Package ‘SciViews’

October 12, 2022

Type Package
Version 0.9-13.1
Date 2018-01-05
Title SciViews - Main package
Description Functions to install SciViews additions to R, and more tools.
Maintainer Philippe Grosjean <phgrosjean@sciviews.org>
Depends R (>= 2.6.0)
Imports ellipse, grDevices, graphics, stats
Suggests MASS, covr, knitr, testthat
Enhances base
ByteCompile yes
License GPL-2
URL https://github.com/SciViews/SciViews,
    http://www.sciviews.org/SciViews-R
RoxygenNote 6.0.1
NeedsCompilation no
Author Philippe Grosjean [aut, cre]
Repository CRAN
Date/Publication 2019-11-16 20:17:58 UTC

R topics documented:

SciViews-package .................................................. 2
colors ............................................................. 2
correlation ....................................................... 3
enum ............................................................... 6
ln ................................................................. 7
colors

nr ............................................................ 8
panels ......................................................... 9
panels.diag .................................................. 11
pcomp ......................................................... 13
timing ........................................................ 18
vectorplot .................................................... 18

Index 21

SciViews-package  SciViews - Main package

Description

The SciViews package provides various functions to install the SciViews::R dialect. It also provides additional utilites besides base, recommended and tidyverse.

Important functions

TODO...

colors Various color palettes.

Description

Create vectors of n contiguous colors.

Usage

rwb_colors(n, alpha = 1, s = 0.9, v = 0.9)
rwb.colors(n, alpha = 1, s = 0.9, v = 0.9)
rgw_colors(n, alpha = 1, s = 0.9, v = 0.9)
rgw.colors(n, alpha = 1, s = 0.9, v = 0.9)
ryg_colors(n, alpha = 1, s = 0.9, v = 0.9)
ryg.colors(n, alpha = 1, s = 0.9, v = 0.9)
cwm_colors(n, alpha = 1, s = 0.9, v = 0.9)
cwm.colors(n, alpha = 1, s = 0.9, v = 0.9)
correlation

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()].
[0, 1]: R:0, [hsv()]: R: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(). rgy_colors() is similar to rainbow(start = 0, end = 2/6). The xxx_colors() (tidyverse name-compatible) and ‘xxx.colors()“ (grDevices name-compatible) functions are synonyms.

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 = rwg.colors(11), main = "Red - white - green")
pie(rep(1, 11), col = rgy.colors(11), main = "Red - yellow - green")
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, ...)

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)

is.correlation(x)

as.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"), ...)

## S3 method for class 'Correlation'
lines(x, choices = 1L:2L, col = par("col"), lty = 2,
       ar.length = 0.1, pos = NULL, cex = par("cex"), labels = rownames(x),
       ...)

Arguments

x A numeric vector, matrix or data frame (or any object for is.Correlation(),
as.Correlation()).

... Further arguments passed to functions.

formula A formula with no response variable, referring only to numeric variables.

data An optional data frame (or similar: see model.frame()) containing the vari-
        ables 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 na.fail() is used if that is not set. The 'factory-fresh' default is na.omit().

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.

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".
**method**

A character string indicating which correlation coefficient is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.

**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.

**choices**

The items to select

**lty**

The line type to draw.

**ar.length**

The length of the arrow head.

**pos**

The position relative to arrows.

**labels**

The label to draw near arrows.

**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 at `panel_cor()`
Examples

# This is a simple correlation coefficient
cor(rnorm(10), runif(10))
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 a correlation from the correlation matrix

---

tenum

enum

Enumerate items in an object.

description

enum() is creating a vector of integers to enumerate items in an object. It is particularly useful in the for(i in enum(object)) construct.

usage

enum(x)

arguments

x Any object.

note

The pattern for(i in 1:length(object)) is often found, but it fails in case length(object) == 0! enum() is indeed a synonym of seq_along(), but the later one is less expressive in the context.

see also

seq_along()
Examples

```r
enum(letters)
enum(numeric(0))
# Compare with:
1:length(numeric(0))
enum(NULL)
```

```r
letters5 <- letters[1:5]
for (i in enum(letters5)) cat("letter", i, ", ", letters5[i], "\n")
```

---

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

```r
ln(x)
ln1p()
lg()
lg1p(x)
E
lb()
```

Arguments

- **x**: A numeric or complex vector.

Format

An object of class `numeric` of length 1.

See Also

- `log()`
Examples

\begin{verbatim}
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!
lb(1:3)         # Wrapper for log2()
\end{verbatim}

nr

Convenience functions for rows or columns manipulations.

Description

nr() and nc() are synonyms of the ugly NROW() or NCOL() that still provide a result, even if dim attribute of the object is not set, on the contrary to nrow() or ncol(). ROWS and COLS are constants that makes call to apply() more expressive.

Usage

nr(x)
nr(x)

nc(x)
nc(x)

ROWS
COLS

Arguments

x Any object.

Format

An object of class numeric of length 1.

See Also

NROW()

Examples

```
mm <- matrix(1:6, nrow = 3)
nr(mm)
nr(mm)
nc(mm)
nc(mm)
vv <- 1:6
nr(vv)
nr(vv)
# ROWS and COLS constants used with apply()
apply(mm, ROWS, mean) # Idem apply(mm, 1, mean)
apply(mm, COLS, mean) # Idem apply(mm, 2, mean)
```
Description

Several panel plots that can be used with functions like `coplot()` and `[pairs)](R:pairs).

Usage

```r
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, ...)
```

```r
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, ...)
```

```r
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, ...)
```

```r
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, ...)
```

```r
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", ...)
```

```r
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", ...)
```

```r
panel_smooth(x, y, col = par("col"), bg = NA, pch = par("pch"), cex = 1,
  col.smooth = "red", span = 2/3, iter = 3, ...)
```

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
line.reg
line.col
line.lwd
untf
...
el.level
el.col
el.border
major
use
method
alternative
digits
prefix
cor.cex
stars.col
col.smooth
span
iter

Details

These 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
library(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 suitable to explore correlations (take care there are potentially many simultaneous tests done here... So, you lose much power in the whole analysis... Use it just as an indication!)
# 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 pairs().

Usage

panel_boxplot(x, col = par("col"), box.col = "cornsilk", ...)

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, ...)
```r
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.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, qq.line.col = qq.col, qq.line.lwd = lwd, ...)

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, qq.line.col = qq.col, qq.line.lwd = lwd, ...)
```

**Arguments**

- `x`: A numeric vector.
- `col`: The color of the points.
- `box.col`: The filling color of the boxplots.
- `...`: Further arguments to plot functions, or functions that construct items, like `density()`, depending on the context.
- `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.
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

```r
# 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)
```

Description

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

Usage

```r
pcomp(x, ...)
```

## S3 method for class 'formula'
```r
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.
- **...**
  - Arguments passed to or from other methods. If `x` is a formula one might specify `scale = TRUE, tol = 0, covmat =`.
- **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 not set. The 'factory-fresh' default is `na.omit()`.
- **method**
  - Either "svd" (using `prcomp()`), "eigen" (using `princomp()`), or an abbreviation.
- **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`. 
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A ‘pcomp’ object.</td>
</tr>
<tr>
<td>loadings</td>
<td>Do we also summarize the loadings?</td>
</tr>
<tr>
<td>cutoff</td>
<td>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.</td>
</tr>
<tr>
<td>digits</td>
<td>The number of digits to print.</td>
</tr>
<tr>
<td>which</td>
<td>The graph to plot.</td>
</tr>
<tr>
<td>choices</td>
<td>Which principal axes to plot. For 2D graphs, specify two integers.</td>
</tr>
<tr>
<td>col</td>
<td>The color to use in graphs.</td>
</tr>
<tr>
<td>bar.col</td>
<td>The color of bars in the screeplot.</td>
</tr>
<tr>
<td>circle.col</td>
<td>The color for the circle in the loadings or correlations plots.</td>
</tr>
<tr>
<td>ar.length</td>
<td>The length of the arrows in the loadings and correlations plots.</td>
</tr>
<tr>
<td>pos</td>
<td>The position of text relative to arrows in loadings and correlation plots.</td>
</tr>
<tr>
<td>labels</td>
<td>The labels to write. If NULL default values are computed.</td>
</tr>
<tr>
<td>cex</td>
<td>The factor of expansion for text (labels) in the graphs.</td>
</tr>
<tr>
<td>main</td>
<td>The title of the graph.</td>
</tr>
<tr>
<td>xlab</td>
<td>The label of the x-axis.</td>
</tr>
<tr>
<td>ylab</td>
<td>The label of the y-axis.</td>
</tr>
<tr>
<td>npcs</td>
<td>The number of principal components to represent in the screeplot.</td>
</tr>
<tr>
<td>type</td>
<td>The type of screeplot (&quot;barplot&quot; or &quot;lines&quot;) or pairs plot (&quot;loadings&quot; or &quot;correlations&quot;).</td>
</tr>
<tr>
<td>pch</td>
<td>The type of symbol to use.</td>
</tr>
<tr>
<td>bg</td>
<td>The background color for symbols.</td>
</tr>
<tr>
<td>groups</td>
<td>A grouping factor.</td>
</tr>
<tr>
<td>border</td>
<td>The color of the border.</td>
</tr>
<tr>
<td>level</td>
<td>The probability level to use to draw the ellipse.</td>
</tr>
<tr>
<td>pc.biplot</td>
<td>Do we create a Gabriel’s biplot (see biplot())?</td>
</tr>
<tr>
<td>ar.col</td>
<td>Color of arrows.</td>
</tr>
<tr>
<td>ar.cex</td>
<td>Expansion factor for text on arrows.</td>
</tr>
<tr>
<td>newdata</td>
<td>New individuals with observations for the same variables as those used for calculating the PCA. You can then plot these additional individuals in the scores plot.</td>
</tr>
<tr>
<td>dim</td>
<td>The number of principal components to keep.</td>
</tr>
<tr>
<td>newvars</td>
<td>New variables with observations for same individuals as those used for calculating the PCA. Correlation with PCs is calculated. You can then plot these additional variables in the correlation plot.</td>
</tr>
</tbody>
</table>
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.

Note

The signs of the columns of the loadings and scores are arbitrary, and so may differ between functions 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()`, `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), also points() or lines()
text(predict(cars.pca, newdata = mtcars[8:14, ]), col = "gray", cex = 0.8)
# Pairs plot for 3 PCs
iris.pca <- pcomp(iris[, -5])
pairs(iris.pca, col = (2:4)[iris$Species])

timing

Timing of R expressions.

Description
Similar to system.time() but returns a more convenient 'difftime' object.

Usage

timing(expr, gc.first = TRUE)

Arguments

expr Valid R expression to be timed. If missing, proc.time() is used instead.
gc.first Logical - should a garbage collection be performed immediately before the timing? Default is TRUE.

See Also

system.time()

Examples

test <- timing(Sys.sleep(0.5))
test
attr(test, "details")

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.
...
     Further arguments passed to plot functions.
y      A numeric vector with 0 < values < 1 of same length as 'x.
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 draw near the arrows.
choices    A vector of two integers indicating the axes to plot.
main     The title of the plot.

Value

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

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

* Vector and circular plot
  vectorplot, 18
* aplot
  panels, 9
  panels.diag, 11
  vectorplot, 18
* color palettes
  colors, 2
* color
  colors, 2
* correlation matrix and plot
  correlation, 3
* datasets
  ln, 7
  nr, 8
* distribution
  correlation, 3
* logarithms and exponentials
  ln, 7
* math
  ln, 7
* models
  pcomp, 13
* panel plots
  panels, 9
  panels.diag, 11
* principal component analysis and biplot
  pcomp, 13
  *
    colors, 2
    correlation, 3
    ln, 7
    panels, 9
    panels.diag, 11
    pcomp, 13
    vectorplot, 18

  as.Correlation (correlation), 3
  as.correlation (correlation), 3
  biplot(), 16
  biplot.pcomp (pcomp), 13
  boxplot(), 13
  cm.colors(), 3
  colorRampPalette(), 3
  colors, 2
  COLS (nr), 8
  coplot(), 9–11
  cor(), 10, 11
  cor.test(), 10, 11
  Correlation (correlation), 3
  correlation, 3
  Correlation(), 17, 19
  correlation.pcomp (pcomp), 13
  cov(), 5
  cov.mcd(), 15
  cov.mve(), 15
  cov.wt(), 5, 15
  cov2cor(), 5
  cwm.colors (colors), 2
  cwm_colors (colors), 2
  density(), 12, 13
  E (ln), 7
  ellipse(), 11
  enum, 6
  hist(), 12, 13
  is.Correlation (correlation), 3
  is.correlation (correlation), 3
  lb (ln), 7
  lg (ln), 7
  lg1p (ln), 7
  lines.Correlation (correlation), 3
  lines.pcomp (pcomp), 13
  lm(), 10, 11
  ln, 7
INDEX

lnlp(ln), 7
loadings(), 17, 19
log(), 7
lowess(), 10
model.frame(), 4, 15
na.fail(), 15
na.omit(), 15
nc (nr), 8
nr, 8
NROW(), 8
options(), 15
pairs(), 10, 11, 13
pairs.pcomp (pcomp), 13
panel.boxplot (panels.diag), 11
panel.cor (panels), 9
panel.density (panels.diag), 11
panel.ellipse (panels), 9
panel.hist (panels.diag), 11
panel.qqnorm (panels.diag), 11
panel.reg (panels), 9
panel.smooth(), 10, 11
panel.boxplot (panels.diag), 11
panel_boxplot(), 13
panel.cor (panels), 9
panel_cor(), 5
panel_density (panels.diag), 11
panel_density(), 13
panel.ellipse (panels), 9
panel_hist (panels.diag), 11
panel_hist(), 13
panel_qqnorm (panels.diag), 11
panel_qqnorm(), 13
panel.reg (panels), 9
panel_smooth (panels), 9
panels, 9
panels.diag, 11
pcomp, 13
pcomp(), 19
print.Correlation (correlation), 3
print.pcomp (pcomp), 13
print.summary.Correlation (correlation), 3
print.summary.pcomp (pcomp), 13
proc.time(), 18
qqnorm(), 13
rlm(), 10
ROWS (nr), 8
rwb.colors (colors), 2
rwb_colors (colors), 2
rwg.colors (colors), 2
rwg_colors (colors), 2
ryg.colors (colors), 2
scale(), 15
SciViews-package, 2
scores (pcomp), 13
screeplot.pcomp (pcomp), 13
seq_along(), 6
summary.Correlation (correlation), 3
summary.pcomp (pcomp), 13
symnum(), 5
system.time(), 18
text.pcomp (pcomp), 13
timing, 18
vectorplot, 18
vectorplot(), 17, 19

proc.time(), 18
print.Correlation (correlation), 3
print.pcomp (pcomp), 13
print.summary.Correlation
   (correlation), 3
print.summary.pcomp (pcomp), 13
plot corre(), 5
points.pcomp (pcomp), 13
pcomp(), 15, 17
predict.pcomp (pcomp), 13
princomp(), 15, 17