

Package: ggbiplot (via r-universe)

October 11, 2026

Type Package

Title A Grammar of Graphics Implementation of Biplots

Version 0.6.6

Date 2026-09-07

Description A 'ggplot2' based implementation of biplots, giving a representation of a dataset in a two dimensional space accounting for the greatest variance, together with variable vectors showing how the data variables relate to this space. It provides a replacement for stats::biplot(), but with many enhancements to control the analysis and graphical display. It implements biplot and scree plot methods which can be used with the results of prcomp(), princomp(), FactoMineR::PCA(), ade4::dudi.pca() or MASS::lda() and can be customized using 'ggplot2' techniques.

Depends R (>= 4.1.0), ggplot2

Imports ggarrow, scales

Suggests corplot, dplyr, MASS, broom, tidyr

License GPL-2

Encoding UTF-8

Language en-US

URL <https://github.com/friendly/ggbiplot>,
<https://friendly.github.io/ggbiplot/>

BugReports <https://github.com/friendly/ggbiplot/issues>

LazyData true

Config/roxygen2/version 8.1.0

Repository <https://friendly.r-universe.dev>

Date/Publication 2026-09-10 16:24:47 UTC

RemoteUrl <https://github.com/friendly/ggbiplot>

RemoteRef HEAD

RemoteSha 6f551215272a78afd890b06734e7e36f0fedf310

Contents

crime	2
get_SVD	3
ggbiplot	4
ggscreeplot	9
ggvector	11
reflect	13
wine	15
Index	17

crime	<i>U. S. Crimes</i>
-------	---------------------

Description

This dataset gives rates of occurrence (per 100,000 people) various serious crimes in each of the 50 U. S. states, originally from the United States Statistical Abstracts (1970). The data were analyzed by John Hartigan (1975) in his book *Clustering Algorithms* and were later reanalyzed by Friendly (1991).

Usage

`data(crime)`

Format

A data frame with 50 observations on the following 10 variables.

- `state` state name, a character vector
- `murder` a numeric vector
- `rape` a numeric vector
- `robbery` a numeric vector
- `assault` a numeric vector
- `burglary` a numeric vector
- `larceny` a numeric vector
- `auto` auto thefts, a numeric vector
- `st` state abbreviation, a character vector
- `region` region of the U.S., a factor with levels Northeast South North Central West

Source

The data are originally from the United States Statistical Abstracts (1970). This dataset also appears in the SAS/Stat Sample library, *Getting Started Example for PROC PRINCOMP*, https://support.sas.com/documentation/onlinedoc/stat/ex_code/131/princgs.html, from which the current copy was derived.

References

- Friendly, M. (1991). *SAS System for Statistical Graphics*. SAS Institute.
 Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley and Sons.

Examples

```
data(crime)
library(ggplot2)
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)

ggbiplot(crime.pca,
  labels = crime$st ,
  circle = TRUE,
  varname.size = 4,
  varname.color = "red") +
  theme_minimal(base_size = 14)
```

get_SVD

Extract the SVD components from a PCA-like object

Description

Biplots are based on the Singular Value Decomposition, which for a data matrix is

$$\mathbf{X}/\sqrt{n} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

but these are computed and returned in quite different forms by various PCA-like methods. This function provides a common interface, returning the components with standard names.

Usage

```
get_SVD(pcobj)
```

Arguments

pcobj an object returned by `prcomp`, `princomp`, `PCA`, `dudi.pca`, or `lda`

Value

A list of four elements

n The sample size on which the analysis was based

U Left singular vectors, giving observation scores

D vector of singular values, the diagonal elements of the matrix **D**, which are also the square roots of the eigenvalues of $\mathbf{X}\mathbf{X}'$

V Right singular vectors, giving variable loadings

Examples

```
data(crime)
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)

crime.svd <- get_SVD(crime.pca)
names(crime.svd)
crime.svd$D
```

ggbiplot

Biplot for Principal Components using ggplot2

Description

A biplot simultaneously displays information on the observations (as points) and the variables (as vectors) in a multidimensional dataset. The 2D biplot is typically based on the first two principal components of a dataset, giving a rank 2 approximation to the data. The “bi” in biplot refers to the fact that two sets of points (i.e., the rows and columns of the data matrix) are visualized by scalar products, not the fact that the display is usually two-dimensional.

The biplot method for principal component analysis was originally defined by Gabriel (1971, 1981). Gower & Hand (1996) give a more complete treatment. Greenacre (2010) is a practical user-oriented guide to biplots. Gower et al. (2011) is the most up to date exposition of biplot methodology.

This implementation handles the results of a principal components analysis using [prcomp](#), [princomp](#), [PCA](#) and [dudi.pca](#); also handles a discriminant analysis using [lda](#).

Usage

```
ggbiplot(
  pcobj,
  choices = 1:2,
  scale = 1,
  pc.biplot = TRUE,
  obs.scale = 1 - scale,
  var.scale = scale,
  var.factor = 1,
  groups = NULL,
  geom.ind = "point",
  geom.var = c("arrow", "text"),
  point.size = 1.5,
  ellipse = FALSE,
  ellipse.prob = 0.68,
  ellipse.linewidth = 1.3,
  ellipse.fill = TRUE,
```

```

    ellipse.alpha = 0.25,
    labels = NULL,
    labels.size = 3,
    alpha = 1,
    var.axes = TRUE,
    circle = FALSE,
    circle.prob = 0.68,
    varname.size = 3,
    varname.adjust = 1.25,
    varname.color = "black",
    varname.abbrev = FALSE,
    varname.gap = 0,
    varname.angle = NULL,
    vector.args = list(),
    axis.title = "PC",
    clip = "off",
    ...
  )

```

Arguments

<code>pcobj</code>	an object returned by <code>prcomp</code> , <code>princomp</code> , <code>PCA</code> , <code>dudi.pca</code> , or <code>lda</code>
<code>choices</code>	Which components to plot? An integer vector of length 2.
<code>scale</code>	Covariance biplot (<code>scale = 1</code>), form biplot (<code>scale = 0</code>). When <code>scale = 1</code> (the default), the inner product between the variables approximates the covariance and the distance between the points approximates the Mahalanobis distance.
<code>pc.biplot</code>	Logical, for compatibility with <code>biplot.princomp()</code> . If <code>TRUE</code> , use what Gabriel (1971) refers to as a "principal component biplot", with $\alpha = 1$ and observations scaled up by $\sqrt{n}$ and variables scaled down by $\sqrt{n}$. Then inner products between variables approximate covariances and distances between observations approximate Mahalanobis distance.
<code>obs.scale</code>	Scale factor to apply to observations
<code>var.scale</code>	Scale factor to apply to variables
<code>var.factor</code>	Factor to be applied to variable vectors after scaling. This allows the variable vectors to be reflected (<code>var.factor = -1</code>) or expanded in length (<code>var.factor > 1</code>) for greater visibility. <code>reflect</code> provides a simpler way to reflect the variables.
<code>groups</code>	Optional factor variable indicating the groups that the observations belong to. If provided the points will be colored according to groups and this allows data ellipses also to be drawn when <code>ellipse = TRUE</code> .
<code>geom.ind</code>	a text specifying the geometry to be used for the observations. Allowed values are among <code>c("point", "text")</code> . Use "point" (to show only points); "text" to show only labels; <code>c("point", "text")</code> to show both.
<code>geom.var</code>	a text specifying the geometry to be used for the variables. Allowed values are among <code>c("arrow", "text")</code> . Use "arrow" (to show only vectors); "text" to show only labels; <code>c("arrow", "text")</code> to show both.
<code>point.size</code>	Size of observation points.

<code>ellipse</code>	Logical; draw a normal data ellipse for each group?
<code>ellipse.prob</code>	Coverage size of the data ellipse in Normal probability
<code>ellipse.linewidth</code>	Thickness of the line outlining the ellipses
<code>ellipse.fill</code>	Logical; should the ellipses be filled?
<code>ellipse.alpha</code>	Transparency value (0 - 1) for filled ellipses
<code>labels</code>	Optional vector of labels for the observations. Often, this will be specified as the <code>row.names()</code> of the dataset.
<code>labels.size</code>	Size of the text used for the point labels
<code>alpha</code>	Alpha transparency value for the points (0 = transparent, 1 = opaque)
<code>var.axes</code>	logical; draw arrows for the variables?
<code>circle</code>	draw a correlation circle? (only applies when <code>prcomp</code> was called with <code>scale = TRUE</code> and when <code>var.scale = 1</code>)
<code>circle.prob</code>	Size of the correlation circle
<code>varname.size</code>	Size of the text for variable names
<code>varname.adjust</code>	Adjustment factor the placement of the variable names, ≥ 1 means farther from the arrow
<code>varname.color</code>	Color for the variable vectors and names
<code>varname.abbrev</code>	logical; whether or not to abbreviate the variable names, using abbreviate .
<code>varname.gap</code>	Distance to pull variable-vector arrowheads back from their true endpoint, leaving a small gap before the label. Given as a plain number, this is in millimeters — a fixed physical distance on the drawn plot, <i>not</i> in the data units of the PC scores — so it does not scale with <code>'obs.scale'/'var.scale'</code> or with the size of the plotting device. Can also be a <code>[grid::unit()]</code> object for other units. Useful to keep arrowheads from overlapping the correlation circle (<code>'circle = TRUE'</code>) or crowding the variable-name labels. Passed to <code>[ggvector()]</code> 's <code>'gap'</code> argument (in turn <code>'resect_head'</code> of <code>[ggarrow::geom_arrow_segment()]</code>). Default <code>'0'</code> (no gap).
<code>varname.angle</code>	Angle (in degrees) for the variable-name text labels. Default <code>'NULL'</code> computes each label's angle from its own vector's direction, as before. Supply a fixed number (e.g. <code>'0'</code> for horizontal labels) to use the same angle for every label instead. Passed to <code>[ggvector()]</code> 's <code>'angle'</code> argument; applies only to the text labels, not the arrows.
<code>vector.args</code>	Named list of further arguments passed to the <code>[ggvector()]</code> call that draws the variable-vector arrows, overriding its defaults (e.g. <code>'color'</code> , <code>'adjust'</code> , <code>'gap'</code> , <code>'linewidth'</code> above) or adding new ones. Useful for arrow appearance not otherwise exposed as a <code>'ggbiplot()'</code> argument, e.g. <code>'list(arrow_head = ggarrow::arrow_head_line())'</code> for a different arrowhead shape, or <code>'list(linewidth = 1.4)'</code> to go back to the heavier shaft width used before the <code>'ggarrow'</code> switch. Anything not matched by a named argument of <code>[ggvector()]</code> itself is passed on to <code>[ggarrow::geom_arrow_segment()]</code> (e.g. <code>'justify'</code> , <code>'force_arrow'</code> , <code>'sep'</code> , <code>'distort'</code>). Applies only to the arrow layer, not to the variable-name text labels.
<code>axis.title</code>	character; the prefix used as the axis labels. Default: <code>"PC"</code> .

`clip` should geoms be clipped at the axis limits? Default: "off" — a long variable name, or one pushed outward by a large 'varname.adjust', can extend past the panel edge (ggplot2 computes axis limits from each layer's anchor coordinates only, not the rendered text's extent); "off" lets that overflow draw into the margin instead of truncating it. Pass "on" to restore hard clipping at the panel edge.

... other arguments passed down

Details

The biplot is constructed by using the singular value decomposition (SVD) to obtain a low-rank approximation to the data matrix $\mathbf{X}_{n \times p}$ (centered, and optionally scaled to unit variances) whose n rows are the observations and whose p columns are the variables.

Using the SVD, the matrix $\mathbf{X}$, of rank $r \leq p$ can be expressed *exactly* as

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}' = \sum_i^r \lambda_i \mathbf{u}_i \mathbf{v}_i',$$

where

- $\mathbf{U}$ is an $n \times r$ orthonormal matrix of observation scores; these are also the eigenvectors of $\mathbf{X}\mathbf{X}'$,
- $\mathbf{\Lambda}$ is an $r \times r$ diagonal matrix of singular values, $\lambda_1 \geq \lambda_2 \geq \dots \lambda_r$.
- $\mathbf{V}$ is an $r \times p$ orthonormal matrix of variable weights and also the eigenvectors of $\mathbf{X}'\mathbf{X}$.

Then, a rank 2 (or 3) PCA approximation $\hat{\mathbf{X}}$ to the data matrix used in the biplot can be obtained from the first 2 (or 3) singular values λ_i and the corresponding $\mathbf{u}_i, \mathbf{v}_i$ as

$$\mathbf{X} \approx \hat{\mathbf{X}} = \lambda_1 \mathbf{u}_1 \mathbf{v}_1' + \lambda_2 \mathbf{u}_2 \mathbf{v}_2'.$$

The variance of $\mathbf{X}$ accounted for by each term is λ_i^2 .

The biplot is then obtained by overlaying two scatterplots that share a common set of axes and have a between-set scalar product interpretation. Typically, the observations (rows of $\mathbf{X}$) are represented as points and the variables (columns of $\mathbf{X}$) are represented as vectors from the origin.

The scale factor, α allows the variances of the components to be apportioned between the row points and column vectors, with different interpretations, by representing the approximation $\hat{\mathbf{X}}$ as the product of two matrices,

$$\hat{\mathbf{X}} = (\mathbf{U}\mathbf{\Lambda}^\alpha)(\mathbf{\Lambda}^{1-\alpha}\mathbf{V}')$$

The choice $\alpha = 1$, assigning the singular values totally to the left factor, gives a distance interpretation to the row display and $\alpha = 0$ gives a distance interpretation to the column display. $\alpha = 1/2$ gives a symmetrically scaled biplot.

When the singular values are assigned totally to the left or to the right factor, the resultant coordinates are called *principal coordinates* and the sum of squared coordinates on each dimension equal the corresponding singular value. The other matrix, to which no part of the singular values is assigned, contains the so-called *standard coordinates* and have sum of squared values equal to 1.0.

Scales and legend

When the ‘groups’ argument is not NULL, the function uses that value to set the aesthetics for ‘color’, ‘fill’ and ‘shape’. If you override the defaults using `scale_color_discrete`, etc., you may find that duplicate legends are produced. To avoid this, you need to have the same name for aesthetics to be merged in the legend, for example,

```
scale_fill_discrete(name = 'Species')
scale_color_discrete(name = 'Species')
```

or,

```
labs(fill = "Species", color = "Species")
```

Value

a ggplot2 plot object of class `c("gg", "ggplot")`

Author(s)

Vincent Q. Vu., Michael Friendly

References

- Gabriel, K. R. (1971). The biplot graphical display of matrices with application to principal component analysis. *Biometrika*, **58**, 453–467. doi:10.2307/2334381.
- Gabriel, K. R. (1981). Biplot display of multivariate matrices for inspection of data and diagnosis. In V. Barnett (Ed.), *Interpreting Multivariate Data*. London: Wiley.
- Greenacre, M. (2010). *Biplots in Practice*. BBVA Foundation, Bilbao, Spain. Available for free at <https://www.fbbva.es/microsite/multivariate-statistics/>.
- J.C. Gower and D. J. Hand (1996). *Biplots*. Chapman & Hall.
- Gower, J. C., Lubbe, S. G., & Roux, N. J. L. (2011). *Understanding Biplots*. Wiley.

See Also

[reflect](#), [ggscreeplot](#); [biplot](#) for the original stats package version; [fviz_pca_biplot](#) for the factoextra package version.

Examples

```
data(wine)
library(ggplot2)
wine.pca <- prcomp(wine, scale. = TRUE)
ggbiplot(wine.pca,
  obs.scale = 1, var.scale = 1,
  varname.size = 4,
  groups = wine.class,
  ellipse = TRUE, circle = TRUE)

# Easier interpretation if the axes are reflected
wine.pca <- reflect(wine.pca)
```

```

# Use direct labels rather than a legend
means <- aggregate(cbind(PC1, PC2) ~ wine.class,
  data = wine.pca$x, FUN = mean)

ggbiplot(wine.pca,
  obs.scale = 1, var.scale = 1,
  groups = wine.class,
  varname.size = 4,
  ellipse = TRUE,
  circle = TRUE) +
  geom_label(data = means, aes(x=PC1, y=PC2, label = wine.class)) +
  theme(legend.position = 'none')

data(iris)
iris.pca <- prcomp (~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
  data=iris,
  scale. = TRUE)
ggbiplot(iris.pca, obs.scale = 1, var.scale = 1,
  groups = iris$Species, point.size=2,
  varname.size = 5,
  varname.color = "black",
  varname.adjust = 1.2,
  varname.gap = 2,      # pull arrowheads off the correlation circle, in mm
  varname.angle = 0,    # draw all variable-name labels horizontally
  ellipse = TRUE,
  circle = TRUE) +
  labs(fill = "Species", color = "Species") +
  theme_minimal(base_size = 14) +
  theme(legend.direction = 'horizontal', legend.position = 'top')

```

ggscreeplot

Screeplot for Principal Components

Description

Produces scree plots (Cattell, 1966) of the variance proportions explained by each dimension against dimension number from various PCA-like dimension reduction techniques.

Usage

```

ggscreeplot(
  pcobj,
  type = c("pev", "cev"),
  size = 4,
  shape = 19,
  color = "black",
  linetype = 1,
  linewidth = 1
)

```

Arguments

<code>pcobj</code>	an object returned by <code>prcomp</code> , <code>princomp</code> , <code>PCA</code> , <code>dudi.pca</code> , or <code>lda</code>
<code>type</code>	the type of scree plot, one of <code>c('pev', 'cev')</code> . 'pev' plots the proportion of explained variance, i.e. the eigenvalues divided by the trace. 'cev' plots the cumulative proportion of explained variance, i.e. the partial sum of the first k eigenvalues divided by the trace.
<code>size</code>	point size
<code>shape</code>	shape of the points. Default: 19, a filled circle.
<code>color</code>	color for points and line. Default: "black".
<code>linetype</code>	type of line
<code>linewidth</code>	width of line

Value

A `ggplot2` object with the aesthetics `x = PC`, `y = yvar`

References

Cattell, R. B. (1966). The Scree Test For The Number Of Factors. *Multivariate Behavioral Research*, 1, 245–276.

Examples

```
library(ggplot2)
data(wine)
wine.pca <- prcomp(wine, scale. = TRUE)
ggscreeplot(wine.pca)

# show horizontal lines for 80, 90% of cumulative variance
ggscreeplot(wine.pca, type = "cev") +
  geom_hline(yintercept = c(0.8, 0.9), color = "blue")

# Make a fancy screeplot, highlighting the scree starting at component 4
data(crime)
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)

(crime.eig <- crime.pca |>
  broom::tidy(matrix = "eigenvalues"))

ggscreeplot(crime.pca) +
  stat_smooth(data = crime.eig |> dplyr::filter(PC>=4),
    aes(x=PC, y=percent), method = "lm",
    se = FALSE,
    fullrange = TRUE)
```

ggvector

*Draw labeled vectors in a ggplot scene***Description**

`'ggvector()'` draws one or more vectors from a common origin, as arrows and/or text labels, using `[ggarrow::geom_arrow_segment()]` to draw the arrows. It is used internally by `[ggbiplot()]` to draw the variable vectors, but it can also be used on its own to add extra vectors (e.g., supplementary variables) to an existing plot.

Usage

```
ggvector(
  x,
  y,
  label = NULL,
  geom.var = c("arrow", "text"),
  scale = 1,
  origin = c(0, 0),
  color = "black",
  linewidth = 0.9,
  arrow_head = ggarrow::arrow_head_wings(),
  length = 4,
  gap = 0,
  adjust = 1.25,
  size = 3,
  lineheight = 0.75,
  angle = NULL,
  ...
)
```

Arguments

<code>x, y</code>	coordinates of the vector ends
<code>label</code>	optional text labels for the vector ends. If <code>'NULL'</code> , no labels are drawn.
<code>geom.var</code>	character vector specifying what to draw: any of <code>"arrow"</code> , <code>"text"</code>
<code>scale</code>	scale factor applied to <code>'x'</code> and <code>'y'</code> before drawing
<code>origin</code>	origin of the vectors, a vector <code>'c(x, y)'</code>
<code>color</code>	color for the vectors and their labels
<code>linewidth</code>	linewidth for the vector arrows. Default <code>'0.9'</code> — thinner than the shaft width <code>'grid::arrow()'</code> used pre- <code>'ggarrow' (1.4)</code> , because <code>[ggarrow::arrow_head_wings()]</code> 's default ornament reads visually heavier than the old plain triangular arrowhead at the same linewidth.
<code>arrow_head</code>	an arrowhead ornament, e.g., from <code>[ggarrow::arrow_head_wings()]</code> or <code>[ggarrow::arrow_head_line()]</code> , passed to <code>[ggarrow::geom_arrow_segment()]</code>

length	length of the arrowhead; passed to <code>[ggarrow::geom_arrow_segment()]</code>
gap	distance to pull the arrowhead back from the vector's true endpoint, passed as 'resect_head' to <code>[ggarrow::geom_arrow_segment()]</code> . Given as a plain number, this is in millimeters — a fixed physical distance on the drawn plot, <i>not</i> in the (arbitrary) data units of 'x'/'y' — so the same 'gap' looks bigger or smaller depending on plot size/scale. Can also be a <code>[grid::unit()]</code> object for other units. Useful to keep arrowheads clear of a correlation circle or of crowded labels; does not move the label itself, which is still placed at the true '(x, y)' endpoint. Default '0' (no gap; arrowhead tip touches the endpoint).
adjust	adjustment factor for label placement, ≥ 1 means farther from the arrowhead
size	text size for labels
lineheight	line height for (possibly multi-line) labels
angle	angle (in degrees) for the text labels. Default 'NULL' computes each label's angle from its own vector's direction, as before. Supply a fixed number (e.g. '0' for horizontal labels) to use the same angle for every label instead, or a vector recycled across labels for per-label control.
...	other arguments passed to <code>[ggarrow::geom_arrow_segment()]</code>

Details

The arrowhead size ('length') scales with 'linewidth', not with the data-space length of the vectors. If several short vectors point in similar directions, a 'scale' too small relative to 'length'/'linewidth' can make the arrowheads overlap into a cluttered mass near 'origin' — scale the vectors up (as `[ggbiplot()]` does internally, relative to the spread of the point scores) so the heads don't dominate the shafts.

Value

A list of ggplot2 layers that can be added to an existing plot with '+'.

Examples

```
data(wine)
library(ggplot2)
wine.pca <- prcomp(wine, scale. = TRUE)
# scale loadings up so vectors are comparable in length to the point scores
v <- as.data.frame(6 * wine.pca$rotation[, 1:2])

ggplot(as.data.frame(wine.pca$x), aes(PC1, PC2)) +
  geom_point() +
  ggvector(v$PC1, v$PC2, label = rownames(v), color = "brown")

# angle = 0 draws all labels horizontally, instead of along each vector
ggplot(as.data.frame(wine.pca$x), aes(PC1, PC2)) +
  geom_point() +
  ggvector(v$PC1, v$PC2, label = rownames(v), color = "brown", angle = 0)
```

reflect

*Reflect Columns in a Principal Component-like Object***Description**

Principal component-like objects have variable loadings (the eigenvectors of the covariance/correlation matrix) whose signs are arbitrary, in the sense that a given column can be reflected (multiplied by -1) without changing the fit. `reflect()` is a generic function, with methods for PCA-like objects (`'prcomp'`, `'princomp'`, `[FactoMineR::PCA()]`, `'lda'`) that reflect the variable loadings and corresponding observation scores (and supplementary variable coordinates, for `'PCA'`). The S3 methods also allow for a plain `'data.frame'`, `'matrix'`, or a `'list'` of these, that negate the specified columns directly.

This is often useful for interpreting a biplot, for example when a component (often the first) has all negative signs.

Usage

```
reflect(pcobj, ...)

## S3 method for class 'prcomp'
reflect(pcobj, columns = 1:2, scale = 1, ...)

## S3 method for class 'princomp'
reflect(pcobj, columns = 1:2, scale = 1, ...)

## S3 method for class 'PCA'
reflect(pcobj, columns = 1:2, scale = 1, ...)

## S3 method for class 'lda'
reflect(pcobj, columns = 1:2, scale = 1, ...)

## S3 method for class 'data.frame'
reflect(pcobj, columns, scale = 1, ...)

## S3 method for class 'matrix'
reflect(pcobj, columns, scale = 1, ...)

## S3 method for class 'list'
reflect(pcobj, columns, scale = 1, ...)

## Default S3 method:
reflect(pcobj, ...)
```

Arguments

`pcobj` an object to reflect: one of `'prcomp'`, `'princomp'`, `[FactoMineR::PCA()]`, `'lda'`, a `'data.frame'`, a `'matrix'`, or a `'list'` of data frames/matrices with matching

	columns
...	arguments passed to methods
columns	for PCA-like objects, a vector of indices of the dimensions to reflect (default '1:2'). For the 'data.frame'/'matrix'/'list' methods, a vector of column indices or names to negate – there is no sensible default for an arbitrary table, so this must be supplied.
scale	a constant applied along with the reflection, so the selected columns become '-scale * columns' rather than just '-1 * columns'. Default '1' (plain reflection, no change in magnitude). Useful, e.g., to stretch or shrink variable vectors in a biplot at the same time as reflecting them; a negative 'scale' scales without reflecting.

Value

The object, of the same class, with the specified columns transformed to '-scale * columns' – the variable loadings and observation scores for PCA-like objects, or the specified columns directly for a 'data.frame'/'matrix'/'list'.

Author(s)

Michael Friendly

See Also

[prcomp](#), [princomp](#), [PCA](#), [lda](#)

Examples

```
data(crime)
crime.pca <-
  crime |>
  dplyr::select(where(is.numeric)) |>
  prcomp(scale. = TRUE)

biplot(crime.pca)

crime.pca <- reflect(crime.pca) # reflect columns 1:2
biplot(crime.pca)

iris.lda <- MASS::lda(Species ~ ., data=iris)
#reflect the first dimension
iris.lda1 <- reflect(iris.lda, columns = 1)
# compare predicted scores
predict(iris.lda)$x |> head()
predict(iris.lda1)$x |> head()

# reflect a column of an ordinary data.frame, by name or index
mtcars2 <- reflect(mtcars, columns = "mpg")
head(mtcars2$mpg)
```

```
# reflect AND stretch the variable vectors by a factor of 2
crime.pca3 <- reflect(crime.pca, scale = 2)
```

wine

Wine dataset

Description

Results of a chemical analysis of wines grown in the same region in Italy, derived from three different cultivars. The analysis determined the quantities of 13 chemical constituents found in each of the three types of wines.

The grape varieties (cultivars), 'barolo', 'barbera', and 'grignolino', are indicated in `wine.class`.

Usage

```
data(wine)
```

Format

A wine data frame consisting of 178 observations (rows) and 13 columns and vector `wine.class` of factors indicating the cultivars. The variables are:

Alcohol a numeric vector

MalicAcid Malic acid, a numeric vector

Ash Ash, a numeric vector

AlcAsh Alkalinity of ash, a numeric vector

Mg Magnesium, a numeric vector

Phenols total phenols, a numeric vector

Flav Flavanoids, a numeric vector

NonFlavPhenols Nonflavanoid phenols, a numeric vector

Proa Proanthocyanins, a numeric vector

Color Color intensity, a numeric vector

Hue a numeric vector

OD D280/OD315 of diluted wines, a numeric vector

Proline a numeric vector

Source

UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/datasets/Wine>)

Examples

```
data(wine)
table(wine.class)

wine.pca <- prcomp(wine, scale. = TRUE)
ggscreeplot(wine.pca)
ggbiplot(wine.pca,
  obs.scale = 1, var.scale = 1,
  groups = wine.class, ellipse = TRUE, circle = TRUE)
```

Index

- * **dataset**
 - crime, [2](#)
 - wine, [15](#)
- abbreviate, [6](#)
- biplot, [8](#)
- crime, [2](#)
- dudi.pca, [3–5](#), [10](#)
- fviz_pca_biplot, [8](#)
- get_SVD, [3](#)
- ggbiplot, [4](#)
- ggscreeplot, [8](#), [9](#)
- ggvector, [11](#)
- lda, [3–5](#), [10](#), [14](#)
- PCA, [3–5](#), [10](#), [14](#)
- prcomp, [3–5](#), [10](#), [14](#)
- princomp, [3–5](#), [10](#), [14](#)
- reflect, [5](#), [8](#), [13](#)
- wine, [15](#)