Package 'factoextra'

August 29, 2016

Type Package

Title Extract and Visualize the Results of Multivariate Data Analyses

Version 1.0.3

Date 2016-03-31

Description Provides some easy-to-use functions to extract and visualize
the output of multivariate data analyses, including 'PCA' (Principal Component
Analysis), 'CA' (Correspondence Analysis), 'MCA' (Multiple Correspondence Analysis),
'MFA' (Multiple Factor Analysis) and 'HMFA' (Hierarchical Multiple Factor Analysis)
functions from different R packages. It contains also functions for simplifying some clustering analysis steps
and provides 'ggplot2' - based elegant data visualization.

License GPL-2

LazyData true

Depends R (>= 3.1.0), ggplot2

Imports cluster, dendextend, grid, stats, reshape2, ggrepel, abind

Suggests ade4, ca, FactoMineR, MASS, knitr

URL http://www.sthda.com/english/rpkgs/factoextra

BugReports https://github.com/kassambara/factoextra/issues

Collate 'cluster_utilities.R' 'decathlon2.R' 'eigenvalue.R'
'utilities.R' 'dist.R' 'fviz_dend.R' 'hcut.R'
'print.factoextra.R' 'get_pca.R' 'fviz_cluster.R' 'eclust.R'
'facto_summarize.R' 'fviz_add.R' 'fviz_ca.R' 'fviz_contrib.R'
'fviz_cos2.R' 'get_hmfa.R' 'fviz_hmfa.R' 'get_mca.R'
'fviz_mca.R' 'get_mfa.R' 'fviz_mfa.R' 'fviz_nbclust.R'
'fviz_pca.R' 'fviz_silhouette.R' 'get_ca.R'
'get_clust_tendency.R' 'hkmeans.R' 'housetasks.R'
'multishapes.R' 'poison.R'

RoxygenNote 5.0.1

NeedsCompilation no

Author Alboukadel Kassambara [aut, cre],
Fabian Mundt [aut]

Maintainer Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Repository CRAN

Date/Publication 2016-04-26 16:50:39

R topics documented:

`decathlon2` *Athletes' performance in decathlon*

Description

Athletes' performance during two sporting meetings

Usage

```
data("decathlon2")
```

Format

A data frame with 27 observations on the following 13 variables.

`X100m` a numeric vector

`Long.jump` a numeric vector

`Shot.put` a numeric vector

`High.jump` a numeric vector

`X400m` a numeric vector

`X110m.hurdle` a numeric vector

`Discus` a numeric vector

`Pole.vault` a numeric vector

`Javeline` a numeric vector

`X1500m` a numeric vector

`Rank` a numeric vector corresponding to the rank

`Points` a numeric vector specifying the point obtained

`Competition` a factor with levels `Decastar` `OlympicG`

Source

This data is a subset of decathlon data in FactoMineR package.

Examples

```
data(decathlon2)
decathlon.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon.active, scale = TRUE)
fviz_pca_biplot(res.pca, data = decathlon.active)
```

dist *Enhanced Distance Matrix Computation and Visualization*

Description

Clustering methods classify data samples into groups of similar objects. This process requires some methods for measuring the distance or the (dis)similarity between the observations. Read more: STHDA website - clarifying distance measures..

- get_dist(): Computes a distance matrix between the rows of a data matrix. Compared to the standard dist() function, it supports correlation-based distance measures including "pearson", "kendall" and "spearman" methods.
- fviz_dist(): Visualizes a distance matrix

Usage

```
get_dist(x, method = "euclidean", stand = FALSE, ...)

fviz_dist(dist.obj, order = TRUE, show_labels = TRUE, lab_size = NULL,
  gradient = list(low = "red", mid = "white", high = "blue"))
```

Arguments

x	a numeric matrix or a data frame.
method	the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman" or "kendall".
stand	logical value; default is FALSE. If TRUE, then the data will be standardized using the function scale(). Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's standard deviation.
...	other arguments to be passed to the function dist().
dist.obj	an object of class "dist" as generated by the function dist() or get_dist().
order	logical value. if TRUE the ordered dissimilarity image (ODI) is shown.
show_labels	logical value. If TRUE, the labels are displayed.
lab_size	the size of labels.
gradient	a list containing three elements specifying the colors for low, mid and high values in the ordered dissimilarity image. The element "mid" can take the value of NULL.

Value

- get_dist(): returns an object of class "dist".
- fviz_dist(): returns a ggplot2

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

dist

Examples

```
data(USArrests)
res.dist <- get_dist(USArrests, stand = TRUE, method = "pearson")

fviz_dist(res.dist,
   gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))
```

---

eclust *Visual enhancement of clustering analysis*

Description

Visual enhancement of clustering analysis.

Usage

```
eclust(x, FUNcluster = c("kmeans", "pam", "clara", "fanny", "hclust", "agnes",
  "diana"), k = NULL, k.max = 10, stand = FALSE, graph = TRUE,
  hc_metric = "euclidean", hc_method = "ward.D2", gap_maxSE = list(method
  = "firstmax", SE.factor = 1), nboot = 100, verbose = interactive(),
  seed = 123, ...)
```

Arguments

x	numeric vector, data matrix or data frame
FUNcluster	a clustering function including "kmeans", "pam", "clara", "fanny", "hclust", "agnes" and "diana". Abbreviation is allowed.
k	the number of clusters to be generated. If NULL, the gap statistic is used to estimate the appropriate number of clusters. In the case of kmeans, k can be either the number of clusters, or a set of initial (distinct) cluster centers.
k.max	the maximum number of clusters to consider, must be at least two.
stand	logical value; default is FALSE. If TRUE, then the data will be standardized using the function scale(). Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's standard deviation.
graph	logical value. If TRUE, cluster plot is displayed.

hc_metric	character string specifying the metric to be used for calculating dissimilarities between observations. Allowed values are those accepted by the function dist() [including "euclidean", "manhattan", "maximum", "canberra", "binary", "minkowski"] and correlation based distance measures ["pearson", "spearman" or "kendall"]. Used only when FUNcluster is a hierarchical clustering function such as one of "hclust", "agnes" or "diana".
hc_method	the agglomeration method to be used (?hclust): "ward.D", "ward.D2", "single", "complete", "average", ...
gap_maxSE	a list containing the parameters (method and SE.factor) for determining the location of the maximum of the gap statistic (Read the documentation ?cluster::maxSE).
nboot	integer, number of Monte Carlo ("bootstrap") samples. Used only for determining the number of clusters using gap statistic.
verbose	logical value. If TRUE, the result of progress is printed.
seed	integer used for seeding the random number generator.
...	other arguments to be passed to FUNcluster.

Value

Returns an object of class "eclust" containing the result of the standard function used (e.g., kmeans, pam, hclust, agnes, diana, etc.).

It includes also:

- cluster: the cluster assignement of observations after cutting the tree
- nbclust: the number of clusters
- silinfo: the silhouette information of observations, including $widths (silhouette width values of each observation), $clus.avg.widths (average silhouette width of each cluster) and $avg.width (average width of all clusters)
- size: the size of clusters
- data: a matrix containing the original or the standardized data (if stand = TRUE)

The "eclust" class has method for fviz_silhouette(), fviz_dend(), fviz_cluster().

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_silhouette, fviz_dend, fviz_cluster

Examples

```
# Load and scale data
data("USArrests")
df <- scale(USArrests)
```

```
# Enhanced k-means clustering
res.km <- eclust(df, "kmeans")
# Silhouette plot
fviz_silhouette(res.km)
# Optimal number of clusters using gap statistics
res.km$nbclust
# Print result
 res.km

 # Enhanced hierarchical clustering
 res.hc <- eclust(df, "hclust") # compute hclust
  fviz_dend(res.hc) # dendrogam
  fviz_silhouette(res.hc) # silhouette plot
```

`eigenvalue` *Extract and visualize the eigenvalues/variances of dimensions*

Description

Eigenvalues correspond to the amount of the variation explained by each principal component (PC). Read more: Principal Component Analysis

- get_eig(): Extract the eigenvalues/variances of the principal dimensions
- fviz_eig(): Plot the eigenvalues/variances against the number of dimensions
- get_eigenvalue(): an alias of get_eig()
- fviz_screeplot(): an alias of fviz_eig()

These functions support the results of Principal Component Analysis (PCA), Correspondence Analysis (CA), Multiple Correspondence Analysis (MCA), Multiple Factor Analysis (MFA) and Hierarchical Multiple Factor Analysis (HMFA) functions.

Usage

```
get_eig(X)

get_eigenvalue(X)

fviz_eig(X, choice = c("variance", "eigenvalue"), geom = c("bar", "line"),
  barfill = "steelblue", barcolor = "steelblue", linecolor = "black",
  ncp = 10, addlabels = FALSE, hjust = 0, ...)

fviz_screeplot(...)
```

Arguments

X	an object of class PCA, CA, MCA, MFA and HMFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca and mjca [ca package].
choice	a text specifying the data to be plotted. Allowed values are "variance" or "eigenvalue".
geom	a text specifying the geometry to be used for the graph. Allowed values are "bar" for barplot, "line" for lineplot or c("bar", "line") to use both types.
barfill	fill color for bar plot.
barcolor	outline color for bar plot.
linecolor	color for line plot (when geom contains "line").
ncp	a numeric value specifying the number of dimensions to be shown.
addlabels	logical value. If TRUE, labels are added at the top of bars or points showing the information retained by each dimension.
hjust	horizontal adjustment of the labels.
...	optional arguments to be passed to the functions geom_bar(), geom_line(), geom_text() or fviz_eig().

Value

- get_eig() (or get_eigenvalue()): returns a data.frame containing 3 columns: the eigenvalues, the percentage of variance and the cumulative percentage of variance retained by each dimension.
- fviz_eig() (or fviz_screeplot()): returns a ggplot2

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

See Also

fviz_pca, fviz_ca, fviz_mca, fviz_mfa, fviz_hmfa

Examples

```
# Principal Component Analysis
# ++++++++++++++++++++++++++
data(iris)
res.pca <- prcomp(iris[, -5],  scale = TRUE)

# Extract eigenvalues/variances
get_eig(res.pca)

# Default plot
```

```
fviz_eig(res.pca)

# Customize the plot
   # - Add labels
   # - Change line color, bar fill and color.
   # - Change axis limits and themes

p <- fviz_eig(res.pca, addlabels = TRUE, hjust = -0.3,
           linecolor = "#FC4E07",
           barfill="white", barcolor ="darkblue")+
     ylim(0, 85)+ # y axis limits
     theme_minimal() # themes: http://www.sthda.com/english/wiki/ggplot2-themes
print(p)

# Change plot title and axis labels
p + labs(title = "Variances - PCA",
        x = "Principal Components", y = "% of variances")


# Scree plot - Eigenvalues
fviz_eig(res.pca, choice = "eigenvalue", addlabels=TRUE)

# Use only bar  or line plot: geom = "bar" or geom = "line"
fviz_eig(res.pca, geom="line")


# Correspondence Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(housetasks)
res.ca <- CA(housetasks, graph = FALSE)
get_eig(res.ca)
fviz_eig(res.ca, linecolor = "#FC4E07",
   barcolor = "#00AFBB", barfill = "#00AFBB")+
   theme_minimal()

# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2,
              quali.sup = 3:4, graph=FALSE)
get_eig(res.mca)
fviz_eig(res.mca, linecolor = "#FC4E07",
   barcolor = "#2E9FDF", barfill = "#2E9FDF")+
   theme_minimal()


# Multiple Factor Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(wine)
res.mfa <- MFA(wine, group=c(2,5,3,10,9,2), type=c("n",rep("s",5)),
```

```
                ncp=5, name.group=c("orig","olf","vis","olfag","gust","ens"),
                num.group.sup=c(1,6), graph=FALSE)
get_eig(res.mfa)
fviz_eig(res.mfa, linecolor = "#FC4E07",
   barcolor = "#E7B800", barfill = "#E7B800")+
   theme_minimal()
```

`facto_summarize` *Subset and summarize the output of factor analyses*

Description

Subset and summarize the results of Principal Component Analysis (PCA), Correspondence Analysis (CA), Multiple Correspondence Analysis (MCA) and Multiple Factor Analysis (MFA) functions from several packages.

Usage

```
facto_summarize(X, element, node.level = 1, group.names, result = c("coord",
  "cos2", "contrib"), axes = 1:2, select = NULL)
```

Arguments

`X`	an object of class PCA, CA, MCA and MFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package].
`element`	the element to subset from the output. Possible values are "row" or "col" for CA; "var" or "ind" for PCA and MCA; 'quanti.var', 'quali.var' or 'ind' for MFA
`node.level`	a single number indicating the HMFA node level.
`group.names`	a vector containing the name of the groups (by default, NULL and the group are named group.1, group.2 and so on).
`result`	the result to be extracted for the element. Possible values are the combination of c("cos2", "contrib", "coord")
`axes`	a numeric vector specifying the axes of interest. Default values are 1:2 for axes 1 and 2.
`select`	a selection of variables. Allowed values are NULL or a list containing the arguments name, cos2 or contrib. Default is list(name = NULL, cos2 = NULL, contrib = NULL): • name: is a character vector containing variable names to be selected • cos2: if cos2 is in [0, 1], ex: 0.6, then variables with a cos2 > 0.6 are selected. if cos2 > 1, ex: 5, then the top 5 variables with the highest cos2 are selected • contrib: if contrib > 1, ex: 5, then the top 5 variables with the highest cos2 are selected.

Details

If length(axes) > 1, then the columns contrib and cos2 correspond to the total contributions and total cos2 of the axes. In this case, the column coord is calculated as x^2 + y^2 + ...+; x, y, ... are the coordinates of the points on the specified axes.

Value

A data frame containing the (total) coord, cos2 and the contribution for the axes.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Principal component analysis
# ++++++++++++++++++++++++++++++
data(decathlon2)
decathlon2.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon2.active,  scale = TRUE)

# Summarize variables on axes 1:2
facto_summarize(res.pca, "var", axes = 1:2)[,-1]
# Select the top 5 contributing variables
facto_summarize(res.pca, "var", axes = 1:2,
           select = list(contrib = 5))[,-1]
# Select variables with cos2 >= 0.6
facto_summarize(res.pca, "var", axes = 1:2,
           select = list(cos2 = 0.6))[,-1]
# Select by names
facto_summarize(res.pca, "var", axes = 1:2,
     select = list(name = c("X100m", "Discus", "Javeline")))[,-1]

# Summarize individuals on axes 1:2
facto_summarize(res.pca, "ind", axes = 1:2)[,-1]

# Correspondence Analysis
# ++++++++++++++++++++++++++
# Install and load FactoMineR to compute CA
# install.packages("FactoMineR")
library("FactoMineR")
data("housetasks")
res.ca <- CA(housetasks, graph = FALSE)
# Summarize row variables on axes 1:2
facto_summarize(res.ca, "row", axes = 1:2)[,-1]
# Summarize column variables on axes 1:2
facto_summarize(res.ca, "col", axes = 1:2)[,-1]
```

```
# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2,
               quali.sup = 3:4, graph=FALSE)
# Summarize variables on axes 1:2
res <- facto_summarize(res.mca, "var", axes = 1:2)
head(res)
# Summarize individuals on axes 1:2
res <- facto_summarize(res.mca, "ind", axes = 1:2)
head(res)

# Multiple factor Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
               name.group=c("desc","desc2","symptom","eat"),
               num.group.sup=1:2, graph=FALSE)
# Summarize categorcial variables on axes 1:2
res <- facto_summarize(res.mfa, "quali.var", axes = 1:2)
head(res)
# Summarize individuals on axes 1:2
res <- facto_summarize(res.mfa, "ind", axes = 1:2)
head(res)
```

`fviz_add` *Add supplementary data to a plot*

Description

Add supplementary data to a plot

Usage

```
fviz_add(ggp, df, axes = c(1, 2), geom = c("point", "arrow"),
  color = "blue", addlabel = TRUE, labelsize = 4, pointsize = 2,
  shape = 19, linetype = "dashed", jitter = list(what = "label", width =
  NULL, height = NULL))
```

Arguments

`ggp`	a ggplot2 plot.
`df`	a data frame containing the x and y coordinates
`axes`	a numeric vector of length 2 specifying the components to be plotted.

geom	a character specifying the geometry to be used for the graph Allowed values are "point" or "arrow" or "text"
color	the color to be used
addlabel	a logical value. If TRUE, labels are added
labelsize	the size of labels. Default value is 4
pointsize	the size of points
shape	point shape when geom ="point"
linetype	the linetype to be used when geom ="arrow"
jitter	a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)). • what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b". • width: degree of jitter in x direction • height: degree of jitter in y direction

Value

a ggplot2 plot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Principal component analysis
data(decathlon2)
decathlon2.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon2.active,  scale = TRUE)

# Visualize variables
p <- fviz_pca_var(res.pca)
print(p)

# Add supplementary variables
coord <- data.frame(PC1 = c(-0.7, 0.9), PC2 = c(0.25, -0.07))
rownames(coord) <- c("Rank", "Points")
print(coord)
fviz_add(p, coord, color ="blue", geom="arrow")
```

fviz_ca *Visualize Correspondence Analysis*

Description

Correspondence analysis (CA) is an extension of Principal Component Analysis (PCA) suited to analyze frequencies formed by two categorical variables. fviz_ca() provides ggplot2-based elegant visualization of CA outputs from the R functions: CA [in FactoMineR], ii) ca [in ca], coa [in ade4] and correspondence [in MASS]. Read more: Correspondence Analysis

- fviz_ca_row(): Graph of row variables
- fviz_ca_col(): Graph of column variables
- fviz_ca_biplot(): Biplot of row and column variables
- fviz_ca(): An alias of fviz_ca_biplot()

Usage

```
fviz_ca_row(X, axes = c(1, 2), shape.row = 19, geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.row = "blue", col.row.sup = "darkblue", alpha.row = 1,
  select.row = list(name = NULL, cos2 = NULL, contrib = NULL),
  map = "symmetric", repel = FALSE, jitter = list(what = "label", width =
  NULL, height = NULL), ...)

fviz_ca_col(X, axes = c(1, 2), shape.col = 17, geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.col = "red", col.col.sup = "darkred", alpha.col = 1,
  select.col = list(name = NULL, cos2 = NULL, contrib = NULL),
  map = "symmetric", repel = FALSE, jitter = list(what = "label", width =
  NULL, height = NULL), ...)

fviz_ca_biplot(X, axes = c(1, 2), shape.row = 19, shape.col = 17,
  geom = c("point", "text"), label = "all", invisible = "none",
  labelsize = 4, pointsize = 2, col.col = "red",
  col.col.sup = "darkred", alpha.col = 1, col.row = "blue",
  col.row.sup = "darkblue", alpha.row = 1, select.col = list(name = NULL,
  cos2 = NULL, contrib = NULL), select.row = list(name = NULL, cos2 = NULL,
  contrib = NULL), map = "symmetric", arrows = c(FALSE, FALSE),
  repel = FALSE, title = "CA factor map - Biplot", jitter = list(what =
  "label", width = NULL, height = NULL), ...)

fviz_ca(X, ...)
```

Arguments

X	an object of class CA [FactoMineR], ca [ca], coa [ade4]; correspondence [MASS].

`axes` a numeric vector of length 2 specifying the dimensions to be plotted.

`shape.row, shape.col`
the point shapes to be used for row/column variables. Default values are 19 for rows and 17 for columns.

`geom` a character specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "arrow", "text"). Use "point" (to show only points); "text" to show only labels; c("point", "text") or c("arrow", "text") to show both types.

`label` a character vector specifying the elements to be labelled. Default value is "all". Allowed values are "none" or the combination of c("row", "row.sup", "col", "col.sup"). Use "col" to label only active column variables; "col.sup" to label only supplementary columns; etc

`invisible` a character value specifying the elements to be hidden on the plot. Default value is "none". Allowed values are the combination of c("row", "row.sup","col", "col.sup").

`labelsize` font size for the labels

`pointsize` the size of points

`map` character string specifying the map type. Allowed options include: "symmetric", "rowprincipal", "colprincipal", "symbiplot", "rowgab", "colgab", "rowgreen" and "colgreen". See details

`repel` a boolean, whether to use ggrepel to avoid overplotting text labels or not.

`jitter` a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).

- what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b".
- width: degree of jitter in x direction
- height: degree of jitter in y direction

`...` optional arguments.

`col.col, col.row`
color for column/row points. The default values are "red" and "blue", respectively. Allowed values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the colors for row/column variables are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2, "coord"), x values("x") or y values("y")

`col.col.sup, col.row.sup`
colors for the supplementary column and row points, respectively.

`alpha.col, alpha.row`
controls the transparency of colors. The value can variate from 0 (total transparency) to 1 (no transparency). Default value is 1. Allowed values include also : "cos2", "contrib", "coord", "x" or "y" as for the arguments col.col and col.row.

`select.col, select.row`
a selection of columns/rows to be drawn. Allowed values are NULL or a list containing the arguments name, cos2 or contrib:

- name is a character vector containing column/row names to be drawn

- cos2 if cos2 is in [0, 1], ex: 0.6, then columns/rows with a cos2 > 0.6 are drawn. if cos2 > 1, ex: 5, then the top 5 columns/rows with the highest cos2 are drawn.
- contrib if contrib > 1, ex: 5, then the top 5 columns/rows with the highest contrib are drawn

`arrows` Vector of two logicals specifying if the plot should contain points (FALSE, default) or arrows (TRUE). First value sets the rows and the second value sets the columns.

`title` the title of the graph

Details

The default plot of CA is a "symmetric" plot in which both rows and columns are in principal coordinates. In this situation, it's not possible to interpret the distance between row points and column points. To overcome this problem, the simplest way is to make an asymmetric plot. This means that, the column profiles must be presented in row space or vice-versa. The allowed options for the argument map are:

- "rowprincipal" or "colprincipal": asymmetric plots with either rows in principal coordinates and columns in standard coordinates, or vice versa. These plots preserve row metric or column metric respectively.
- "symbiplot": Both rows and columns are scaled to have variances equal to the singular values (square roots of eigenvalues), which gives a symmetric biplot but does not preserve row or column metrics.
- "rowgab" or "colgab": Asymmetric maps, proposed by Gabriel & Odoroff (1990), with rows (respectively, columns) in principal coordinates and columns (respectively, rows) in standard coordinates multiplied by the mass of the corresponding point.
- "rowgreen" or "colgreen": The so-called contribution biplots showing visually the most contributing points (Greenacre 2006b). These are similar to "rowgab" and "colgab" except that the points in standard coordinates are multiplied by the square root of the corresponding masses, giving reconstructions of the standardized residuals.

Value

a ggplot2 plot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

See Also

`get_ca`, `fviz_pca`, `fviz_mca`

Examples

```
# Correspondence Analysis
# ++++++++++++++++++++++++++++++
# Install and load FactoMineR to compute CA
# install.packages("FactoMineR")

library("FactoMineR")
data(housetasks)
head(housetasks)
res.ca <- CA(housetasks, graph=FALSE)

# Graph of row variables
# +++++++++++++++++++++

# Default plot
fviz_ca_row(res.ca)

# Customize the plot
 # - Show text only: geom = "text" (to show point only: geom = "point")
 # - Change color: col.row ="#00AFBB"
 # - Change title and axis labels
 # - Change axis limits by specifying the min and max
 # - Change themes: http://www.sthda.com/english/wiki/ggplot2-themes

fviz_ca_row(res.ca, geom = "text", col.row = "#00AFBB") +
   labs(title = "CA", x = "Dim.1", y ="Dim.2" )+ # titles
   xlim(-1.3, 1.7) + ylim (-1.5, 1)+ # axis limits
   theme_minimal() # change theme


# Control automatically the color of row points
   # using the "cos2" or the contributions "contrib"
   # cos2 = the quality of the rows on the factor map

fviz_ca_row(res.ca, col.row = "cos2")+
theme_minimal()

# Change gradient color
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_ca_row(res.ca, col.row = "cos2", repel = TRUE) +
      scale_color_gradient2(low = "white", mid = "#2E9FDF",
      high = "#FC4E07", midpoint = 0.5, space = "Lab")+
theme_minimal()

# Color by the contributions
fviz_ca_row(res.ca, col.row = "contrib") +
      scale_color_gradient2(low = "white", mid = "#00AFBB",
      high="#E7B800", midpoint = 10, space = "Lab")+
theme_minimal()

# You can also control the transparency
# of the color by the "cos2" or "contrib"
```

```
fviz_ca_row(res.ca, alpha.row="contrib") +
     theme_minimal()

# Select and visualize some rows with select.row argument.
 # - Rows with cos2 >= 0.5: select.row = list(cos2 = 0.5)
 # - Top 7 rows according to the cos2: select.row = list(cos2 = 7)
 # - Top 7 contributing rows: select.row = list(contrib = 7)
 # - Select rows by names: select.row = list(name = c("Breakfeast", "Repairs", "Holidays"))

 # Example: Select the top 7 contributing rows
fviz_ca_row(res.ca, select.row = list(contrib = 7))


# Graph of column points
# +++++++++++++++++++++++++++

# Default plot
fviz_ca_col(res.ca, col.col = "red")+
theme_minimal()


# Control colors using their contributions
fviz_ca_col(res.ca, col.col = "contrib")+
 scale_color_gradient2(low = "white", mid = "blue",
           high = "red", midpoint = 25, space = "Lab") +
 theme_minimal()

# Select columns with select.col argument
   # You can select by contrib, cos2 and name
   # as previously described for ind
# Select the top 3 contributing columns
fviz_ca_col(res.ca, select.col = list(contrib = 3))


# Biplot of rows and columns
# ++++++++++++++++++++++++++
# Symetric Biplot of rows and columns
fviz_ca_biplot(res.ca)

# Asymetric biplot, use arrows for columns
fviz_ca_biplot(res.ca, map ="rowprincipal",
 arrow = c(FALSE, TRUE))

# Keep only the labels for row points
fviz_ca_biplot(res.ca, label ="row")

# Keep only labels for column points
fviz_ca_biplot(res.ca, label ="col")


# You can hide row or column points using
# invisible = "row" or invisible = "col", respectively
fviz_ca_biplot(res.ca, invisible ="row")
```

```
# Control automatically the color of rows using the cos2
fviz_ca_biplot(res.ca, col.row="cos2") +
       theme_minimal()

# Select the top 7 contributing rows
# And the top 3 columns
fviz_ca_biplot(res.ca,
               select.row = list(contrib = 7),
               select.col = list(contrib = 3))+
 theme_minimal()
```

`fviz_cluster` *Visualize Clustering Results*

Description

Provides ggplot2-based elegant visualization of partitioning methods including kmeans [stats package]; pam, clara and fanny [cluster package]; dbscan [fpc package]; Mclust [mclust package]; HCPC [FactoMineR]; hkmeans [factoextra]. Observations are represented by points in the plot, using principal components if ncol(data) > 2. An ellipse is drawn around each cluster.

Usage

```
fviz_cluster(object, data = NULL, stand = TRUE, geom = c("point", "text"),
  repel = FALSE, show.clust.cent = TRUE, frame = TRUE,
  frame.type = "convex", frame.level = 0.95, frame.alpha = 0.2,
  pointsize = 2, labelsize = 4, title = "Cluster plot",
  jitter = list(what = "label", width = NULL, height = NULL),
  outlier.color = "black", outlier.shape = 19)
```

Arguments

`object`	an object of class "partition" created by the functions pam(), clara() or fanny() in cluster package; "kmeans" [in stats package]; "dbscan" [in fpc package]; "Mclust" [in mclust]; "hkmeans", "eclust" [in factoextra]. Possible value are also any list object with data and cluster components (e.g.: object = list(data = mydata, cluster = myclust)).
`data`	the data that has been used for clustering. Required only when object is a class of kmeans or dbscan.
`stand`	logical value; if TRUE, data is standardized before principal component analysis
`geom`	a text specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "text"). Use "point" (to show only points); "text" to show only labels; c("point", "text") to show both types.
`repel`	a boolean, whether to use ggrepel to avoid overplotting text labels or not.

`show.clust.cent`
: logical; if TRUE, shows cluster centers

`frame`
: logical value; if TRUE, draws outline around points of each cluster

`frame.type`
: Character specifying frame type. Possible values are 'convex' or types supporeted by `ggplot2::stat_ellipse` including one of c("t", "norm", "euclid").

`frame.level`
: Passed for `ggplot2::stat_ellipse` 's level. Ignored in 'convex'. Default value is 0.95.

`frame.alpha`
: Alpha for frame specifying the transparency level of fill color.

`pointsize`
: the size of points

`labelsize`
: font size for the labels

`title`
: the title of the graph

`jitter`
: a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).
  - what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b".
  - width: degree of jitter in x direction
  - height: degree of jitter in y direction

`outlier.color, outlier.shape`
: the color and the shape of outliers. Outliers can be detected only in DBSCAN clustering.

Value

return a ggpplot.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_silhouette, hcut, hkmeans, eclust, fviz_dend

Examples

```
set.seed(123)

# Data preparation
# +++++++++++++++
data("iris")
head(iris)
# Remove species column (5) and scale the data
iris.scaled <- scale(iris[, -5])

# K-means clustering
# +++++++++++++++++++++
km.res <- kmeans(iris.scaled, 3, nstart = 25)
```

```
# Visualize kmeans clustering
# use repel = TRUE to avoid overplotting
fviz_cluster(km.res, iris[, -5], frame.type = "norm")


# Change the color and theme
fviz_cluster(km.res, iris[, -5]) +
 scale_color_brewer(palette = "Set2")+
 scale_fill_brewer(palette = "Set2") +
 theme_minimal()

 ## Not run:
# Show points only
fviz_cluster(km.res, iris[, -5], geom = "point")
# Show text only
fviz_cluster(km.res, iris[, -5], geom = "text")

# PAM clustering
# ++++++++++++++++++++
require(cluster)
pam.res <- pam(iris.scaled, 3)
 # Visualize pam clustering
fviz_cluster(pam.res, geom = "point", frame.type = "norm")

## End(Not run)

# Hierarchical clustering
# ++++++++++++++++++++++++
# Use hcut() which compute hclust and cut the tree
hc.cut <- hcut(iris.scaled, k = 3, hc_method = "complete")
# Visualize dendrogram
fviz_dend(hc.cut, show_labels = FALSE, rect = TRUE)
# Visualize cluster
fviz_cluster(hc.cut, frame.type = "convex")
```

`fviz_contrib` *Visualize the contributions of row/column elements*

Description

This function can be used to visualize the quality of representation (cos2) of rows/columns from the results of Principal Component Analysis (PCA), Correspondence Analysis (CA), Multiple Correspondence Analysis (MCA) and Multiple Factor Analysis (MFA) functions.

Usage

```
fviz_contrib(X, choice = c("row", "col", "var", "ind", "quanti.var",
  "quali.var", "group", "partial.axes"), axes = 1, fill = "steelblue",
  color = "steelblue", sort.val = c("desc", "asc", "none"), top = Inf)

fviz_pca_contrib(X, choice = c("var", "ind"), axes = 1,
  fill = "steelblue", color = "steelblue", sortcontrib = c("desc", "asc",
  "none"), top = Inf, ...)
```

Arguments

X	an object of class PCA, CA, MCA and MFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package].
choice	allowed values are "row" and "col" for CA; "var" and "ind" for PCA or MCA
axes	a numeric vector specifying the dimension(s) of interest.
fill	a fill color for the bar plot.
color	an outline color for the bar plot.
sort.val	a string specifying whether the value should be sorted. Allowed values are "none" (no sorting), "asc" (for ascending) or "desc" (for descending).
top	a numeric value specifying the number of top elements to be shown.
sortcontrib	see the argument sort.val
...	not used

Details

The function fviz_contrib() creates a barplot of row/column contributions. A reference dashed line is also shown on the barplot. This reference line corresponds to the expected value if the contribution where uniform.

For a given dimension, any row/column with a contribution above the reference line could be considered as important in contributing to the dimension.

Value

a ggplot2 plot

Functions

- `fviz_pca_contrib`: deprecated function. Use fviz_contrib()

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Principal component analysis
# ++++++++++++++++++++++++++
data(decathlon2)
decathlon2.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon2.active,  scale = TRUE)

# variable contributions on axis 1
fviz_contrib(res.pca, choice="var", axes = 1 )
# sorting
fviz_contrib(res.pca, choice="var", axes = 1,
           sort.val ="asc")

# select the top 7 contributing variables
fviz_contrib(res.pca, choice="var", axes = 1, top = 7 )

# Change theme and color
fviz_contrib(res.pca, choice="var", axes = 1,
         fill = "lightgray", color = "black") +
         theme_minimal() +
         theme(axis.text.x = element_text(angle=45))

# Variable contributions on axis 2
fviz_contrib(res.pca, choice="var", axes = 2)
# Variable contributions on axes 1 + 2
fviz_contrib(res.pca, choice="var", axes = 1:2)

# Contributions of individuals on axis 1
fviz_contrib(res.pca, choice="ind", axes = 1)

# Correspondence Analysis
# ++++++++++++++++++++++++++
# Install and load FactoMineR to compute CA
# install.packages("FactoMineR")
library("FactoMineR")
data("housetasks")
res.ca <- CA(housetasks, graph = FALSE)

# Visualize row contributions on axes 1
fviz_contrib(res.ca, choice ="row", axes = 1)
# Visualize row contributions on axes 1 + 2
fviz_contrib(res.ca, choice ="row", axes = 1:2)
# Visualize column contributions on axes 1
fviz_contrib(res.ca, choice ="col", axes = 1)

# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2,
              quali.sup = 3:4, graph=FALSE)
```

```
# Visualize individual contributions on axes 1
fviz_contrib(res.mca, choice ="ind", axes = 1)
# Select the top 20
fviz_contrib(res.mca, choice ="ind", axes = 1, top = 20)
# Visualize variable categorie contributions on axes 1
fviz_contrib(res.mca, choice ="var", axes = 1)

# Multiple Factor Analysis
# ++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
               name.group=c("desc","desc2","symptom","eat"),
               num.group.sup=1:2, graph=FALSE)

# Visualize individual contributions on axes 1
fviz_contrib(res.mfa, choice ="ind", axes = 1)
# Select the top 20
fviz_contrib(res.mfa, choice ="ind", axes = 1, top = 20)
# Visualize catecorical variable categorie contributions on axes 1
fviz_contrib(res.mfa, choice ="quali.var", axes = 1)
```

`fviz_cos2` *Visualize the quality of representation of rows/columns*

Description

This function can be used to visualize the quality of representation (cos2) of rows/columns from the results of Principal Component Analysis (PCA), Correspondence Analysis (CA), Multiple Correspondence Analysis (MCA) and Multiple Factor Analysis (MFA) functions.

Usage

```
fviz_cos2(X, choice = c("row", "col", "var", "ind", "quanti.var", "quali.var",
  "group"), axes = 1, fill = "steelblue", color = "steelblue",
  sort.val = c("desc", "asc", "none"), top = Inf)
```

Arguments

`X`	an object of class PCA, CA, MCA and MFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package].
`choice`	allowed values are "row" and "col" for CA; "var" and "ind" for PCA or MCA
`axes`	a numeric vector specifying the dimension(s) of interest.
`fill`	a fill color for the bar plot.
`color`	an outline color for the bar plot.

sort.val	a string specifying whether the value should be sorted. Allowed values are "none" (no sorting), "asc" (for ascending) or "desc" (for descending).
top	a numeric value specifying the number of top elements to be shown.

Value

a ggplot2 plot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Principal component analysis
# ++++++++++++++++++++++++++
data(decathlon2)
decathlon2.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon2.active,  scale = TRUE)

# variable cos2 on axis 1
fviz_cos2(res.pca, choice="var", axes = 1 )
# sorting
fviz_cos2(res.pca, choice="var", axes = 1,
           sort.val ="asc")

# select the top 7 contributing variables
fviz_cos2(res.pca, choice="var", axes = 1, top = 7 )

# Change theme and color
fviz_cos2(res.pca, choice="var", axes = 1,
         fill = "lightgray", color = "black") +
         theme_minimal() +
         theme(axis.text.x = element_text(angle=45))

# Variable cos2 on axis 2
fviz_cos2(res.pca, choice="var", axes = 2)
# Variable cos2 on axes 1 + 2
fviz_cos2(res.pca, choice="var", axes = 1:2)

# cos2 of individuals on axis 1
fviz_cos2(res.pca, choice="ind", axes = 1)

# Correspondence Analysis
# ++++++++++++++++++++++++++
# Install and load FactoMineR to compute CA
# install.packages("FactoMineR")
```

```
library("FactoMineR")
data("housetasks")
res.ca <- CA(housetasks, graph = FALSE)

# Visualize row cos2 on axes 1
fviz_cos2(res.ca, choice ="row", axes = 1)
# Visualize row cos2 on axes 1 + 2
fviz_cos2(res.ca, choice ="row", axes = 1:2)
# Visualize column cos2 on axes 1
fviz_cos2(res.ca, choice ="col", axes = 1)

# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2,
               quali.sup = 3:4, graph=FALSE)

# Visualize individual cos2 on axes 1
fviz_cos2(res.mca, choice ="ind", axes = 1)
# Select the top 20
fviz_cos2(res.mca, choice ="ind", axes = 1, top = 20)
# Visualize variable categorie cos2 on axes 1
fviz_cos2(res.mca, choice ="var", axes = 1)

# Multiple Factor Analysis
# ++++++++++++++++++++++++
library(FactoMineR)
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
               name.group=c("desc","desc2","symptom","eat"),
               num.group.sup=1:2, graph=FALSE)
# Visualize individual cos2 on axes 1
fviz_cos2(res.mfa, choice ="ind", axes = 1)
# Select the top 20
fviz_cos2(res.mfa, choice ="ind", axes = 1, top = 20)
# Visualize catecorical variable categorie cos2 on axes 1
fviz_cos2(res.mfa, choice ="quali.var", axes = 1)
```

`fviz_dend` *Enhanced Visualization of Dendrogram*

Description

Enhanced visualization of dendrogram.

Usage

```
fviz_dend(x, k = NULL, k_colors = NULL, show_labels = TRUE,
  color_labels_by_k = FALSE, label_cols = NULL, type = c("rectangle",
  "triangle"), rect = FALSE, rect_border = "gray", rect_lty = 2,
  rect_lwd = 1.5, cex = 0.8, main = "Cluster Dendrogram", xlab = "",
  ylab = "Height", ...)
```

Arguments

x	an object of class dendrogram, hclust, agnes, diana, hcut or hkmeans.
k	the number of groups for cutting the tree.
k_colors	a vector containing colors to be used for the groups. It should contains k number of colors.
show_labels	a logical value. If TRUE, leaf labels are shown. Default value is TRUE.
color_labels_by_k	logical value. If TRUE, labels are colored automatically by group when k != NULL.
label_cols	a vector containing the colors for labels.
type	type of plot. Allowed values are one of "rectangle" or "triangle"
rect	logical value specifying whether to add a rectangle around groups. Used only when k != NULL.
rect_border, rect_lty, rect_lwd	border color, line type and line width for rectangles
cex	size of labels
main, xlab, ylab	main and axis titles
...	other arguments to be passed to the function plot.dendrogram()

Examples

```
# Load and scale the data
data(USArrests)
df <- scale(USArrests)

# Hierarchical clustering
res.hc <- hclust(dist(df))

# Default plot
fviz_dend(res.hc)

# Cut the tree
fviz_dend(res.hc, cex = 0.5, k = 4, color_labels_by_k = TRUE)

# Don't color labels, add rectangles
fviz_dend(res.hc, cex = 0.5, k = 4,
 color_labels_by_k = FALSE, rect = TRUE)
```

```
# Triangle
fviz_dend(res.hc, cex = 0.5, k = 4, type = "triangle")

# Change the color of tree using black color for all groups
# Change rectangle border colors
fviz_dend(res.hc, rect = TRUE, k_colors ="black",
rect_border = 2:5, rect_lty = 1)

# Customized color for groups
fviz_dend(res.hc, k = 4,
 k_colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A"))


 # Color labels using k-means clusters
 km.clust <- kmeans(df, 4)$cluster
 fviz_dend(res.hc, k = 4,
   k_colors = c("blue", "green3", "red", "black"),
   label_cols =  km.clust[res.hc$order], cex = 0.6)
```

`fviz_hmfa` *Visualize Hierarchical Multiple Factor Analysis*

Description

Graph of individuals/quantitative variables/qualitative variables/group/partial axes from the output of Hierarchical Multiple Factor Analysis (HMFA).

- fviz_hmfa_ind(): Graph of individuals
- fviz_hmfa_ind_starplot(): Graph of partial individuals
- fviz_hmfa_quanti_var(): Graph of quantitative variables
- fviz_hmfa_quali_var(): Graph of qualitative variables
- fviz_hmfa_quali_biplot(): Biplot of individuals and qualitative variables
- fviz_hmfa_group(): Graph of the groups representation
- fviz_hmfa(): An alias of fviz_hmfa_ind()

Usage

```
fviz_hmfa_ind(X, axes = c(1, 2), geom = c("point", "text"), label = "all",
  invisible = "none", labelsize = 4, pointsize = 2, habillage = "none",
  addEllipses = FALSE, ellipse.level = 0.95, ellipse.type = "norm",
  ellipse.alpha = 0.1, col.ind = "blue", col.ind.sup = "darkblue",
  alpha.ind = 1, shape.ind = 19, repel = FALSE,
```

```
  axes.linetype = "dashed", select.ind = list(name = NULL, cos2 = NULL,
  contrib = NULL), title = "Individuals factor map - HMFA",
  jitter = list(what = "label", width = NULL, height = NULL), ...)

fviz_hmfa_quanti_var(X, axes = c(1, 2), geom = c("arrow", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.var = "red", alpha.var = 1, shape.var = 17,
  col.quanti.sup = "blue", col.quali.sup = "darkgreen",
  col.circle = "grey70", select.var = list(name = NULL, cos2 = NULL, contrib
  = NULL), axes.linetype = "dashed",
  title = "Quantitative Variable categories - MFA", repel = FALSE,
  jitter = list(what = "label", width = NULL, height = NULL))

fviz_hmfa_quali_var(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.var = "red", alpha.var = 1, shape.var = 17,
  col.quanti.sup = "blue", col.quali.sup = "darkgreen", repel = FALSE,
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  axes.linetype = "dashed", title = "Qualitative Variable categories - MFA",
  jitter = list(what = "label", width = NULL, height = NULL))

fviz_hmfa_quali_biplot(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  col.ind = "blue", col.ind.sup = "darkblue", alpha.ind = 1,
  col.var = "red", alpha.var = 1, col.quanti.sup = "blue",
  col.quali.sup = "darkgreen", axes.linetype = "dashed", shape.ind = 19,
  shape.var = 17, select.var = list(name = NULL, cos2 = NULL, contrib =
  NULL), select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
  arrows = c(FALSE, FALSE), repel = FALSE,
  title = "HMFA factor map - Biplot", jitter = list(what = "label", width =
  NULL, height = NULL), ...)

fviz_hmfa_ind_starplot(X, axes = c(1, 2), geom = c("point", "text"),
  group.names = NULL, label = "all", invisible = "none",
  legend.partial.title = NULL, labelsize = 4, pointsize = 2,
  linesize = 0.5, repel = FALSE, habillage = "none",
  addEllipses = FALSE, ellipse.level = 0.95, ellipse.type = "norm",
  ellipse.alpha = 0.1, col.ind = "black", col.ind.sup = "darkblue",
  col.partial = "black", alpha.ind = 1, shape.ind = 19,
  alpha.partial = 1, node.level = 1, select.ind = list(name = NULL, cos2 =
  NULL, contrib = NULL), select.partial = list(name = NULL, cos2 = NULL,
  contrib = NULL), axes.linetype = "dashed",
  title = "Individuals factor map - MFA", jitter = list(what = "label",
  width = NULL, height = NULL), ...)

fviz_hmfa_group(X, axes = c(1, 2), geom = c("point", "text"),
  alpha.group = 1, shape.group = 17, label = "all", invisible = "none",
```

```
  labelsize = 4, pointsize = 2, col.group = "blue",
  col.group.sup = "darkgreen", repel = FALSE, select.group = list(name =
  NULL, cos2 = NULL, contrib = NULL), title = "MFA - Groups Representations",
  jitter = list(what = "label", width = NULL, height = NULL), ...)

fviz_hmfa(X, ...)
```

Arguments

X	an object of class HMFA [FactoMineR].
axes	a numeric vector of length 2 specifying the dimensions to be plotted.
geom	a text specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "arrow", "text"). Use "point" (to show only points); "text" to show only labels; c("point", "text") or c("arrow", "text") to show both types.
label	a text specifying the elements to be labelled. Default value is "all". Allowed values are "none" or the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup"). "ind" can be used to label only active individuals. "ind.sup" is for supplementary individuals. "var" is for active variable categories. "quali.sup" is for supplementary qualitative variable categories. "quanti.sup" is for quantitative supplementary variables.
invisible	a text specifying the elements to be hidden on the plot. Default value is "none". Allowed values are the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup").
labelsize	font size for the labels
pointsize	the size of points
habillage	an optional factor variable for coloring the observations by groups. Default value is "none". If X is an MFA object from FactoMineR package, habillage can also specify the index of the factor variable in the data.
addEllipses	logical value. If TRUE, draws ellipses around the individuals when habillage != "none".
ellipse.level	the size of the concentration ellipse in normal probability.
ellipse.type	Character specifying frame type. Possible values are 'convex' or types supporeted by stat_ellipse including one of c("t", "norm", "euclid").
ellipse.alpha	Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.
col.ind, col.partial, col.var, col.group	color for individuals, partial individuals, variables, groups and axes, respectively. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the colors for individuals/variables are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use automatic coloring (by cos2, contrib, ...), make sure that habillage ="none".
col.ind.sup	color for supplementary individuals

`alpha.ind, alpha.partial, alpha.var, alpha.group`
: controls the transparency of individual, partial individual, variable, group and axes colors, respectively. The value can variate from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the transparency for individual/variable colors are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use this, make sure that habillage ="none".

`shape.ind, shape.var, shape.group`
: point shapes of individuals, variables, groups and axes

`repel`
: a boolean, whether to use ggrepel to avoid overplotting text labels or not.

`axes.linetype`
: linetype of x and y axes.

`select.ind, select.partial, select.var, select.group`
: a selection of individuals/partial individuals/ variables/groups/axes to be drawn. Allowed values are NULL or a list containing the arguments name, cos2 or contrib:

  - name is a character vector containing individuals/variables to be drawn
  - cos2 if cos2 is in [0, 1], ex: 0.6, then individuals/variables with a cos2 > 0.6 are drawn. if cos2 > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn.
  - contrib if contrib > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn

`title`
: the title of the graph

`jitter`
: a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).

  - what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b"
  - width: degree of jitter in x direction
  - height: degree of jitter in y direction

`...`
: Arguments to be passed to the function fviz_mfa_quali_biplot()

`col.quanti.sup, col.quali.sup, col.group.sup`
: a color for the quantitative/qualitative supplementary variables.

`col.circle`
: a color for the correlation circle.

`arrows`
: Vector of two logicals specifying if the plot should contain points (FALSE, default) or arrows (TRUE). First value sets the rows and the second value sets the columns.

`group.names`
: a vector containing the name of the groups (by default, NULL and the group are named group.1, group.2 and so on).

`legend.partial.title`
: the title of the partal groups legend.

`linesize`
: size of partial point connecting line.

`node.level`
: a single number indicating the HMFA node level to plot.

Value

a ggplot2 plot

Author(s)

Fabian Mundt <f.mundt@inventionate.de>

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Hierarchical Multiple Factor Analysis
# ++++++++++++++++++++++++
# Install and load FactoMineR to compute MFA
# install.packages("FactoMineR")
library("FactoMineR")
data(wine)
hierar <- list(c(2,5,3,10,9,2), c(4,2))
res.hmfa <- HMFA(wine, H = hierar, type=c("n",rep("s",5)), graph = FALSE)

# Graph of individuals
# ++++++++++++++++++++
# Default plot
# Color of individuals: col.ind = "#2E9FDF"
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_hmfa_ind(res.hmfa, repel = TRUE, col.ind = "#2E9FDF")

## Not run:
# 1. Control automatically the color of individuals
   # using the "cos2" or the contributions "contrib"
   # cos2 = the quality of the individuals on the factor map
# 2. To keep only point or text use geom = "point" or geom = "text".
# 3. Change themes: http://www.sthda.com/english/wiki/ggplot2-themes

fviz_hmfa_ind(res.hmfa, col.ind="cos2")+
theme_minimal()

## End(Not run)

# Color individuals by groups, add concentration ellipses
# Remove labels: label = "none".
grp <- as.factor(wine[,1])
p <- fviz_hmfa_ind(res.hmfa, label="none", habillage=grp,
       addEllipses=TRUE, ellipse.level=0.95)+
       theme_minimal()
print(p)

## Not run:
```

```
# Change group colors using RColorBrewer color palettes
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_brewer(palette="Paired") +
    scale_fill_brewer(palette="Paired") +
     theme_minimal()

## End(Not run)

# Change group colors manually
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
 scale_fill_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
 theme_minimal()

## Not run:
# Select and visualize some individuals (ind) with select.ind argument.
 # - ind with cos2 >= 0.1: select.ind = list(cos2 = 0.1)
 # - Top 20 ind according to the cos2: select.ind = list(cos2 = 20)
 # - Top 20 contributing individuals: select.ind = list(contrib = 20)
 # - Select ind by names: select.ind = list(name = c("1VAU", "1FON") )

# Example: Select the top 10 according to the cos2
fviz_hmfa_ind(res.hmfa, select.ind = list(cos2 = 100))

## End(Not run)


# Graph of qantitative variable categories
# ++++++++++++++++++++++++++++++++++++++++
data(wine)
hierar <- list(c(2,5,3,10,9,2), c(4,2))
res.hmfa <- HMFA(wine, H = hierar, type=c("n",rep("s",5)), graph = FALSE)

# Plot
# Control variable colors using their contributions
fviz_hmfa_quanti_var(res.hmfa, col.var = "contrib")+
 scale_color_gradient2(low = "white", mid = "blue",
           high = "red", midpoint = 12) +
 theme_minimal()

## Not run:
# Select variables with select.var argument
   # You can select by contrib, cos2 and name
   # as previously described for ind
# Select the top 10 contributing variables
fviz_hmfa_quanti_var(res.hmfa, select.var = list(contrib = 10))

## End(Not run)

# Graph of categorical variable categories
# ++++++++++++++++++++++++++++++++++++++++
data(poison)
hierar <- list(c(2,2,5,6), c(1,3))
```

```
res.hmfa <- HMFA(poison, H = hierar, type=c("s","n","n","n"), graph = FALSE)

# Default plot
fviz_hmfa_quali_var(res.hmfa, col.var = "contrib")+
 theme_minimal()


# Biplot of categorical variable categories and individuals
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
grp <- as.factor(poison[, "Vomiting"])
# Use repel = TRUE to avoid overplotting
fviz_hmfa_quali_biplot(res.hmfa, col.var = "#E7B800", repel = FALSE,
    habillage = grp, addEllipses = TRUE)+
theme_minimal()


# Graph of partial individuals (starplot)
# +++++++++++++++++++++++++++++++++++++++
fviz_hmfa_ind_starplot(res.hmfa, col.partial = "group.name")+
scale_color_brewer(palette = "Dark2")
 theme_minimal()


## Not run:
# Select the partial points of the top 5
# contributing individuals
fviz_hmfa_ind_starplot(res.hmfa,
                       select.partial = list(contrib = 2))+
                       theme_minimal()

# Change colours of star segments
fviz_hmfa_ind_starplot(res.hmfa, select.partial = list(contrib = 5),
                       col.partial = "group.name") +
                       scale_color_brewer(palette = "Dark2") +
                       theme_minimal()

## End(Not run)


# Graph of groups (correlation square)
# ++++++++++++++++++++++++++++++++++++
fviz_hmfa_group(res.hmfa)
```

---

fviz_mca *Visualize Multiple Correspondence Analysis*

Description

Multiple Correspondence Analysis (MCA) is an extension of simple CA to analyse a data table containing more than two categorical variables. fviz_mca() provides ggplot2-based elegant visual-

ization of MCA outputs from the R functions: MCA [in FactoMineR], and acm [in ade4]. Read more: Multiple Correspondence Analysis Essentials.

- fviz_mca_ind(): Graph of individuals
- fviz_mca_var(): Graph of variables
- fviz_mca_biplot(): Biplot of individuals and variables
- fviz_mca(): An alias of fviz_mca_biplot()

Usage

```
fviz_mca_ind(X, axes = c(1, 2), geom = c("point", "text"), label = "all",
  invisible = "none", labelsize = 4, pointsize = 2, repel = FALSE,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  ellipse.type = "norm", ellipse.alpha = 0.1, col.ind = "blue",
  col.ind.sup = "darkblue", alpha.ind = 1, shape.ind = 19,
  axes.linetype = "dashed", select.ind = list(name = NULL, cos2 = NULL,
  contrib = NULL), map = "symmetric",
  title = "Individuals factor map - MCA", jitter = list(what = "label",
  width = NULL, height = NULL), ...)

fviz_mca_var(X, axes = c(1, 2), geom = c("point", "text"), label = "all",
  invisible = "none", labelsize = 4, pointsize = 2, col.var = "red",
  alpha.var = 1, shape.var = 17, col.quanti.sup = "blue",
  col.quali.sup = "darkgreen", repel = FALSE,
  title = "Variable categories- MCA", select.var = list(name = NULL, cos2 =
  NULL, contrib = NULL), axes.linetype = "dashed", map = "symmetric",
  jitter = list(what = "label", width = NULL, height = NULL))

fviz_mca_biplot(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  col.ind = "blue", col.ind.sup = "darkblue", alpha.ind = 1,
  col.var = "red", alpha.var = 1, col.quanti.sup = "blue",
  col.quali.sup = "darkgreen", repel = FALSE, shape.ind = 19,
  shape.var = 17, axes.linetype = "dashed", select.var = list(name = NULL,
  cos2 = NULL, contrib = NULL), select.ind = list(name = NULL, cos2 = NULL,
  contrib = NULL), map = "symmetric", arrows = c(FALSE, FALSE),
  title = "MCA factor map - Biplot", jitter = list(what = "label", width =
  NULL, height = NULL), ...)

fviz_mca(X, ...)
```

Arguments

X	an object of class MCA [FactoMineR], acm [ade4].
axes	a numeric vector of length 2 specifying the dimensions to be plotted.
geom	a text specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "arrow", "text"). Use "point" (to show only points);

"text" to show only labels; c("point", "text") or c("arrow", "text") to show both types.

`label` a text specifying the elements to be labelled. Default value is "all". Allowed values are "none" or the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup"). "ind" can be used to label only active individuals. "ind.sup" is for supplementary individuals. "var" is for active variable categories. "quali.sup" is for supplementary qualitative variable categories. "quanti.sup" is for quantitative supplementary variables.

`invisible` a text specifying the elements to be hidden on the plot. Default value is "none". Allowed values are the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup").

`labelsize` font size for the labels

`pointsize` the size of points

`repel` a boolean, whether to use ggrepel to avoid overplotting text labels or not.

`habillage` an optional factor variable for coloring the observations by groups. Default value is "none". If X is an MCA object from FactoMineR package, habillage can also specify the index of the factor variable in the data.

`addEllipses` logical value. If TRUE, draws ellipses around the individuals when habillage != "none".

`ellipse.level` the size of the concentration ellipse in normal probability.

`ellipse.type` Character specifying frame type. Possible values are 'convex' or types supporeted by `stat_ellipse` including one of c("t", "norm", "euclid").

`ellipse.alpha` Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.

`col.ind, col.var`
color for individuals and variables, respectively. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the colors for individuals/variables are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use automatic coloring (by cos2, contrib, ...), make sure that habillage ="none".

`col.ind.sup` color for supplementary individuals

`alpha.ind, alpha.var`
controls the transparency of individual and variable colors, respectively. The value can variate from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the transparency for individual/variable colors are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use this, make sure that habillage ="none".

`shape.ind, shape.var`
point shapes of individuals and variables.

`axes.linetype` linetype of x and y axes.

`select.ind, select.var`
: a selection of individuals/variables to be drawn. Allowed values are NULL or a list containing the arguments name, cos2 or contrib:
  - name is a character vector containing individuals/variables to be drawn
  - cos2 if cos2 is in [0, 1], ex: 0.6, then individuals/variables with a cos2 > 0.6 are drawn. if cos2 > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn.
  - contrib if contrib > 1, ex: 5, then the top 5 individuals/variables with the highest contrib are drawn

`map`
: character string specifying the map type. Allowed options include: "symmetric", "rowprincipal", "colprincipal", "symbiplot", "rowgab", "colgab", "rowgreen" and "colgreen". See details

`title`
: the title of the graph

`jitter`
: a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).
  - what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b"
  - width: degree of jitter in x direction
  - height: degree of jitter in y direction

`...`
: Arguments to be passed to the function fviz_mca_biplot()

`col.quanti.sup, col.quali.sup`
: a color for the quantitative/qualitative supplementary variables.

`arrows`
: Vector of two logicals specifying if the plot should contain points (FALSE, default) or arrows (TRUE). First value sets the rows and the second value sets the columns.

Details

The default plot of MCA is a "symmetric" plot in which both rows and columns are in principal coordinates. In this situation, it's not possible to interpret the distance between row points and column points. To overcome this problem, the simplest way is to make an asymmetric plot. This means that, the column profiles must be presented in row space or vice-versa. The allowed options for the argument map are:

- "rowprincipal" or "colprincipal": asymmetric plots with either rows in principal coordinates and columns in standard coordinates, or vice versa. These plots preserve row metric or column metric respectively.
- "symbiplot": Both rows and columns are scaled to have variances equal to the singular values (square roots of eigenvalues), which gives a symmetric biplot but does not preserve row or column metrics.
- "rowgab" or "colgab": Asymmetric maps, proposed by Gabriel & Odoroff (1990), with rows (respectively, columns) in principal coordinates and columns (respectively, rows) in standard coordinates multiplied by the mass of the corresponding point.
- "rowgreen" or "colgreen": The so-called contribution biplots showing visually the most contributing points (Greenacre 2006b). These are similar to "rowgab" and "colgab" except that the points in standard coordinates are multiplied by the square root of the corresponding masses, giving reconstructions of the standardized residuals.

Value

a ggplot2 plot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

get_mca, fviz_pca, fviz_ca, fviz_mfa, fviz_hmfa

Examples

```
# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++
# Install and load FactoMineR to compute MCA
# install.packages("FactoMineR")
library("FactoMineR")
data(poison)
poison.active <- poison[1:55, 5:15]
head(poison.active)
res.mca <- MCA(poison.active, graph=FALSE)

# Graph of individuals
# +++++++++++++++++++++

# Default Plot
# Color of individuals: col.ind = "steelblue"
fviz_mca_ind(res.mca, col.ind = "steelblue")

# 1. Control automatically the color of individuals
   # using the "cos2" or the contributions "contrib"
   # cos2 = the quality of the individuals on the factor map
# 2. To keep only point or text use geom = "point" or geom = "text".
# 3. Change themes: http://www.sthda.com/english/wiki/ggplot2-themes

fviz_mca_ind(res.mca, col.ind = "cos2", repel = TRUE)+
theme_minimal()

## Not run:
# You can also control the transparency
# of the color by the cos2
fviz_mca_ind(res.mca, alpha.ind="cos2") +
     theme_minimal()

## End(Not run)

# Color individuals by groups, add concentration ellipses
# Remove labels: label = "none".
grp <- as.factor(poison.active[, "Vomiting"])
p <- fviz_mca_ind(res.mca, label="none", habillage=grp,
       addEllipses=TRUE, ellipse.level=0.95)
```

```
print(p)


# Change group colors using RColorBrewer color palettes
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_brewer(palette="Dark2") +
    scale_fill_brewer(palette="Dark2") +
     theme_minimal()

# Change group colors manually
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_manual(values=c("#999999", "#E69F00"))+
 scale_fill_manual(values=c("#999999", "#E69F00"))+
 theme_minimal()


# Select and visualize some individuals (ind) with select.ind argument.
 # - ind with cos2 >= 0.4: select.ind = list(cos2 = 0.4)
 # - Top 20 ind according to the cos2: select.ind = list(cos2 = 20)
 # - Top 20 contributing individuals: select.ind = list(contrib = 20)
 # - Select ind by names: select.ind = list(name = c("44", "38", "53",  "39") )

# Example: Select the top 40 according to the cos2
fviz_mca_ind(res.mca, select.ind = list(cos2 = 20))


# Graph of variable categories
# +++++++++++++++++++++++++++++
# Default plot: use repel = TRUE to avoid overplotting
fviz_mca_var(res.mca, col.var = "#FC4E07")+
theme_minimal()

# Control variable colors using their contributions
# use repel = TRUE to avoid overplotting
fviz_mca_var(res.mca, col.var = "contrib")+
 scale_color_gradient2(low="white", mid="blue",
           high="red", midpoint=2, space = "Lab") +
 theme_minimal()


# Select variables with select.var argument
   # You can select by contrib, cos2 and name
   # as previously described for ind
# Select the top 10 contributing variables
fviz_mca_var(res.mca, select.var = list(contrib = 10))

# Biplot
# ++++++++++++++++++++++++++
grp <- as.factor(poison.active[, "Vomiting"])
fviz_mca_biplot(res.mca, repel = TRUE, col.var = "#E7B800",
 habillage = grp, addEllipses = TRUE, ellipse.level = 0.95)+
 theme_minimal()
```

```
 ## Not run:
# Keep only the labels for variable categories:
fviz_mca_biplot(res.mca, label ="var")

# Keep only labels for individuals
fviz_mca_biplot(res.mca, label ="ind")

# Hide variable categories
fviz_mca_biplot(res.mca, invisible ="var")

# Hide individuals
fviz_mca_biplot(res.mca, invisible ="ind")

# Control automatically the color of individuals using the cos2
fviz_mca_biplot(res.mca, label ="var", col.ind="cos2") +
       theme_minimal()

# Change the color by groups, add ellipses
fviz_mca_biplot(res.mca, label="var", col.var ="blue",
   habillage=grp, addEllipses=TRUE, ellipse.level=0.95) +
   theme_minimal()

# Select the top 30 contributing individuals
# And the top 10 variables
fviz_mca_biplot(res.mca,
               select.ind = list(contrib = 30),
               select.var = list(contrib = 10))

## End(Not run)
```

`fviz_mfa` *Visualize Multiple Factor Analysis*

Description

Graph of individuals/quantitative variables/qualitative variables/group/partial axes from the output of Multiple Factor Analysis (MFA).

- fviz_mfa_ind(): Graph of individuals
- fviz_mfa_ind_starplot(): Star graph of individuals
- fviz_mfa_quanti_var(): Graph of quantitative variables
- fviz_mfa_quali_var(): Graph of qualitative variables
- fviz_mfa_quali_biplot(): Biplot of individuals and qualitative variables
- fviz_mfa_group(): Graph of the groups representation
- fviz_mfa_axes(): Graph of partial axes
- fviz_mfa(): An alias of fviz_mfa_ind_starplot()

Usage

```
fviz_mfa_ind(X, axes = c(1, 2), geom = c("point", "text"), label = "all",
  invisible = "none", labelsize = 4, pointsize = 2, habillage = "none",
  addEllipses = FALSE, ellipse.level = 0.95, ellipse.type = "norm",
  ellipse.alpha = 0.1, col.ind = "blue", col.ind.sup = "darkblue",
  alpha.ind = 1, shape.ind = 19, repel = FALSE,
  axes.linetype = "dashed", select.ind = list(name = NULL, cos2 = NULL,
  contrib = NULL), title = "Individuals factor map - MFA",
  jitter = list(what = "label", width = NULL, height = NULL), ...)

fviz_mfa_quanti_var(X, axes = c(1, 2), geom = c("arrow", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.var = "red", alpha.var = 1, shape.var = 17,
  col.quanti.sup = "blue", col.quali.sup = "darkgreen",
  col.circle = "grey70", select.var = list(name = NULL, cos2 = NULL, contrib
  = NULL), axes.linetype = "dashed",
  title = "Quantitative Variable categories - MFA", repel = FALSE,
  jitter = list(what = "label", width = NULL, height = NULL))

fviz_mfa_quali_var(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  col.var = "red", alpha.var = 1, shape.var = 17,
  col.quanti.sup = "blue", col.quali.sup = "darkgreen", repel = FALSE,
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  axes.linetype = "dashed", title = "Qualitative Variable categories - MFA",
  jitter = list(what = "label", width = NULL, height = NULL))

fviz_mfa_quali_biplot(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  col.ind = "blue", col.ind.sup = "darkblue", alpha.ind = 1,
  col.var = "red", alpha.var = 1, col.quanti.sup = "blue",
  col.quali.sup = "darkgreen", shape.ind = 19, shape.var = 17,
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
  axes.linetype = "dashed", title = "MFA factor map - Biplot",
  arrows = c(FALSE, FALSE), repel = FALSE, jitter = list(what = "label",
  width = NULL, height = NULL), ...)

fviz_mfa_ind_starplot(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", legend.partial.title = NULL,
  labelsize = 4, pointsize = 2, linesize = 0.5, repel = FALSE,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  ellipse.type = "norm", ellipse.alpha = 0.1, col.ind = "black",
  col.ind.sup = "darkblue", col.partial = "black", alpha.ind = 1,
  shape.ind = 19, alpha.partial = 1, select.ind = list(name = NULL, cos2 =
  NULL, contrib = NULL), select.partial = list(name = NULL, cos2 = NULL,
  contrib = NULL), axes.linetype = "dashed",
```

```
  title = "Individuals factor map - MFA", jitter = list(what = "label",
  width = NULL, height = NULL), ...)

fviz_mfa_group(X, axes = c(1, 2), geom = c("point", "text"),
  alpha.group = 1, shape.group = 17, label = "all", invisible = "none",
  labelsize = 4, pointsize = 2, col.group = "blue",
  col.group.sup = "darkgreen", repel = FALSE, select.group = list(name =
  NULL, cos2 = NULL, contrib = NULL), title = "MFA - Groups Representations",
  jitter = list(what = "label", width = NULL, height = NULL), ...)

fviz_mfa_axes(X, axes = c(1, 2), geom = c("arrow", "text"), label = "all",
  invisible = "none", labelsize = 4, pointsize = 2, col.axes = "red",
  alpha.axes = 1, col.circle = "grey70", select.axes = list(name = NULL,
  contrib = NULL), axes.linetype = "dashed",
  title = "MFA - Partial Axes Representations", arrows = c(FALSE, FALSE),
  repel = FALSE, jitter = list(what = "label", width = NULL, height = NULL),
  ...)

fviz_mfa(X, ...)
```

Arguments

X	an object of class MFA [FactoMineR].
axes	a numeric vector of length 2 specifying the dimensions to be plotted.
geom	a text specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "arrow", "text"). Use "point" (to show only points); "text" to show only labels; c("point", "text") or c("arrow", "text") to show both types.
label	a text specifying the elements to be labelled. Default value is "all". Allowed values are "none" or the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup"). "ind" can be used to label only active individuals. "ind.sup" is for supplementary individuals. "var" is for active variable categories. "quali.sup" is for supplementary qualitative variable categories. "quanti.sup" is for quantitative supplementary variables.
invisible	a text specifying the elements to be hidden on the plot. Default value is "none". Allowed values are the combination of c("ind", "ind.sup","var", "quali.sup", "quanti.sup").
labelsize	font size for the labels
pointsize	the size of points
habillage	an optional factor variable for coloring the observations by groups. Default value is "none". If X is an MFA object from FactoMineR package, habillage can also specify the index of the factor variable in the data.
addEllipses	logical value. If TRUE, draws ellipses around the individuals when habillage != "none".
ellipse.level	the size of the concentration ellipse in normal probability.

`ellipse.type` Character specifying frame type. Possible values are 'convex' or types supporeted by `stat_ellipse` including one of c("t", "norm", "euclid").

`ellipse.alpha` Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.

`col.ind, col.partial, col.var, col.group, col.group.sup, col.axes`
color for individuals, partial individuals, variables, groups and axes, respectively. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the colors for individuals/variables are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use automatic coloring (by cos2, contrib, ...), make sure that habillage ="none".

`col.ind.sup` color for supplementary individuals

`alpha.ind, alpha.partial, alpha.var, alpha.group, alpha.axes`
controls the transparency of individual, partial individual, variable, group and axes colors, respectively. The value can variate from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the transparency for individual/variable colors are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2 + y^2 , "coord"), x values("x") or y values("y"). To use this, make sure that habillage ="none".

`shape.ind, shape.var, shape.group`
point shapes of individuals, variables, groups and axes

`repel` a boolean, whether to use ggrepel to avoid overplotting text labels or not.

`axes.linetype` linetype of x and y axes.

`select.ind, select.partial, select.var, select.group, select.axes`
a selection of individuals/partial individuals/ variables/groups/axes to be drawn. Allowed values are NULL or a list containing the arguments name, cos2 or contrib:

- name is a character vector containing individuals/variables to be drawn
- cos2 if cos2 is in [0, 1], ex: 0.6, then individuals/variables with a cos2 > 0.6 are drawn. if cos2 > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn.
- contrib if contrib > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn

`title` the title of the graph

`jitter` a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).

- what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b"
- width: degree of jitter in x direction
- height: degree of jitter in y direction

`...` Arguments to be passed to the function fviz_mfa_quali_biplot()

`col.quanti.sup, col.quali.sup`
a color for the quantitative/qualitative supplementary variables.

col.circle	a color for the correlation circle.
arrows	Vector of two logicals specifying if the plot should contain points (FALSE, default) or arrows (TRUE). First value sets the rows and the second value sets the columns.
legend.partial.title	the title of the partal groups legend.
linesize	size of partial point connecting line.

Value

a ggplot2 plot

Author(s)

Fabian Mundt <f.mundt@inventionate.de>

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Multiple Factor Analysis
# ++++++++++++++++++++++++
# Install and load FactoMineR to compute MFA
# install.packages("FactoMineR")
library("FactoMineR")
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
               name.group=c("desc","desc2","symptom","eat"),
               num.group.sup=1:2, graph=FALSE)

# Graph of individuals
# ++++++++++++++++++++

# Default plot
# Use repel = TRUE to avoid overplotting (slow if many points)
# Color of individuals: col.ind = "#2E9FDF"
fviz_mfa_ind(res.mfa, repel = TRUE, col.ind = "#2E9FDF")+
theme_minimal()

## Not run:
# 1. Control automatically the color of individuals
   # using the "cos2" or the contributions "contrib"
   # cos2 = the quality of the individuals on the factor map
# 2. To keep only point or text use geom = "point" or geom = "text".
# 3. Change themes: http://www.sthda.com/english/wiki/ggplot2-themes

fviz_mfa_ind(res.mfa, col.ind = "cos2")+
 theme_minimal()
```

```
# Change gradient color
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_mfa_ind(res.mfa, col.ind="cos2", repel = TRUE) +
      scale_color_gradient2(low = "white", mid = "#2E9FDF",
      high= "#FC4E07", midpoint=0.4, space = "Lab")+
 theme_minimal()

## End(Not run)

# Color individuals by groups, add concentration ellipses
# Remove labels: label = "none".
grp <- as.factor(poison[, "Vomiting"])
p <- fviz_mfa_ind(res.mfa, label="none", habillage=grp,
       addEllipses=TRUE, ellipse.level=0.95)
print(p)

 ## Not run:
# Change group colors using RColorBrewer color palettes
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_brewer(palette="Paired") +
    scale_fill_brewer(palette="Paired") +
     theme_minimal()

## End(Not run)

# Change group colors manually
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_manual(values=c("#999999", "#E69F00"))+
 scale_fill_manual(values=c("#999999", "#E69F00"))+
 theme_minimal()

 ## Not run:
# Select and visualize some individuals (ind) with select.ind argument.
 # - ind with cos2 >= 0.4: select.ind = list(cos2 = 0.4)
 # - Top 20 ind according to the cos2: select.ind = list(cos2 = 20)
 # - Top 20 contributing individuals: select.ind = list(contrib = 20)
 # - Select ind by names: select.ind = list(name = c("44", "38", "53",  "39") )

# Example: Select the top 20 according to the cos2
fviz_mfa_ind(res.mfa, select.ind = list(cos2 = 20))

## End(Not run)

# Graph of qantitative variable categories
# ++++++++++++++++++++++++++++++++++++++++
data(wine)
res.mfa <- MFA(wine, group=c(2,5,3,10,9,2), type=c("n",rep("s",5)),
               ncp=5, name.group=c("orig","olf","vis","olfag","gust","ens"),
               num.group.sup=c(1,6), graph=FALSE)

# Default plot
fviz_mfa_quanti_var(res.mfa, col.var = "#FC4E07")+
```

```
theme_minimal()

## Not run:
# Control variable colors using their contributions
fviz_mfa_quanti_var(res.mfa, col.var = "contrib")+
 scale_color_gradient2(low = "white", mid = "blue",
           high = "red", midpoint = 20) +
 theme_minimal()

# Select variables with select.var argument
   # You can select by contrib, cos2 and name
   # as previously described for ind
# Select the top 10 contributing variables
fviz_mfa_quanti_var(res.mfa, select.var = list(contrib = 10))

## End(Not run)

# Graph of categorical variable categories
# ++++++++++++++++++++++++++++++++++++++++
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
               name.group=c("desc","desc2","symptom","eat"),
               num.group.sup=1:2, graph=FALSE)

# Plot
# Control variable colors using their contributions
fviz_mfa_quali_var(res.mfa, col.var = "contrib")+
 scale_color_gradient2(low = "white", mid = "blue",
           high = "red", midpoint = 2) +
 theme_minimal()


 ## Not run:
# Select the top 10 contributing variable categories
fviz_mfa_quali_var(res.mfa, select.var = list(contrib = 10))

## End(Not run)


# Biplot of categorical variable categories and individuals
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 # Use repel = TRUE to avoid overplotting
grp <- as.factor(poison[, "Vomiting"])
fviz_mfa_quali_biplot(res.mfa, repel = FALSE, col.var = "#E7B800",
   habillage = grp, addEllipses = TRUE, ellipse.level = 0.95)+
 theme_minimal()


# Graph of partial individuals (starplot)
# ++++++++++++++++++++++++++++++++++++++++
fviz_mfa_ind_starplot(res.mfa, col.partial = "group.name")+
scale_color_brewer(palette = "Dark2")+
theme_minimal()
```

```
## Not run:
# Select the partial points of the top 5
# contributing individuals
fviz_mfa_ind_starplot(res.mfa,
                      select.partial = list(contrib = 2)) +
                      theme_minimal()

# Change colours of star segments
fviz_mfa_ind_starplot(res.mfa, select.partial = list(contrib = 5),
                      col.partial = "group.name") +
                      scale_color_brewer(palette = "Dark2") +
                      theme_minimal()

## End(Not run)


# Graph of groups (correlation square)
# +++++++++++++++++++++++++++++++++++
fviz_mfa_group(res.mfa)

#' # Graph of partial axes
# ++++++++++++++++++++++++
fviz_mfa_axes(res.mfa)
```

`fviz_nbclust` *Dertermining and Visualizing the Optimal Number of Clusters*

Description

Partitioning methods, such as k-means clustering require the users to specify the number of clusters to be generated.

- fviz_nbclust(): Dertemines and visualize the optimal number of clusters using different methods: **within cluster sums of squares**, **average silhouette** and **gap statistics**.
- fviz_gap_stat(): Visualize the gap statistic generated by the function `clusGap()` [in cluster package]. The optimal number of clusters is specified using the "firstmax" method (?cluster::clustGap).

Read more: Determining the optimal number of clusters

Usage

```
fviz_nbclust(x, FUNcluster = NULL, method = c("silhouette", "wss",
  "gap_stat"), diss = NULL, k.max = 10, nboot = 100,
  verbose = interactive(), barfill = "steelblue", barcolor = "steelblue",
  linecolor = "steelblue", print.summary = TRUE, ...)

fviz_gap_stat(gap_stat, linecolor = "steelblue", maxSE = list(method =
  "firstmax", SE.factor = 1))
```

Arguments

x	numeric matrix or data frame. In the function fviz_nbclust(), x can be the results of the function NbClust().
FUNcluster	a partitioning function which accepts as first argument a (data) matrix like x, second argument, say k, k >= 2, the number of clusters desired, and returns a list with a component named cluster which contains the grouping of observations. Allowed values include: kmeans, cluster::pam, cluster::clara, cluster::fanny, hcut, etc. This argument is not required when x is an output of the function NbClust().
method	the method to be used for estimating the optimal number of clusters. Possible values are "silhouette" (for average silhouette width), "wss" (for total within sum of square) and "gap_stat" (for gap statistics).
diss	dist object as produced by dist(), i.e.: diss = dist(x, method = "euclidean"). Used to compute the average silhouette width of clusters, the within sum of square and hierarchical clustering. If NULL, dist(x) is computed with the default method = "euclidean"
k.max	the maximum number of clusters to consider, must be at least two.
nboot	integer, number of Monte Carlo ("bootstrap") samples. Used only for determining the number of clusters using gap statistic.
verbose	logical value. If TRUE, the result of progress is printed.
barfill, barcolor	fill color and outline color for bars
linecolor	color for lines
print.summary	logical value. If true, the optimal number of clusters are printed in fviz_nbclust().
...	optionally further arguments for FUNcluster()
gap_stat	an object of class "clusGap" returned by the function clusGap() [in cluster package]
maxSE	a list containing the parameters (method and SE.factor) for determining the location of the maximum of the gap statistic (Read the documentation ?cluster::maxSE). Allowed values for maxSE$method include: • "globalmax": simply corresponds to the global maximum, i.e., is which.max(gap) • "firstmax": gives the location of the first local maximum • "Tibs2001SEmax": uses the criterion, Tibshirani et al (2001) proposed: "the smallest k such that gap(k) >= gap(k+1) - s_k+1". It's also possible to use "the smallest k such that gap(k) >= gap(k+1) - SE.factor*s_k+1" where SE.factor is a numeric value which can be 1 (default), 2, 3, etc. • "firstSEmax": location of the first f() value which is not larger than the first local maximum minus SE.factor * SE.f[], i.e, within an "f S.E." range of that maximum. • see ?cluster::maxSE for more options

Value

- fviz_nbclust, fviz_gap_stat: return a ggplot2

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_cluster, eclust

Examples

```
set.seed(123)

# Data preparation
# +++++++++++++++
data("iris")
head(iris)
# Remove species column (5) and scale the data
iris.scaled <- scale(iris[, -5])


# Optimal number of clusters in the data
# ++++++++++++++++++++++++++++++++++++
# Examples are provided only for kmeans, but
# you can also use cluster::pam (for pam) or
#  hcut (for hierarchical clustering)

### Elbow method (look at the knee)
# Elbow method for kmeans
fviz_nbclust(iris.scaled, kmeans, method = "wss") +
geom_vline(xintercept = 3, linetype = 2)

# Average silhouette for kmeans
fviz_nbclust(iris.scaled, kmeans, method = "silhouette")

### Gap statistic
library(cluster)
set.seed(123)
# Compute gap statistic for kmeans
# we used B = 10 for demo. Recommended value is ~500
gap_stat <- clusGap(iris.scaled, FUN = kmeans, nstart = 25,
 K.max = 10, B = 10)
 print(gap_stat, method = "firstmax")
fviz_gap_stat(gap_stat)

# Gap statistic for hierarchical clustering
gap_stat <- clusGap(iris.scaled, FUN = hcut, K.max = 10, B = 10)
fviz_gap_stat(gap_stat)
```

fviz_pca *Visualize Principal Component Analysis*

Description

Principal component analysis (PCA) reduces the dimensionality of multivariate data, to two or three that can be visualized graphically with minimal loss of information. fviz_pca() provides ggplot2-based elegant visualization of PCA outputs from: i) prcomp and princomp [in built-in R stats], ii) PCA [in FactoMineR] and iii) dudi.pca [in ade4]. Read more: Principal Component Analysis

- fviz_pca_ind(): Graph of individuals
- fviz_pca_var(): Graph of variables
- fviz_pca_biplot(): Biplot of individuals and variables
- fviz_pca(): An alias of fviz_pca_biplot()

Usage

```
fviz_pca(X, ...)

fviz_pca_ind(X, axes = c(1, 2), geom = c("point", "text"), repel = FALSE,
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  ellipse.type = "norm", ellipse.alpha = 0.1, col.ind = "black",
  col.ind.sup = "blue", alpha.ind = 1, select.ind = list(name = NULL, cos2
  = NULL, contrib = NULL), jitter = list(what = "label", width = NULL, height
  = NULL), title = "Individuals factor map - PCA", axes.linetype = "dashed",
  ...)

fviz_pca_var(X, axes = c(1, 2), geom = c("arrow", "text"), label = "all",
  invisible = "none", repel = FALSE, labelsize = 4, col.var = "black",
  alpha.var = 1, col.quanti.sup = "blue", col.circle = "grey70",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  jitter = list(what = "label", width = NULL, height = NULL),
  title = "Variables factor map - PCA", axes.linetype = "dashed")

fviz_pca_biplot(X, axes = c(1, 2), geom = c("point", "text"),
  label = "all", invisible = "none", labelsize = 4, pointsize = 2,
  habillage = "none", addEllipses = FALSE, ellipse.level = 0.95,
  col.ind = "black", col.ind.sup = "blue", alpha.ind = 1,
  col.var = "steelblue", alpha.var = 1, col.quanti.sup = "blue",
  col.circle = "grey70", repel = FALSE, axes.linetype = "dashed",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
  title = "Biplot of variables and individuals", jitter = list(what =
  "label", width = NULL, height = NULL), ...)
```

Arguments

X	an object of class PCA [FactoMineR]; prcomp and princomp [stats]; dudi and pca [ade4].
...	Arguments to be passed to the function fviz_pca_biplot().
axes	a numeric vector of length 2 specifying the dimensions to be plotted.
geom	a text specifying the geometry to be used for the graph. Allowed values are the combination of c("point", "arrow", "text"). Use "point" (to show only points); "text" to show only labels; c("point", "text") or c("arrow", "text") to show both types.
repel	a boolean, whether to use ggrepel to avoid overplotting text labels or not.
label	a text specifying the elements to be labelled. Default value is "all". Allowed values are "none" or the combination of c("ind", "ind.sup", "quali", "var", "quanti.sup"). "ind" can be used to label only active individuals. "ind.sup" is for supplementary individuals. "quali" is for supplementary qualitative variables. "var" is for active variables. "quanti.sup" is for quantitative supplementary variables.
invisible	a text specifying the elements to be hidden on the plot. Default value is "none". Allowed values are the combination of c("ind", "ind.sup", "quali", "var", "quanti.sup").
labelsize	font size for the labels
pointsize	the size of points
habillage	an optional factor variable for coloring the observations by groups. Default value is "none". If X is a PCA object from FactoMineR package, habillage can also specify the supplementary qualitative variable (by its index or name) to be used for coloring individuals by groups (see ?PCA in FactoMineR).
addEllipses	logical value. If TRUE, draws ellipses around the individuals when habillage != "none".
ellipse.level	the size of the concentration ellipse in normal probability.
ellipse.type	Character specifying frame type. Possible values are 'convex' or types supporeted by stat_ellipse including one of c("t", "norm", "euclid").
ellipse.alpha	Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.
col.ind, col.var	color for individuals and variables, respectively. Possible values include also : "cos2", "contrib", "coord", "x" or "y". In this case, the colors for individuals/variables are automatically controlled by their qualities of representation ("cos2"), contributions ("contrib"), coordinates (x^2+y^2, "coord"), x values ("x") or y values ("y"). To use automatic coloring (by cos2, contrib, ...), make sure that habillage ="none".
col.ind.sup	color for supplementary individuals
alpha.ind, alpha.var	controls the transparency of individual and variable colors, respectively. The value can variate from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values include also : "cos2", "contrib", "coord", "x" or

"y". In this case, the transparency for the individual/variable colors are automatically controlled by their qualities ("cos2"), contributions ("contrib"), coordinates (x^2+y^2, "coord"), x values("x") or y values("y"). To use this, make sure that habillage ="none".

`select.ind, select.var`
: a selection of individuals/variables to be drawn. Allowed values are NULL or a list containing the arguments name, cos2 or contrib:
  - name: is a character vector containing individuals/variables to be drawn
  - cos2: if cos2 is in [0, 1], ex: 0.6, then individuals/variables with a cos2 > 0.6 are drawn. if cos2 > 1, ex: 5, then the top 5 individuals/variables with the highest cos2 are drawn.
  - contrib: if contrib > 1, ex: 5, then the top 5 individuals/variables with the highest contrib are drawn

`jitter`
: a parameter used to jitter the points in order to reduce overplotting. It's a list containing the objects what, width and height (i.e jitter = list(what, width, height)).
  - what: the element to be jittered. Possible values are "point" or "p"; "label" or "l"; "both" or "b".
  - width: degree of jitter in x direction
  - height: degree of jitter in y direction

`title`
: the title of the graph

`axes.linetype`
: linetype of x and y axes.

`col.quanti.sup`
: a color for the quantitative supplementary variables.

`col.circle`
: a color for the correlation circle.

Value

a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_ca, fviz_mca

Examples

```
# Principal component analysis
# ++++++++++++++++++++++++++++++
data(iris)
res.pca <- prcomp(iris[, -5],  scale = TRUE)

# Graph of individuals
# +++++++++++++++++++++
```

```
# Default plot
fviz_pca_ind(res.pca, col.ind = "#00AFBB")


# 1. Control automatically the color of individuals
   # using the "cos2" or the contributions "contrib"
   # cos2 = the quality of the individuals on the factor map
# 2. To keep only point or text use geom = "point" or geom = "text".
# 3. Change themes: http://www.sthda.com/english/wiki/ggplot2-themes

fviz_pca_ind(res.pca, col.ind="cos2", geom = "point")+
 theme_minimal()

# Change gradient color
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_pca_ind(res.pca, col.ind="cos2", repel = TRUE) +
      scale_color_gradient2(low = "white", mid = "#2E9FDF",
      high= "#FC4E07", midpoint=0.6, space = "Lab")+
      theme_minimal()

# You can also control the transparency
# of the color by the cos2
fviz_pca_ind(res.pca, alpha.ind="cos2") +
     theme_minimal()

# Color individuals by groups, add concentration ellipses
# Remove labels: label = "none".
p <- fviz_pca_ind(res.pca, label="none", habillage=iris$Species,
       addEllipses=TRUE, ellipse.level=0.95)
print(p)

# Change group colors using RColorBrewer color palettes
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_brewer(palette="Dark2") +
    scale_fill_brewer(palette="Dark2") +
     theme_minimal()

# Change group colors manually
# Read more: http://www.sthda.com/english/wiki/ggplot2-colors
p + scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
 scale_fill_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
 theme_minimal()

# Select and visualize some individuals (ind) with select.ind argument.
 # - ind with cos2 >= 0.96: select.ind = list(cos2 = 0.96)
 # - Top 20 ind according to the cos2: select.ind = list(cos2 = 20)
 # - Top 20 contributing individuals: select.ind = list(contrib = 20)
 # - Select ind by names: select.ind = list(name = c("23", "42", "119") )

 # Example: Select the top 40 according to the cos2
fviz_pca_ind(res.pca, select.ind = list(cos2 = 40))
```

```
# Graph of variables
# +++++++++++++++++++++++++++

# Default plot
fviz_pca_var(res.pca, col.var = "steelblue")+
theme_minimal()

# Control variable colors using their contributions
fviz_pca_var(res.pca, col.var = "contrib")+
 scale_color_gradient2(low="white", mid="blue",
           high="red", midpoint=96, space = "Lab") +
 theme_minimal()

# Select variables with select.var argument
   # You can select by contrib, cos2 and name
   # as previously described for ind
# Select the top 3 contributing variables
fviz_pca_var(res.pca, select.var = list(contrib = 3))


# Biplot of individuals and variables
# ++++++++++++++++++++++++++
fviz_pca_biplot(res.pca)

# Keep only the labels for variables
# Change the color by groups, add ellipses
fviz_pca_biplot(res.pca, label = "var", habillage=iris$Species,
               addEllipses=TRUE, ellipse.level=0.95)+
theme_minimal()
```

`fviz_silhouette` *Visualize Silhouette Information from Clustering*

Description

Silhouette (Si) analysis is a cluster validation approach that measures how well an observation is clustered and it estimates the average distance between clusters. fviz_silhouette() provides ggplot2-based elegant visualization of silhouette information from i) the result of `silhouette`(), `pam`(), `clara`() and `fanny`() [in cluster package]; ii) `eclust`() and `hcut`() [in factoextra].

Read more: Clustering Validation Statistics.

Usage

```
fviz_silhouette(sil.obj, label = FALSE, print.summary = TRUE)
```

Arguments

sil.obj	an object of class silhouette: pam, clara, fanny [in cluster package]; eclust and hcut [in factoextra].
label	logical value. If true, x axis tick labels are shown
print.summary	logical value. If true a summary of cluster silhouettes are printed in fviz_silhouette().

Details

- Observations with a large silhouhette Si (almost 1) are very well clustered.
- A small Si (around 0) means that the observation lies between two clusters.
- Observations with a negative Si are probably placed in the wrong cluster.

Value

return a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_cluster, hcut, hkmeans, eclust, fviz_dend

Examples

```
set.seed(123)

# Data preparation
# +++++++++++++++
data("iris")
head(iris)
# Remove species column (5) and scale the data
iris.scaled <- scale(iris[, -5])

# K-means clustering
# +++++++++++++++++++++
km.res <- kmeans(iris.scaled, 3, nstart = 25)

# Visualize kmeans clustering
fviz_cluster(km.res, iris[, -5], frame.type = "norm")+
theme_minimal()

# Visualize silhouhette information
require("cluster")
sil <- silhouette(km.res$cluster, dist(iris.scaled))
fviz_silhouette(sil)

# Identify observation with negative silhouette
neg_sil_index <- which(sil[, "sil_width"] < 0)
```

```
sil[neg_sil_index, , drop = FALSE]

# PAM clustering
# ++++++++++++++++++++
require(cluster)
pam.res <- pam(iris.scaled, 3)
# Visualize pam clustering
fviz_cluster(pam.res, frame.type = "norm")+
theme_minimal()
# Visualize silhouhette information
fviz_silhouette(pam.res)

# Hierarchical clustering
# ++++++++++++++++++++++++
# Use hcut() which compute hclust and cut the tree
hc.cut <- hcut(iris.scaled, k = 3, hc_method = "complete")
# Visualize dendrogram
fviz_dend(hc.cut, show_labels = FALSE, rect = TRUE)
# Visualize silhouhette information
fviz_silhouette(hc.cut)
```

`get_ca` *Extract the results for rows/columns - CA*

Description

Extract all the results (coordinates, squared cosine, contributions and inertia) for the active row/column variables from Correspondence Analysis (CA) outputs.

- get_ca(): Extract the results for rows and columns
- get_ca_row(): Extract the results for rows only
- get_ca_col(): Extract the results for columns only

Usage

```
get_ca(res.ca, element = c("row", "col"))

get_ca_col(res.ca)

get_ca_row(res.ca)
```

Arguments

`res.ca`	an object of class CA [FactoMineR], ca [ca], coa [ade4]; correspondence [MASS].
`element`	the element to subset from the output. Possible values are "row" or "col".

Value

a list of matrices containing the results for the active rows/columns including :

coord	coordinates for the rows/columns
cos2	cos2 for the rows/columns
contrib	contributions of the rows/columns
inertia	inertia of the rows/columns

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Install and load FactoMineR to compute CA
# install.packages("FactoMineR")
 library("FactoMineR")
 data("housetasks")
 res.ca <- CA(housetasks, graph = FALSE)

# Result for column variables
 col <- get_ca_col(res.ca)
 col # print
 head(col$coord) # column coordinates
 head(col$cos2) # column cos2
 head(col$contrib) # column contributions

# Result for row variables
 row <- get_ca_row(res.ca)
 row # print
 head(row$coord) # row coordinates
 head(row$cos2) # row cos2
 head(row$contrib) # row contributions

 # You can also use the function get_ca()
 get_ca(res.ca, "row") # Results for rows
 get_ca(res.ca, "col") # Results for columns
```

get_clust_tendency *Assessing Clustering Tendency*

Description

Before applying cluster methods, the first step is to assess whether the data is clusterable, a process defined as the **assessing of clustering tendency**. get_clust_tendency() assesses clustering tendency using Hopkins' statistic and a visual approach. An ordered dissimilarity image (ODI) is shown. Objects belonging to the same cluster are displayed in consecutive order using hierarchical clustering. For more details and interpretation, see STHDA website: Assessing clustering tendency.

Usage

```
get_clust_tendency(data, n, graph = TRUE, gradient = list(low = "red", mid =
  "white", high = "blue"), seed = 123)
```

Arguments

data	a numeric data frame or matrix. Columns are variables and rows are samples. Computation are done on rows (samples) by default. If you want to calculate Hopkins statistic on variables, transpose the data before.
n	the number of points selected from sample space which is also the number of points selected from the given sample(data).
graph	logical value; if TRUE the ordered dissimilarity image (ODI) is shown.
gradient	a list containing three elements specifying the colors for low, mid and high values in the ordered dissimilarity image. The element "mid" can take the value of NULL.
seed	an integer specifying the seed for random number generator. Specify seed for reproducible results.

Details

Hopkins statistic: If the value of Hopkins statistic is close to zero (far below 0.5), then we can conclude that the dataset is significantly clusterable.

VAT (Visual Assessment of cluster Tendency): The VAT detects the clustering tendency in a visual form by counting the number of square shaped dark (or colored) blocks along the diagonal in a VAT image.

Value

A list containing the elements:

- hopkins_stat for Hopkins statistic value

- plot for ordered dissimilarity image. This is generated using the function fviz_dist(dist.obj).

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

fviz_dist

Examples

```
data(iris)

# Clustering tendency
gradient_col = list(low = "steelblue", high = "white")
get_clust_tendency(iris[,-5], n = 50, gradient = gradient_col)

# Random uniformly distributed dataset
# (without any inherent clusters)
set.seed(123)
random_df <- apply(iris[, -5], 2,
                   function(x){runif(length(x), min(x), max(x))}
                   )
get_clust_tendency(random_df, n = 50, gradient = gradient_col)
```

---

get_hmfa *Extract the results for individuals/quantitative variables/qualitative variables/group/partial axes - HMFA*

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals/quantitative variable categories/qualitative variable categories/groups/partial axes from Hierarchical Multiple Factor Analysis (HMFA) outputs.

- get_hmfa(): Extract the results for variables and individuals
- get_hmfa_ind(): Extract the results for individuals only
- get_hmfa_var_qanti(): Extract the results for quantitative variables only
- get_hmfa_var_qali(): Extract the results for qualitative variables only
- get_hmfa_group(): Extract the results for groups only

Usage

```
get_hmfa(res.hmfa, element = c("ind", "quanti.var", "quali.var", "group"))

get_hmfa_ind(res.hmfa)

get_hmfa_quanti_var(res.hmfa)

get_hmfa_quali_var(res.hmfa)

get_hmfa_group(res.hmfa)

get_hmfa_partial(res.hmfa)
```

Arguments

res.hmfa	an object of class HMFA [FactoMineR].
element	the element to subset from the output. Possible values are "ind", "quanti.var", "quali.var" or "group".

Value

a list of matrices containing the results for the active individuals/quantitative variable categories/qualitative variable categories/groups/partial axes including :

coord	coordinates for the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
cos2	cos2 for the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
contrib	contributions of the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
inertia	inertia of the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Fabian Mundt <f.mundt@inventionate.de>

References

http://www.sthda.com

Examples

```
# Multiple Factor Analysis
# ++++++++++++++++++++++++
# Install and load FactoMineR to compute MFA
```

```
# install.packages("FactoMineR")
library("FactoMineR")
data(wine)
hierar <- list(c(2,5,3,10,9,2), c(4,2))
res.hmfa <- HMFA(wine, H = hierar, type=c("n",rep("s",5)), graph = FALSE)

 # Extract the results for qualitative variable categories
 var <- get_hmfa_quali_var(res.hmfa)
 print(var)
 head(var$coord) # coordinates of qualitative variables
 head(var$cos2) # cos2 of qualitative variables
 head(var$contrib) # contributions of qualitative variables

 # Extract the results for individuals
 ind <- get_hmfa_ind(res.hmfa)
 print(ind)
 head(ind$coord) # coordinates of individuals
 head(ind$cos2) # cos2 of individuals
 head(ind$contrib) # contributions of individuals

 # You can also use the function get_hmfa()
 get_hmfa(res.hmfa, "ind") # Results for individuals
 get_hmfa(res.hmfa, "quali.var") # Results for qualitative variable categories
```

`get_mca` *Extract the results for individuals/variables - MCA*

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals/variable categories from Multiple Correspondence Analysis (MCA) outputs.

- get_mca(): Extract the results for variables and individuals
- get_mca_ind(): Extract the results for individuals only
- get_mca_var(): Extract the results for variables only

Usage

```
get_mca(res.mca, element = c("var", "ind"))

get_mca_var(res.mca)

get_mca_ind(res.mca)
```

Arguments

res.mca	an object of class MCA [FactoMineR], acm [ade4].
element	the element to subset from the output. Possible values are "var" or "ind".

Value

a list of matrices containing the results for the active individuals/variable categories including :

coord	coordinates for the individuals/variable categories
cos2	cos2 for the individuals/variable categories
contrib	contributions of the individuals/variable categories
inertia	inertia of the individuals/variable categories

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Multiple Correspondence Analysis
# ++++++++++++++++++++++++++++++
# Install and load FactoMineR to compute MCA
# install.packages("FactoMineR")
library("FactoMineR")
data(poison)
poison.active <- poison[1:55, 5:15]
head(poison.active[, 1:6])
res.mca <- MCA(poison.active, graph=FALSE)

 # Extract the results for variable categories
 var <- get_mca_var(res.mca)
 print(var)
 head(var$coord) # coordinates of variables
 head(var$cos2) # cos2 of variables
 head(var$contrib) # contributions of variables

 # Extract the results for individuals
 ind <- get_mca_ind(res.mca)
 print(ind)
 head(ind$coord) # coordinates of individuals
 head(ind$cos2) # cos2 of individuals
 head(ind$contrib) # contributions of individuals

 # You can also use the function get_mca()
 get_mca(res.mca, "ind") # Results for individuals
```

```
get_mca(res.mca, "var") # Results for variable categories
```

`get_mfa` *Extract the results for individuals/quantitative variables/qualitative variables/group/partial axes - MFA*

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals/quantitative variable categories/qualitative variable categories/groups/partial axes from Multiple Factor Analysis (MFA) outputs.

- get_mfa(): Extract the results for variables and individuals
- get_mfa_ind(): Extract the results for individuals only
- get_mfa_var_qanti(): Extract the results for quantitative variables only
- get_mfa_var_qali(): Extract the results for qualitative variables only
- get_mfa_group(): Extract the results for groups only
- get_mfa_partial_axes(): Extract the results for partial axes only

Usage

```
get_mfa(res.mfa, element = c("ind", "quanti.var", "quali.var", "group",
  "partial.axes"))

get_mfa_ind(res.mfa)

get_mfa_quanti_var(res.mfa)

get_mfa_quali_var(res.mfa)

get_mfa_group(res.mfa)

get_mfa_partial_axes(res.mfa)
```

Arguments

`res.mfa`	an object of class MFA [FactoMineR].
`element`	the element to subset from the output. Possible values are "ind", "quanti.var", "quali.var", "group" or "partial.axes".

Value

a list of matrices containing the results for the active individuals/quantitative variable categories/qualitative variable categories/groups/partial axes including :

coord	coordinates for the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
cos2	cos2 for the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
contrib	contributions of the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes
inertia	inertia of the individuals/quantitative variable categories/qualitative variable categories/groups/partial axes

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Fabian Mundt <f.mundt@inventionate.de>

References

http://www.sthda.com

Examples

```
# Multiple Factor Analysis
# ++++++++++++++++++++++++
# Install and load FactoMineR to compute MFA
# install.packages("FactoMineR")
library("FactoMineR")
data(poison)
res.mfa <- MFA(poison, group=c(2,2,5,6), type=c("s","n","n","n"),
name.group=c("desc","desc2","symptom","eat"), num.group.sup=1:2,
graph = FALSE)

 # Extract the results for qualitative variable categories
 var <- get_mfa_quali_var(res.mfa)
 print(var)
 head(var$coord) # coordinates of qualitative variables
 head(var$cos2) # cos2 of qualitative variables
 head(var$contrib) # contributions of qualitative variables

 # Extract the results for individuals
 ind <- get_mfa_ind(res.mfa)
 print(ind)
 head(ind$coord) # coordinates of individuals
 head(ind$cos2) # cos2 of individuals
 head(ind$contrib) # contributions of individuals

 # You can also use the function get_mfa()
```

```
get_mfa(res.mfa, "ind") # Results for individuals
get_mfa(res.mfa, "quali.var") # Results for qualitative variable categories
```

`get_pca` *Extract the results for individuals/variables - PCA*

Description

Extract all the results (coordinates, squared cosine, contributions) for the active individuals/variables from Principal Component Analysis (PCA) outputs.

- get_pca(): Extract the results for variables and individuals
- get_pca_ind(): Extract the results for individuals only
- get_pca_var(): Extract the results for variables only

Usage

```
get_pca(res.pca, element = c("var", "ind"))

get_pca_ind(res.pca, ...)

get_pca_var(res.pca)
```

Arguments

`res.pca`	an object of class PCA [FactoMineR]; prcomp and princomp [stats]; pca, dudi [adea4].
`element`	the element to subset from the output. Allowed values are "var" (for active variables) or "ind" (for active individuals).
`...`	not used

Value

a list of matrices containing all the results for the active individuals/variables including:

`coord`	coordinates for the individuals/variables
`cos2`	cos2 for the individuals/variables
`contrib`	contributions of the individuals/variables

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
# Principal Component Analysis
# +++++++++++++++++++++++++++++
 data(iris)
 res.pca <- prcomp(iris[, -5],  scale = TRUE)
 # Extract the results for individuals
 ind <- get_pca_ind(res.pca)
 print(ind)
 head(ind$coord) # coordinates of individuals
 head(ind$cos2) # cos2 of individuals
 head(ind$contrib) # contributions of individuals

 # Extract the results for variables
 var <- get_pca_var(res.pca)
 print(var)
 head(var$coord) # coordinates of variables
 head(var$cos2) # cos2 of variables
 head(var$contrib) # contributions of variables

 # You can also use the function get_pca()
 get_pca(res.pca, "ind") # Results for individuals
 get_pca(res.pca, "var") # Results for variable categories
```

---

hcut *Computes Hierarchical Clustering and Cut the Tree*

Description

Computes hierarchical clustering (hclust, agnes, diana) and cut the tree into k clusters. It also accepts correlation based distance measure methods such as "pearson", "spearman" and "kendall".

Usage

```
hcut(x, k = 2, isdiss = inherits(x, "dist"), hc_func = c("hclust",
  "agnes", "diana"), hc_method = "ward.D2", hc_metric = "euclidean",
  stand = FALSE, graph = FALSE, ...)
```

Arguments

x	a numeric matrix, numeric data frame or a dissimilarity matrix.
k	the number of clusters to be generated.
isdiss	logical value specifying wether x is a dissimilarity matrix.

hc_func	the hierarchical clustering function to be used. Default value is "hclust". Possible values is one of "hclust", "agnes", "diana". Abbreviation is allowed.
hc_method	the agglomeration method to be used (?hclust) for hclust() and agnes(): "ward.D", "ward.D2", "single", "complete", "average", ...
hc_metric	character string specifying the metric to be used for calculating dissimilarities between observations. Allowed values are those accepted by the function dist() [including "euclidean", "manhattan", "maximum", "canberra", "binary", "minkowski"] and correlation based distance measures ["pearson", "spearman" or "kendall"].
stand	logical value; default is FALSE. If TRUE, then the data will be standardized using the function scale(). Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's standard deviation.
graph	logical value. If TRUE, the dendrogram is displayed.
...	not used.

Value

an object of class "hcut" containing the result of the standard function used (read the documentation of hclust, agnes, diana).

It includes also:

- cluster: the cluster assignement of observations after cutting the tree
- nbclust: the number of clusters
- silinfo: the silhouette information of observations (if k > 1)
- size: the size of clusters
- data: a matrix containing the original or the standardized data (if stand = TRUE)

See Also

fviz_dend, hkmeans, eclust

Examples

```
data(USArrests)

# Compute hierarchical clustering and cut into 4 clusters
res <- hcut(USArrests, k = 4, stand = TRUE)

# Cluster assignements of observations
res$cluster
# Size of clusters
res$size

# Visualize the dendrogram
fviz_dend(res, rect = TRUE)
```

```
# Visualize the silhouette
fviz_silhouette(res)

# Visualize clusters as scatter plots
fviz_cluster(res)
```

---

hkmeans *Hierarchical k-means clustering*

Description

The final k-means clustering solution is very sensitive to the initial random selection of cluster centers. This function provides a solution using an hybrid approach by combining the hierarchical clustering and the k-means methods. The procedure is explained in "Details" section.

- hkmeans(): compute hierarchical k-means clustering
- print.hkmeans(): prints the result of hkmeans
- hkmeans_tree(): plots the initial dendrogram

Usage

```
hkmeans(x, k, hc.metric = "euclidean", hc.method = "ward.D2",
  iter.max = 10, km.algorithm = "Hartigan-Wong")

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

hkmeans_tree(hkmeans, rect.col = NULL, ...)
```

Arguments

x	a numeric matrix, data frame or vector
k	the number of clusters to be generated
hc.metric	the distance measure to be used. Possible values are "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski" (see ?dist).
hc.method	the agglomeration method to be used. Possible values include "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median"or "centroid" (see ?hclust).
iter.max	the maximum number of iterations allowed for k-means.
km.algorithm	the algorithm to be used for kmeans (see ?kmeans).
...	others arguments to be passed to the function plot.hclust(); (see ? plot.hclust)
hkmeans	an object of class hkmeans (returned by the function hkmeans())
rect.col	Vector with border colors for the rectangles around clusters in dendrogram

Details

The procedure is as follow:

1. Compute hierarchical clustering
2. Cut the tree in k-clusters
3. compute the center (i.e the mean) of each cluster
4. Do k-means by using the set of cluster centers (defined in step 3) as the initial cluster centers

Value

hkmeans returns an object of class "hkmeans" containing the following components:

- The elements returned by the standard function kmeans() (see ?kmeans)
- data: the data used for the analysis
- hclust: an object of class "hclust" generated by the function hclust()

Examples

```
# Load data
data(USArrests)
# Scale the data
df <- scale(USArrests)

# Compute hierarchical k-means clustering
res.hk <-hkmeans(df, 4)

# Elements returned by hkmeans()
names(res.hk)

# Print the results
res.hk

# Visualize the tree
hkmeans_tree(res.hk, cex = 0.6)
# or use this
fviz_dend(res.hk, cex = 0.6)


# Visualize the hkmeans final clusters
fviz_cluster(res.hk, frame.type = "norm", frame.level = 0.68)
```

---

housetasks *House tasks contingency table*

Description

A data frame containing the frequency of execution of 13 house tasks in the couple. This table is also available in ade4 package.

Usage

```
data("housetasks")
```

Format

A data frame with 13 observations (house tasks) on the following 4 columns.

`Wife` a numeric vector

`Alternating` a numeric vector

`Husband` a numeric vector

`Jointly` a numeric vector

Source

This data is from FactoMineR package.

Examples

```
library(FactoMineR)
data(housetasks)
res.ca <- CA(housetasks, graph=FALSE)
fviz_ca_biplot(res.ca, repel = TRUE)+
theme_minimal()
```

---

multishapes *A dataset containing clusters of multiple shapes*

Description

Data containing clusters of any shapes. Useful for comparing density-based clustering (DBSCAN) and standard partitioning methods such as k-means clustering.

Usage

```
data("multishapes")
```

Format

A data frame with 1100 observations on the following 3 variables.

`x` a numeric vector containing the x coordinates of observations

`y` a numeric vector containing the y coordinates of observations

`shape` a numeric vector corresponding to the cluster number of each observations.

Details

The dataset contains 5 clusters and some outliers/noises.

Examples

```
data(multishapes)
plot(multishapes[,1], multishapes[, 2],
    col = multishapes[, 3], pch = 19, cex = 0.8)
```

---

poison *Poison*

Description

This data is a result from a survey carried out on children of primary school who suffered from food poisoning. They were asked about their symptoms and about what they ate.

Usage

```
data("poison")
```

Format

A data frame with 55 rows and 15 columns.

Source

This data is from FactoMineR package.

Examples

```
library(FactoMineR)
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2, quali.sup = c(3,4),
   graph = FALSE)
fviz_mca_biplot(res.mca, repel = TRUE)+
theme_minimal()
```

print.factoextra *Print method for an object of class factoextra*

Description

Print method for an object of class factoextra

Usage

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

Arguments

x	an object of class factoextra
...	further arguments to be passed to print method

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

http://www.sthda.com

Examples

```
data(iris)
res.pca <- princomp(iris[, -5],  cor = TRUE)
ind <- get_pca_ind(res.pca, data = iris[, -5])
print(ind)
```

Index