.reg)
## Robust regression lines
require(MASS)  # For rlm()
pairs(trees, panel = panel.reg, diag.panel = panel.boxplot,
    reg.line = rlm, line.col = "blue", line.lwd = 2)
## A Double log graph
pairs(trees, lower.panel = panel.smooth, upper.panel = panel.reg, log = "xy")

## Graph suitables to explore correlations (take care that there are potentially
## many simultaneous tests done here... So, you loose much power is the whole
## analysis... use it just as an indication, nothing more!)
## Pearson's r
pairs(trees, lower.panel = panel.ellipse, upper.panel = panel.cor)
## Spearman's rho (ellipse and straight lines not suitable here!)
pairs(trees, lower.panel = panel.smooth, upper.panel = panel.cor,
    method = "spearman", span = 1)
## Several groups (visualize how bad it is to consider the whole set at once!)
pairs(iris[, -5], lower.panel = panel.smooth, upper.panel = panel.cor,
    method = "kendall", span = 1, col = c("red3", "blue3", "green3")[iris$Species])
## Now analyze correlation for one species only
pairs(iris[iris$Species == "virginica", -5], lower.panel = panel.ellipse,
    upper.panel = panel.cor)
```

```
## A coplot with custom panes
coplot(Petal.Length ~ Sepal.Length | Species, data = iris, panel = panel.ellipse)
```

`panels.diag` *More univariate panel plots*

Description

Several panel plots that can be used with function `pairs`.

Usage

```
panel.boxplot(x, col = par("col"), box.col = "cornsilk", ...)
panel.density(x, adjust = 1, rug = TRUE, col = par("col"), lwd = par("lwd"),
    line.col = col, line.lwd = lwd, ...)
panel.hist(x, breaks = "Sturges", hist.col = "cornsilk", hist.border = NULL,
    hist.density = NULL, hist.angle = 45, ...)
panel.qqnorm(x, pch = par("pch"), col = par("col"), bg = par("bg"),
    cex = par("cex"), lwd = par("lwd"), qq.pch = pch, qq.col = col, qq.bg = bg,
    qq.cex = cex, qqline.col = qq.col, qqline.lwd = lwd, ...)
```

Arguments

`x`	a numeric vector.
`col`	the color of the points.
`box.col`	the filling color of the boxplots.
`adjust`	the bandwidth adjustment factor, see `density`.
`rug`	do we add a rug representation (1-d plot) of the points too?
`lwd`	the line width.
`line.col`	the color of the line.
`line.lwd`	the width of the line.
`breaks`	the number of breaks, the name of a break algorithm, a vector of breakpoints, or any other acceptable value for `breaks` argument of `hist`
`hist.col`	the filling color for the histograms.
`hist.border`	the border color for the histograms.
`hist.density`	the density for filling lines in the histograms.
`hist.angle`	the angle for filling lines in the histograms.
`pch`	the symbol used for the points.
`bg`	the background color for symbol used for the points.
`cex`	the expansion factor used for the points.
`qq.pch`	the symbol used to plot points in the QQ-plots.
`qq.col`	the color of the symbol used to plot points in the QQ-plots.
`qq.bg`	the background color of the symbol used to plot points in the QQ-plots.
`qq.cex`	the expansion factor for points in the QQ-plots.
`qqline.col`	the color for the QQ-plot lines.
`qqline.lwd`	the width for the QQ-plot lines.
`...`	further arguments to plot functions, or functions that construct items, like `density()`, depending on the context.

Details

Panel functions `panel.boxplot()`, `panel.density()`, `panel.hist()` and `panel.qqnorm()` should be used only to plot univariate data on the diagonals of pair plots (or scatterplot matrix).

Value

These functions return nothing and are used for their side effect of plotting in panels of composite plots.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>, but code inspired from `spm()` in package car.

See Also

`pairs`, `boxplot`, `hist`, `density`, `qqnorm`

Examples

```
## Example of scatterplot matrices with custom plots on the diagonal
## Boxplots
pairs(trees, panel = panel.smooth, diag.panel = panel.boxplot)
pairs(trees, diag.panel = panel.boxplot, box.col = "gray")
## Densities
pairs(trees, panel = panel.smooth, diag.panel = panel.density)
pairs(trees, diag.panel = panel.density, line.col = "red", adjust = 0.5)
## Histograms
pairs(trees, panel = panel.smooth, diag.panel = panel.hist)
pairs(trees, diag.panel = panel.hist, hist.col = "gray", breaks = "Scott")
## QQ-plots against Normal theoretical distribution
pairs(trees, panel = panel.smooth, diag.panel = panel.qqnorm)
pairs(trees, diag.panel = panel.qqnorm, qqline.col = 2, qq.cex = .5, qq.pch = 3)
```

`pcomp` *Principal Components Analysis*

Description

Perform a principal components analysis on a matrix or data frame and return a `pcomp` object.

Usage

```
pcomp(x, ...)
## S3 method for class 'formula'
pcomp(formula, data = NULL, subset, na.action,
    method = c("svd", "eigen"), ...)
## Default S3 method:
pcomp(x, method = c("svd", "eigen"), scores = TRUE,
    center = TRUE, scale = TRUE, tol = NULL, covmat = NULL,
subset = rep(TRUE, nrow(as.matrix(x))), ...)

## S3 method for class 'pcomp'
```

```
print(x, ...)
## S3 method for class 'pcomp'
summary(object, loadings = TRUE, cutoff = 0.1, ...)
## S3 method for class 'summary.pcomp'
print(x, digits = 3, loadings = x$print.loadings,
    cutoff = x$cutoff, ...)

## S3 method for class 'pcomp'
plot(x, which = c("screeplot", "loadings", "correlations", "scores"),
    choices = 1L:2L, col = par("col"), bar.col = "gray", circle.col = "gray",
    ar.length = 0.1, pos = NULL, labels = NULL, cex = par("cex"),
    main = paste(deparse(substitute(x)), which, sep = " - "), xlab, ylab, ...)
## S3 method for class 'pcomp'
screeplot(x, npcs = min(10, length(x$sdev)), type = c("barplot", "lines"),
    col = "cornsilk", main = deparse(substitute(x)), ...)
## S3 method for class 'pcomp'
points(x, choices = 1L:2L, type = "p", pch = par("pch"),
    col = par("col"), bg = par("bg"), cex = par("cex"), ...)
## S3 method for class 'pcomp'
lines(x, choices = 1L:2L, groups, type = c("p", "e"),
    col = par("col"), border = par("fg"), level = 0.9, ...)
## S3 method for class 'pcomp'
text(x, choices = 1L:2L, labels = NULL, col = par("col"),
    cex = par("cex"), pos = NULL, ...)
## S3 method for class 'pcomp'
biplot(x, choices = 1L:2L, scale = 1, pc.biplot = FALSE, ...)

## S3 method for class 'pcomp'
pairs(x, choices = 1L:3L, type = c("loadings", "correlations"),
    col = par("col"), circle.col = "gray", ar.col = par("col"), ar.length = 0.05
    pos = NULL, ar.cex = par("cex"), cex = par("cex"), ...)

## S3 method for class 'pcomp'
predict(object, newdata, dim = length(object$sdev), ...)
## S3 method for class 'pcomp'
correlation(x, newvars, dim = length(x$sdev), ...)
scores(x, ...)
## S3 method for class 'pcomp'
scores(x, labels = NULL, dim = length(x$sdev), ...)
```

Arguments

x	a matrix or data frame with numeric data.
formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit.
method	either "svd" (the function uses prcomp), or "eigen" (the function uses princomp), or an abbreviation.

...	arguments passed to or from other methods. If x is a formula one might specify scale, tol or covmat.
scores	a logical value indicating whether the score on each principal component should be calculated.
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. Note that this argument is ignored for method = "eigen" and the dataset is always centered in this case.
scale	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is TRUE, which in general, is advisable. Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
tol	only when method = "svd". A value indicating the magnitude below which components should be omitted. (Components are omitted if their standard deviations are less than or equal to tol times the standard deviation of the first component.) With the default null setting, no components are omitted. Other settings for tol could be tol = 0 or tol = sqrt(.Machine$double.eps), which would omit essentially constant components.
covmat	a covariance matrix, or a covariance list as returned by cov.wt (and cov.mve or cov.mcd from package MASS). If supplied, this is used rather than the covariance matrix of x.
object	a 'pcomp' object.
loadings	do we also summarize the loadings?
cutoff	the cutoff value below which loadings are replaced by white spaces in the table. That way, larger values are easier to spot and to read in large tables.
digits	the number of digits to print.
which	the graph to plot.
choices	which principal axes to plot. For 2D graphs, specify two integers.
col	the color to use in graphs.
bar.col	the color of bars in the screeplot.
circle.col	the color for the circle in the loadings or correlations plots.
ar.length	the length of the arrows in the loadings and correlations plots.
pos	the position of text relative to arrows in loadings and correlations plots.
labels	the labels to write. If NULL default values are computed.
cex	the factor of expansion for text (labels) in the graphs.
main	the title of the graph.
xlab	the label of X-axis.
ylab	the label of Y-axis.
pch	type of symbol to use.
bg	background color for symbols.
groups	a grouping factor.
border	the color of the border.
level	the probability level to use to draw the ellipse.
pc.biplot	do we create a Gabriel's biplot (see biplot() documentation)?

npcs	the number of principal components to represent in the screeplot.
type	the type of screeplot ("barplot" or "lines") or pairs plot ("loadings" or "correlations").
ar.col	color of arrows.
ar.cex	expansion factor for terxt on arrows.
newdata	new individuals with observations for the same variables as those used for making the PCA. You can then plot these additional individuals in the scores graph.
newvars	new variables with observations for same individuals as those used for making the PCA. Correlation with PCs is calculated. You can then plot these additional variables in the correlation graph.
dim	The number of principal components to keep.

Details

pcomp() is a generic function with "formula" and "default" methods. It is essentially a wrapper around prcomp() and princomp() to provide a coherent interface and object for both methods.

A 'pcomp' object is created. It inherits from 'pca' (as in labdsv package, but not compatible with the 'pca' object of package ade4!) and of 'princomp'.

For more information on calculation done, refer to prcomp for method = "svd" or princomp for method = "eigen".

Value

A c("pcomp", "pca", "princomp") object containing list components:

comp_i	Description of comp_i.

TODO: complete this (also speak about the various methods)!

Note

The signs of the columns of the loadings and scores are arbitrary, and so may differ between different programs for PCA, and even between different builds of R.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>, but the core code is indeed in package stats.

See Also

vectorplot, prcomp, princomp, loadings, link{correlation}

Examples

```
## We will analyze mtcars without the Mercedes data (rows 8:14)
data(mtcars)
cars.pca <- pcomp(~mpg+cyl+disp+hp+drat+wt+qsec, data = mtcars, subset = -(8:14))
cars.pca
summary(cars.pca)
screeplot(cars.pca)

## Loadings are extracted and plotted like this
```

```
(cars.ldg <- loadings(cars.pca))
plot(cars.pca, which = "loadings") # Equivalent to vectorplot(cars.ldg)

## Similarly, correlations of variables with PCs are extracted and plotted
(cars.cor <- correlation(cars.pca))
plot(cars.pca, which = "correlations") # Equivalent to vectorplot(cars.cor)
## One can add supplementary variables on this graph
lines(correlation(cars.pca,
    newvars = mtcars[-(8:14), c("vs", "am", "gear", "carb")]))

## Plot the scores
plot(cars.pca, which = "scores", cex = 0.8) # Similar to plot(scores(x)[, 1:2])
## Add supplementary individuals to this plot (labels), use also points() or lines()
text(predict(cars.pca, newdata = mtcars[8:14, ]), col = "gray", cex = 0.8)

## More scores plot
## TODO...

## Pairs plot for 3 PCs
iris.pca <- pcomp(iris[, -5])
pairs(iris.pca, col = (2:4)[iris$Species])

## rgl plot for 3 PCs
## TODO...
```

`snippets` *SciViews snippet help*

Description

We are now *1311664818.03334*

Arguments

The content of the arguments section...

`vectorplot` *Plot vectors inside a unit circle (PCA loadings or correlations plots)*

Description

Plots vectors with 0<norms<1 inside a circle. These plots are mainly designed to represent variables in principal components space for PCAs.

Usage

```
vectorplot(x, ...)
## Default S3 method:
vectorplot(x, y, col = par("col"), circle.col = "gray",
    ar.length = 0.1, pos = NULL, cex = par("cex"), labels = NULL, ...)
## S3 method for class 'loadings'
```

```
vectorplot(x, choices = 1L:2L, col = par("col"),
    circle.col = "gray", ar.length = 0.1, pos = NULL, cex = par("cex"),
    labels = rownames(x), main = deparse(substitute(x)), ...)
## S3 method for class 'correlation'
vectorplot(x, choices = 1L:2L, col = par("col"),
    circle.col = "gray", ar.length = 0.1, pos = NULL, cex = par("cex"),
    labels = rownames(x), main = deparse(substitute(x)), ...)
```

Arguments

x	an object that has a `vectorplot()` method, like 'loadings' or 'correlation', or a numeric vector with 0<values<1.
y	a numeric vector with 0<values<1 of same length as `x`.
choices	a vector of two integers indicating the axes to plot.
col	color of the arrows and labels.
circle.col	the color for the circle around the vector plot.
ar.length	the length of the arrows.
pos	the position of text relative to arrows. If `NULL`, a suitable position is calculated according to the direction where the arrows are pointing.
cex	the factor of expansion for labels in the graph.
labels	the labels to write.
main	the title of the graph.
...	further arguments passed to plot functions.

Value

The object 'x' is returned invisibly. These functions are called for their side-effect of drawing a vector plot.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>.

See Also

pcomp, loadings, correlation

Examples

```
## Create a PCA and plot loadings and correlations
iris.pca <- pcomp(iris[, -5])
vectorplot(loadings(iris.pca))
vectorplot(correlation(iris.pca))
## Note: on screen devices, change aspect ratio of the graph by resizing
## the window to reveal cropped labels...
```

Index