

Package ‘factoextra’

July 24, 2026

Type Package

Title Extract and Visualize the Results of Multivariate Data Analyses

Version 2.2.0

Date 2026-07-24

Description Provides 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), 'FAMD' (Factor Analysis of Mixed Data), 'MFA' (Multiple Factor Analysis), and 'HMFA' (Hierarchical Multiple Factor Analysis) from different R packages. It standardizes coordinates, squared cosines, contributions, and eigenvalues from several analysis backends and produces 'ggplot2'-based factor maps, scree plots, contribution plots, clustering diagnostics, dendrograms, and silhouette plots. It also visualizes two-dimensional 'UMAP' and t-SNE embeddings and adapts precomputed or 'tidymodels' principal-component results for the same plotting interface.

License GPL-2

LazyData true

LazyDataCompression xz

Encoding UTF-8

Depends R (>= 4.1.0), ggplot2 (>= 3.5.2)

Imports cluster (>= 2.1.8.2), dendextend (>= 1.19.1), FactoMineR (>= 2.13), ggpubr (>= 0.6.3), grid, rlang (>= 1.1.7), stats, ggrepel (>= 0.9.5)

Suggests ade4, ca, hardhat, igraph, MASS, knitr, mclust, parsnip, recipes, rmarkdown, Rtsne, testthat (>= 3.0.0), umap, uwot, workflows

VignetteBuilder knitr

URL <https://github.com/kassambara/factoextra>,
<https://rpkgs.datanovia.com/factoextra/>

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

Collate 'fviz_add.R' 'eigenvalue.R' 'utilities.R' 'as_factoextra.R'
 'cluster_utilities.R' 'decathlon2.R' 'print.factoextra.R'
 'get_hmfa.R' 'fviz_hmfa.R' 'get_mfa.R' 'fviz_mfa.R'
 'deprecated.R' 'dist.R' 'fviz_dend.R' 'hcut.R' 'get_pca.R'
 'fviz_cluster.R' 'eclust.R' 'facto_summarize.R' 'fviz.R'
 'fviz_ca.R' 'fviz_contrib.R' 'fviz_cos2.R' 'fviz_ellipses.R'
 'fviz_famd.R' 'get_mca.R' 'fviz_mca.R' 'fviz_mclust.R'
 'fviz_nbclust.R' 'fviz_pca.R' 'fviz_silhouette.R' 'fviz_umap.R'
 'get_ca.R' 'get_clust_tendency.R' 'get_famd.R' 'hkmeans.R'
 'housetasks.R' 'multishapes.R' 'poison.R' 'theme_palette.R'
 'zzz.R'

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Alboukadel Kassambara [aut, cre] (ORCID:
<https://orcid.org/0009-0002-9136-0791>),
 Fabian Mundt [aut],
 Laszlo Erdey [ctb] (ORCID: <https://orcid.org/0000-0002-6781-4303>),
 affiliation: Faculty of Economics and Business, University of
 Debrecen, Hungary, contribution: Modern compatibility fixes, tests,
 and maintenance updates)

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

Repository CRAN

Date/Publication 2026-07-24 09:00:02 UTC

Contents

as_factoextra_pca	3
decathlon2	6
deprecated	7
dist	9
eclust	10
eigenvalue	13
factoextra_palette	16
factominer_category_map	18
facto_summarize	18
fviz	21
fviz_add	26
fviz_ca	27
fviz_cluster	32
fviz_contrib	36
fviz_cos2	39
fviz_dend	42
fviz_ellipses	46
fviz_famd	48
fviz_hmfa	51

fviz_mca	55
fviz_mclust	61
fviz_mfa	64
fviz_nbclust	68
fviz_pca	72
fviz_silhouette	78
fviz_umap	80
get_ca	83
get_clust_tendency	85
get_famd	87
get_hmfa	89
get_mca	90
get_mfa	92
get_pca	94
hcut	96
hkmeans	98
housetasks	100
map_factominer_legacy_names	101
multishapes	102
poison	102
print.factoextra	103
theme_factoextra	104

Index**106**

as_factoextra_pca	<i>Build a factoextra-ready object from pre-computed coordinates</i>
-------------------	--

Description

as_factoextra_pca() wraps pre-computed individual (and, optionally, variable) coordinates into an object that the [fviz_pca](#) family (fviz_pca_ind(), fviz_pca_var(), fviz_pca_biplot()), [fviz_eig](#), [fviz_contrib](#) and [fviz_cos2](#) can plot directly. Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

It lets you apply factoextra's visualizations to the output of an *eigenvalue-based* dimension reduction - for example stats::cmdscale(), ape::pcoa(), vegan::rda()/cca(), a tidymodels recipe/workflow with step_pca() (see the methods below), or a custom analysis - without having to write a dedicated backend. You bring the coordinates; factoextra draws the biplot, scree plot, contributions and cos2.

The scree plot, contributions and cos2 assume real eigenvalues, so this constructor is for PCA-family results. Non-linear embeddings (UMAP, t-SNE) have no eigenvalues; plot their coordinates directly (a scree/loadings display would be meaningless for them).

Usage

```
as_factoextra_pca(ind.coord, ...)

## Default S3 method:
as_factoextra_pca(
  ind.coord,
  var.coord = NULL,
  eig = NULL,
  ind.cos2 = NULL,
  ind.contrib = NULL,
  var.cos2 = NULL,
  var.contrib = NULL,
  var.cor = NULL,
  scale.unit = FALSE,
  ...
)

## S3 method for class 'recipe'
as_factoextra_pca(ind.coord, ...)

## S3 method for class 'workflow'
as_factoextra_pca(ind.coord, ...)
```

Arguments

ind.coord	the object to convert. For the default method, individual (observation) coordinates: a numeric matrix or data frame with one column per dimension (the "scores"). For the recipe / workflow methods, a prepped recipe or a fitted workflow whose PCA is done with <code>recipes::step_pca()</code> (the scores, loadings and eigenvalues are then extracted for you). Required.
...	passed to methods (unused by the default method).
var.coord	optional variable coordinates / loadings: a numeric matrix or data frame with one column per dimension. Supplying it enables <code>fviz_pca_var()</code> and <code>fviz_pca_biplot()</code> .
eig	optional numeric vector of eigenvalues ($\text{length} \geq \text{number of dimensions}$). Used by the scree plot and to label the axes with the percentage of explained variance. When NULL (default) it is set to the variance of each coordinate column (the natural definition of a PCA eigenvalue).
ind.cos2, ind.contrib, var.cos2, var.contrib, var.cor	optional pre-computed quality (cos2), contribution and (variable) correlation matrices, with the same dimensions as the corresponding coordinates. When omitted they are derived from the coordinates (see Details).
scale.unit	logical. If TRUE, the variable coordinates are treated as correlations and the correlation circle is drawn by <code>fviz_pca_var()</code> . Default is FALSE.

Details

When `cos2/contrib` are not supplied they are computed from the coordinates:

- $\text{contrib} = 100 * \text{coord}^2 / \text{colSums}(\text{coord}^2)$ - the exact contribution of each element to each dimension.
- $\text{cos2} = \text{coord}^2 / \text{rowSums}(\text{coord}^2)$ - the quality of representation *within the supplied dimensions*. This equals the true cos2 only when all components are provided; with a truncated set of dimensions it is the quality restricted to that sub-space. Pass $\text{ind.cos2}/\text{var.cos2}$ explicitly if you have the exact values.

tidymodels (recipe / workflow). The recipe and workflow methods extract a PCA fitted with `recipes::step_pca()` through the public recipes/workflows API: the scores from the baked training data (or, for a fitted workflow, `workflows::extract_mold()`), the loadings from `tidy(step, type = "coef")`, and the *full* set of eigenvalues from `tidy(step, type = "variance")` (so the scree plot and axis percentages are honest even when `num_comp` keeps only a few components). Variable coordinates are loading times the square root of the eigenvalue. Exact variable-component correlations and cos2 are recovered from the full PCA inertia when every PCA input is provably centered. When those metrics cannot be recovered (for example, for a bare `step_pca()` with no centering or a zero-inertia variable), their entries in `get_pca_var()` are NULL. Scores, eigenvalues, variable coordinates and contributions are still returned, so the individual, scree, variable-arrow and biplot displays remain available; correlation/cos2-dependent displays fail with an explicit unavailable-metric error. The correlation circle is omitted and a warning explains how to recover the metrics. `scale.unit` is set to TRUE only when every PCA input is both centered and unit-scaled at the PCA boundary; partial scaling therefore does not draw a correlation circle. For fully normalized data, variable coordinates, correlations, cos2 and contributions, and the eigenvalue percentages match a full `FactoMineR::PCA()` regardless of how many components `step_pca()` keeps; the *individual* cos2 is computed over the retained components (it equals the full-space cos2 only when all components are kept). The two-dimensional plots (`fviz_pca_ind()`, `fviz_pca_var()`, `fviz_pca_biplot()`) need `num_comp >= 2`. The recipe must be prepped, and its single PCA step must be an unweighted `step_pca()` and must be the final recipe step. Case-weighted PCA is rejected because the adapter cannot yet propagate those weights into individual contributions. Later steps could transform the baked/mold PCA scores without updating the fitted PCA loadings or eigenvalues, so such recipes fail explicitly instead of returning internally inconsistent geometry. Non-linear embedding steps such as `step_umap()` have no eigenvalues and are rejected with a message. For the loadings display of a recipe PCA see also `learn_tidymodels::plot_top_loadings()`.

Value

An object of class `c("factoextra_pca", "list")` holding the standardized ind (and var) results and eigenvalues, ready for the `fviz_pca_*`() functions.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

For UMAP / t-SNE embeddings, which have no eigenvalues, use `fviz_umap` / `fviz_tsne` instead. Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

Examples

```
# 1. Bring your own coordinates: classical MDS (cmdscale) -> factoextra
```

```

d <- dist(scale(mtcars))
mds <- cmdscale(d, k = 3)
obj <- as_factoextra_pca(ind.coord = mds)
fviz_pca_ind(obj, repel = TRUE)
fviz_eig(obj)

# 2. Round-trip a prcomp result through the constructor (biplot)
pca <- prcomp(iris[, -5], scale. = TRUE)
obj2 <- as_factoextra_pca(
  ind.coord = pca$x,
  var.coord = sweep(pca$rotation, 2, pca$sdev, "*"),
  eig        = pca$sdev^2,
  scale.unit = TRUE
)
fviz_pca_biplot(obj2, label = "var", col.ind = "steelblue")

# 3. A tidymodels recipe PCA (step_pca) -> factoextra biplot + honest scree
if (requireNamespace("recipes", quietly = TRUE)) {
  library(recipes)
  rec <- recipe(~ ., data = iris[, 1:4]) |>
    step_normalize(all_numeric_predictors()) |>
    step_pca(all_numeric_predictors(), num_comp = 4)
  obj3 <- as_factoextra_pca(prepare(rec))
  fviz_pca_biplot(obj3, label = "var")
  fviz_eig(obj3, addlabels = TRUE)
}

```

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)

```

 deprecated

Deprecated Functions

Description

Deprecated functions. Will be removed in a future version.

- `get_mfa_quanti_var()`. Deprecated. Use `get_mfa_var(res.mfa, "quanti.var")` instead.
- `get_mfa_quali_var()`. Deprecated. Use `get_mfa_var(res.mfa, "quali.var")` instead.
- `get_mfa_group()`. Deprecated. Use `get_mfa_var(res.mfa, "group")` instead.
- `fviz_mfa_ind_starplot()`: Star graph of individuals (draws partial points). Deprecated. Use `fviz_mfa_ind(res.mfa, partial = "all")` instead.
- `fviz_mfa_quanti_var()`: Graph of quantitative variables. Deprecated. Use `fviz_mfa(X, "quanti.var")` instead.
- `fviz_mfa_quali_var()`: Graph of qualitative variables. Deprecated. Use `fviz_mfa(X, "quali.var")` instead.
- `get_hmfa_quanti_var()`. Deprecated. Use `get_hmfa_var(res.hmfa, "quanti.var")` instead.
- `get_hmfa_quali_var()`. Deprecated. Use `get_hmfa_var(res.hmfa, "quali.var")` instead.
- `get_hmfa_group()`. Deprecated. Use `get_hmfa_var(res.hmfa, "group")` instead.
- `fviz_hmfa_ind_starplot()`: Graph of partial individuals. Deprecated. Use `fviz_hmfa_ind(X, partial = "all")` instead.

- `fviz_hmfa_quanti_var()`: Graph of quantitative variables. Deprecated. Use `fviz_hmfa_var(X, "quanti.var")` instead.
- `fviz_hmfa_quali_var()`: Graph of qualitative variables. Deprecated. Use `fviz_hmfa_var(X, "quali.var")` instead.
- `fviz_hmfa_group()`: Graph of the groups representation. Deprecated. Use `fviz_hmfa_var(X, "group")` instead.

Usage

```
get_mfa_quanti_var(res.mfa)

get_mfa_quali_var(res.mfa)

get_mfa_group(res.mfa)

fviz_mfa_ind_starplot(X, ...)

fviz_mfa_group(X, ...)

fviz_mfa_quanti_var(X, ...)

fviz_mfa_quali_var(X, ...)

get_hmfa_quanti_var(res.hmfa)

get_hmfa_quali_var(res.hmfa)

get_hmfa_group(res.hmfa)

fviz_hmfa_quanti_var(X, ...)

fviz_hmfa_quali_var(X, ...)

fviz_hmfa_ind_starplot(X, ...)

fviz_hmfa_group(X, ...)
```

Arguments

<code>res.mfa</code>	an object of class MFA [FactoMineR].
<code>X</code>	an object of class MFA or HMFA [FactoMineR].
<code>...</code>	Other arguments.
<code>res.hmfa</code>	an object of class HMFA [FactoMineR].

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

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: [Clustering Distance Measures in R](#).

- `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

When `stand = TRUE`, scaling that produces non-finite values is rejected. `fviz_dist()` also validates that supplied distance objects contain only finite values before plotting.

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

<code>x</code>	a numeric matrix or a data frame.
<code>method</code>	the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman" or "kendall".
<code>stand</code>	logical value; default is FALSE. If TRUE, then the data will be standardized using the function <code>scale()</code> . Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's standard deviation. If scaling produces NA values, <code>get_dist()</code> stops with a package-level error.
<code>...</code>	other arguments to be passed to the function <code>dist()</code> when using <code>get_dist()</code> .
<code>dist.obj</code>	an object of class "dist" as generated by the function <code>dist()</code> or <code>get_dist()</code> , containing only finite dissimilarity values.
<code>order</code>	logical value. if TRUE the ordered dissimilarity image (ODI) is shown.
<code>show_labels</code>	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](#). Online tutorial: [Clustering Distance Measures in R](#).

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

Provides a convenient workflow for clustering analyses and ggplot2-based data visualization. When `k = NULL`, the gap statistic selects the number of clusters. Hierarchical backends may validly return `k = 1`; in that case `eclust()` returns a one-cluster result without silhouette information. Read more: [Enhanced Cluster Analysis in R with factoextra's eclust\(\)](#).

Usage

```
eclust(
  x,
  FUNcluster = c("kmeans", "pam", "clara", "fanny", "hclust", "agnes", "diana",
    "hkmeans"),
  k = NULL,
  k.max = 10,
  stand = FALSE,
  graph = TRUE,
  hc_metric = "euclidean",
  hc_method = "ward.D2",
```

```

gap_maxSE = list(method = "firstSEmax", SE.factor = 1),
nboot = 100,
verbose = interactive(),
seed = 123,
...
)

```

Arguments

x	numeric vector, data matrix or data frame. For hierarchical clustering (FUNcluster = "hclust", "agnes" or "diana"), a precomputed dissimilarity matrix (an object of class "dist") may be supplied directly; in that case hc_metric is ignored and k must be specified. This allows custom distances such as Bray-Curtis (e.g. <code>vegan::vegdist(df, "bray")</code>).
FUNcluster	a clustering function including "kmeans", "pam", "clara", "fanny", "hkmeans", "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. For hierarchical clustering, this automatic selection may return k = 1. 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 <code>scale()</code> . Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's standard deviation. If scaling produces NA values, <code>eclust()</code> stops with a package-level error.
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 <code>dist()</code> [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". Ignored when x is already a "dist" object.
hc_method	the agglomeration method to be used (?hclust): "ward.D", "ward.D2", "single", "complete", "average", ...
gap_maxSE	a list containing the method and SE.factor parameters passed to <code>maxSE</code> . The default is <code>list(method = "firstSEmax", SE.factor = 1)</code> .
nboot	integer, number of Monte Carlo ("bootstrap") samples. Used only for determining the number of clusters using the gap statistic.
verbose	logical value. If TRUE, progress information 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 also includes:

- cluster: the cluster assignment of observations after cutting the tree
- nbclust: the number of clusters
- silinfo: the silhouette information of observations, when available for solutions with at least two clusters, 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 methods for `fviz_silhouette()`, `fviz_dend()`, and `fviz_cluster()`.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[fviz_silhouette](#), [fviz_dend](#), [fviz_cluster](#). Online tutorials: [Enhanced Cluster Analysis in R with factoextra's eclust\(\)](#) and [Choosing the Best Clustering Algorithm in R](#).

Examples

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

# Enhanced k-means clustering
# nboot >= 500 is recommended
res.km <- eclust(df, "kmeans", nboot = 2)
# Silhouette plot
fviz_silhouette(res.km)
# Optimal number of clusters using gap statistics
res.km$nbclust
# Print result
res.km

## Not run:
# Enhanced hierarchical clustering
res.hc <- eclust(df, "hclust", nboot = 2) # compute hclust
fviz_dend(res.hc) # dendrogram
if (res.hc$nbclust > 1) fviz_silhouette(res.hc) # silhouette plot

## End(Not run)
```

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

Description

Eigenvalues correspond to the amount of the variation explained by each principal component (PC).

- `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_screplot()`: an alias of `fviz_eig()`

These functions support the results of Principal Component Analysis (PCA), Correspondence Analysis (CA), Multiple Correspondence Analysis (MCA), Factor Analysis of Mixed Data (FAMD), Multiple Factor Analysis (MFA) and Hierarchical Multiple Factor Analysis (HMFA) functions. `fviz_eig()` validates `ncp`, `parallel.iter`, and `parallel.seed` before plotting, accepting integer-like numeric values while still rejecting fractional inputs.

Recent versions of `FactoMineR::PCA()` may retain only the number of eigenvalues requested through its `ncp` argument. When the stored eigenvalues do not account for the result's recorded total inertia, `fviz_eig()` warns that the scree plot is incomplete. Refit the PCA with a sufficiently large `ncp` to plot the complete spectrum. `get_eig()` continues to return the eigenvalues stored in the object.

Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

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,  
  main = NULL,  
  xlab = NULL,  
  ylab = NULL,  
  ggtheme = theme_minimal(),  
  parallel = FALSE,
```

```

parallel.color = "red",
parallel.lty = "dashed",
parallel.iter = 100,
parallel.seed = NULL,
...
)

fviz_screplot(...)

```

Arguments

<code>X</code>	an object of class PCA, CA, MCA, FAMD, MFA, or HMFA [FactoMineR]; prcomp or princomp [stats]; factoextra_pca; dudi, pca, coa, acm, between, or within [ade4]; ca or mjca [ca]; correspondence [MASS]; or expoOutput [ExPosition].
<code>choice</code>	a text specifying the data to be plotted. Allowed values are "variance" or "eigenvalue".
<code>geom</code>	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.
<code>barfill</code>	fill color for bar plot.
<code>barcolor</code>	outline color for bar plot.
<code>linecolor</code>	color for line plot (when geom contains "line").
<code>ncp</code>	a single positive integer specifying the number of dimensions to be shown. Integer-like numeric values are accepted.
<code>addlabels</code>	logical value. If TRUE, labels are added at the top of bars or points showing the information retained by each dimension.
<code>hjust</code>	horizontal adjustment of the labels.
<code>main, xlab, ylab</code>	plot main and axis titles.
<code>ggtheme</code>	function, ggplot2 theme name. The default is set by each function's ggtheme argument; see the function usage for the actual default. Set ggtheme = NULL to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via theme_set(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
<code>parallel</code>	logical value. If TRUE, adds a Horn parallel-analysis threshold curve. Each point is the 95th percentile of the corresponding ordered eigenvalue from independently simulated reference data. Components whose eigenvalues exceed their component-specific thresholds are retained by the parallel-analysis rule. Correlation PCA uses standardized reference data; covariance PCA uses reconstructed marginal scales. This is available only when choice = "eigenvalue" and X is a sufficiently complete prcomp or princomp object. Mean centering is required; prcomp also requires retained scores. Custom scaling and incomplete covariance decompositions are rejected when their reference distribution cannot be reconstructed. Default is FALSE.
<code>parallel.color</code>	color of the parallel-analysis threshold curve. Default is "red".

<code>parallel.lty</code>	line type for the parallel-analysis curve. Default is "dashed".
<code>parallel.iter</code>	a single positive integer giving the number of iterations for parallel analysis simulation. Integer-like numeric values are accepted. Default is 100.
<code>parallel.seed</code>	NULL or a single non-negative integer seed for reproducible parallel analysis simulation. If NULL (default), the current RNG stream is used. Integer-like numeric values are accepted.
<code>...</code>	optional arguments to be passed to the function <code>ggpar</code> .

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_screplot()`): returns a ggplot2

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

`fviz_pca`, `fviz_ca`, `fviz_mca`, `fviz_mfa`, `fviz_hmfa`. Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

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, addlabels = TRUE, ylim = c(0, 85))

# 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")

# Parallel analysis (Horn's method) to determine the number of components
# Retain components whose eigenvalues exceed their component-specific points
fviz_eig(res.pca, choice = "eigenvalue", parallel = TRUE,
```

```

addlabels = TRUE, parallel.color = "red",
parallel.iter = 10, parallel.seed = 123)

## Not run:
# 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")

# 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")

## End(Not run)

```

factoextra_palette *Colorblind-safe color palette for factoextra plots*

Description

Returns a colorblind-safe categorical color palette as a character vector of hex codes, to pass explicitly to the palette argument of the `fviz_*`() functions (e.g. `fviz_cluster(res, palette = factoextra_palette("okabe"))`). The default is the Okabe-Ito color-universal-design palette, which stays distinguishable under the common forms of color-vision deficiency.

The palette is returned as a plain color vector (not a `ggplot2` scale), for the existing palette argument. Use it for *discrete* group coloring (habillage, a factor `col.ind`, or clusters), where it also tints the matching ellipse fills. For a *continuous* metric (e.g. `col.ind = "cos2"`) use `gradient.cols` instead — a categorical palette does not apply there. It introduces no global option and no hidden state. The colors are ordered so the highest-contrast hues come first; yellow, the palest on a white background, is placed late, so plots with a few groups stay crisp.

Read more: [ggplot2 Colours in R: Change Colours by Group](#).

Usage

```
factoextra_palette(palette = "okabe", n = NULL)
```

Arguments

palette	name of the palette. Currently "okabe" (alias "okabe-ito"), the Okabe-Ito color-universal-design set.
n	number of colors to return. NULL (default) returns the whole palette. If n is greater than the number of available colors the palette is recycled (with a message), since recycling defeats the distinguishability the palette is chosen for.

Details

The Okabe-Ito colors are those of `grDevices::palette.colors( palette = "Okabe-Ito")` (here reordered so the vivid hues lead, with grey and black last). They were designed by Masataka Okabe and Kei Ito as a color-universal-design set, and popularized for figures by Wong (2011).

Value

A character vector of hex color codes.

References

Okabe, M. and Ito, K. (2008). *Color Universal Design (CUD): How to make figures and presentations that are friendly to colorblind people*. <https://jfly.uni-koeln.de/color/>.

Wong, B. (2011). Points of view: Color blindness. *Nature Methods*, 8(6), 441. doi:10.1038/nmeth.1618.

See Also

[theme_factoextra](#). Online tutorial: [ggplot2 Colours in R: Change Colours by Group](#).

Examples

```
# A colorblind-safe categorical palette
factoextra_palette("okabe")
factoextra_palette("okabe", n = 3)

# Publication-grade recipe: colorblind-safe groups + a clean theme
data(iris)
km <- kmeans(scale(iris[, 1:4]), 3, nstart = 25)
fviz_cluster(km, data = iris[, 1:4],
             palette = factoextra_palette("okabe"),
             ggtheme = theme_factoextra())
```

factominer_category_map

Map FactoMineR category labels to legacy naming patterns

Description

Map FactoMineR category labels to legacy naming patterns

Usage

```
factominer_category_map(X, element = c("quali.var", "quali.sup", "var"))
```

Arguments

<code>X</code>	a FactoMineR object (MCA, MFA, FAMD, HMFA).
<code>element</code>	element to map. Use "var" for MCA categories or "quali.var" for MFA/FAMD/HMFA qualitative categories. "quali.sup" maps supplementary qualitative categories when available.

Value

A data.frame with current labels, variable names, levels, and legacy naming patterns.

Examples

```
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  data(poison)
  res.mca <- FactoMineR::MCA(poison, quanti.sup = 1:2, quali.sup = 3:4, graph = FALSE)
  head(factominer_category_map(res.mca, element = "var"))
}
```

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), Factor Analysis of Mixed Data (FAMD), Multiple Factor Analysis (MFA) and Hierarchical Multiple Factor Analysis (HMFA) functions from several packages. Axis indices are validated before extraction, and MCA quantitative supplementary summaries inherit the package-level error raised when that result is unavailable.

Usage

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

Arguments

<code>X</code>	an object of class PCA, CA, MCA, FAMD, MFA and HMFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package]; expoOutput [ExPosition].
<code>element</code>	the element to subset from the output. Possible values are "row" or "col" for CA; "var", "ind", "mca.cor" or "quanti.sup" for MCA; "var" or "ind" for PCA; and 'quanti.var', 'quali.var', 'quali.sup', 'group' or 'ind' for FAMD, MFA and HMFA.
<code>node.level</code>	a single number indicating the HMFA node level.
<code>group.names</code>	a vector containing the name of the groups (by default, NULL and the groups are named group.1, group.2, and so on).
<code>result</code>	the result to be extracted for the element. Possible values are any combination of c("coord", "cos2", "contrib").
<code>axes</code>	a numeric vector specifying the axes of interest. Values must be positive integer indices within the available dimensions. Default values are 1:2 for axes 1 and 2.
<code>select</code>	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 contributions 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 $x^2 + y^2 + \dots$; x, y, ... are the coordinates of the points on the specified axes.

Value

A data frame containing the requested coordinates and the cos2 and contribution metrics aggregated over the requested axes.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

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)
# Summarize quantitative supplementary variables on axes 1:2
res <- facto_summarize(res.mca, "quanti.sup", 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 categorical 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)

```

Description

Generic function to create a scatter plot of multivariate analysis outputs, including PCA, CA, MCA and MFA.

Usage

```

fviz(
  X,
  element,
  axes = c(1, 2),
  geom = "auto",
  label = "all",
  invisible = "none",
  labelsize = 4,
  pointsize = 1.5,
  pointshape = 19,
  arrowsize = 0.5,
  arrow.linetype = "solid",
  habillage = "none",
  addEllipses = FALSE,
  ellipse.level = 0.95,
  ellipse.type = "norm",
  ellipse.alpha = 0.1,
  mean.point = TRUE,

```

```

color = "black",
fill = "white",
alpha = 1,
gradient.cols = NULL,
col.row.sup = "darkblue",
col.col.sup = "darkred",
select = list(name = NULL, cos2 = NULL, contrib = NULL),
title = NULL,
axes.linetype = "dashed",
repel = FALSE,
col.circle = "grey70",
circlesize = 0.5,
add.circle = NULL,
rotate.labels = FALSE,
ggtheme = theme_minimal(),
ggp = NULL,
font.family = "",
max.points = NULL,
sample.seed = 123,
...
)

```

Arguments

X	an object of class PCA, CA, MCA, FAMD, MFA and HMFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package]; expoOutput [ExPosition].
element	the element to subset from the output. Possible values are "row" or "col" for CA; "var", "ind", "mca.cor" or "quanti.sup" for MCA; "var" or "ind" for PCA; and 'quanti.var', 'quali.var', 'quali.sup', 'group' or 'ind' for FAMD, MFA and HMFA.
axes	a numeric vector specifying the axes of interest. Values must be positive integer indices within the available dimensions. Default values are 1:2 for axes 1 and 2.
geom	a text specifying the geometry to be used for the graph. Default value is "auto". 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 "all", "none", or a combination of c("ind", "ind.sup", "quali", "var", "quanti.sup", "group.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 "all", "none", or a combination of c("ind", "ind.sup", "quali", "var", "quanti.sup", "group.sup").
labelsize	font size for the labels

<code>pointsize</code>	the size of points
<code>pointshape</code>	the shape of points
<code>arrowsize</code>	the size of arrows. Controls the thickness of arrows.
<code>arrow.linetype</code>	linetype of the variable arrows (e.g. "solid", "dashed", "dotted"). Default is "solid".
<code>habillage</code>	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).
<code>addEllipses</code>	logical value. If TRUE, draws ellipses around the individuals when habillage != "none".
<code>ellipse.level</code>	the size of the concentration ellipse in normal probability.
<code>ellipse.type</code>	Character specifying frame type. Possible values are "convex", "confidence" or types supported by <code>stat_ellipse()</code> including one of <code>c("t", "norm", "euclid")</code> for plotting concentration ellipses. • "convex": plot convex hull of a set of points. • "confidence": plot confidence ellipses around group mean points as <code>coord.ellipse()</code> [in FactoMineR]. • "t": assumes a multivariate t-distribution. • "norm": assumes a multivariate normal distribution. • "euclid": draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless <code>coord_fixed()</code> is applied.
<code>ellipse.alpha</code>	Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.
<code>mean.point</code>	logical value. If TRUE (default), group mean points are added to the plot.
<code>color</code>	color to be used for the specified geometries (point, text). Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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".
<code>fill</code>	same as the argument color, but for point fill color. Useful when pointshape = 21, for example.
<code>alpha</code>	controls the transparency of individual and variable colors, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values also include "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".
<code>gradient.cols</code>	vector of colors to use for n-colour gradient. Allowed values include brewer and ggsci color palettes.

<code>col.col.sup</code> , <code>col.row.sup</code>	colors for the supplementary column and row points, respectively.
<code>select</code>	a selection of individuals/variables to be drawn. Allowed values are NULL or a list containing the arguments <code>name</code>, <code>cos2</code> or <code>contrib</code>: • <code>name</code>: is a character vector containing individuals/variables to be drawn • <code>cos2</code>: if <code>cos2</code> is in $[0, 1]$, ex: 0.6, then individuals/variables with a <code>cos2</code> > 0.6 are drawn. if <code>cos2</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>cos2</code> are drawn. • <code>contrib</code>: if <code>contrib</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>contrib</code> are drawn • <code>union</code>: logical. When several of <code>name</code>/<code>cos2</code>/<code>contrib</code> are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-<code>cos2</code> ones.
<code>title</code>	the title of the graph
<code>axes.linetype</code>	linetype of x and y axes.
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>col.circle</code>	a color for the correlation circle. Used only when X is a PCA output.
<code>circlesize</code>	the size of the variable correlation circle.
<code>add.circle</code>	logical or NULL controlling the variable correlation circle. Default NULL shows it only when meaningful: for PCA variables on unit-variance (scaled) data, and for quantitative variables of MCA/MFA/HMFA/FAMD. Use <code>add.circle = TRUE</code> to force the circle (e.g. a <code>prcomp(scale = FALSE)</code> fit on data you scaled manually) or <code>add.circle = FALSE</code> to suppress it.
<code>rotate.labels</code>	logical. If TRUE, the text labels of the plotted element are rotated to the angle of their arrows (<code>ggbiplot</code> style); labels in the left half-plane are flipped to stay upright. Default is FALSE (no rotation). Use together with <code>repel = FALSE</code> , as <code>ggrepel</code> ignores the label angle. Most useful for variable plots/biplots (e.g. <code>fviz_pca_var</code> , <code>fviz_pca_biplot</code>).
<code>ggtheme</code>	function, <code>ggplot2</code> theme name. The default is set by each function's <code>ggtheme</code> argument; see the function usage for the actual default. Set <code>ggtheme = NULL</code> to skip applying a <code>ggpubr</code> theme, so the plot keeps <code>ggplot2</code> default theme or the theme set globally via <code>theme_set()</code> . Allowed values include <code>ggplot2</code> official themes: <code>theme_gray()</code> , <code>theme_bw()</code> , <code>theme_minimal()</code> , <code>theme_classic()</code> , <code>theme_void()</code> , ...
<code>ggp</code>	a <code>ggplot</code> . If not NULL, points are added to an existing plot.
<code>font.family</code>	character vector specifying font family.
<code>max.points</code>	integer or NULL. When the individual / row / column cloud has more than <code>max.points</code> points, a random subset of that many <i>points</i> is drawn so the plot stays readable (usable labels instead of an over-plotted cloud). Only the drawn points/labels are thinned: any ellipse (<code>addEllipses</code>) and the group mean point are still computed on the full data, so a convex / confidence frame is not shrunk

or inflated by the draw. When points are coloured/split by a group (e.g. `habillage`), the draw is **stratified** so every group keeps at least a minimum number of points and none is decimated. When colour, fill or size is mapped to a continuous metric (e.g. `col.ind = "cos2"`), its scale and legend are pinned to the full-data range, so a point's colour, fill and size do not depend on how many points are drawn. A message reports how many points are shown. The subset is reproducible (see `sample.seed`) and does not change the caller's random stream. `NULL` (default) draws every point.

`sample.seed` the random seed used to pick the `max.points` subset, for a reproducible figure. Ignored when `max.points` is `NULL`.

... Arguments to be passed to the functions `ggpubr::ggscatter()` & `ggpubr::ggpar()`.

Value

a `ggplot`

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Examples

```
# Principal component analysis
# ++++++
data(decathlon2)
decathlon2.active <- decathlon2[1:23, 1:10]
res.pca <- prcomp(decathlon2.active, scale = TRUE)
fviz(res.pca, "ind") # Individuals plot
fviz(res.pca, "var") # Variables plot

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

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

fviz(res.mca, "ind") # Individuals plot
fviz(res.mca, "var") # Variables plot
```

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",
  repel = FALSE,
  font.family = "",
  ...
)
```

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"
repel	logical; whether to use ggrepel to avoid overplotting text labels. The old jitter argument is kept for backward compatibility and is converted to repel = TRUE with a deprecation warning.
font.family	character vector specifying font family.
...	Additional arguments, not used

Value

a ggplot2 plot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

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`], `ca` [in `ca`], `coa` [in `ade4`], `correspondence` [in `MASS`] and `expoOutput/epCA` [in `ExPosition`]. Read more: [Correspondence Analysis \(CA\) in R: Contingency Tables, Biplot & Interpretation](#).

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

Arguments

<code>X</code>	an object of class CA [FactoMineR], ca [ca], coa [ade4]; correspondence [MASS] and expoOutput/epCA [ExPosition].
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	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.
<code>geom.row, geom.col</code>	as geom but for row and column elements, respectively. Default is geom.row = c("point", "text"), geom.col = c("point", "text").
<code>shape.row, shape.col</code>	the point shapes to be used for row/column variables. Default values are 19 for rows and 17 for columns.
<code>map</code>	character string specifying the map type. Allowed options include: "symmetric", "rowprincipal", "colprincipal", "sympbiplot", "rowgab", "colgab", "rowgreen" and "colgreen". See details
<code>repel</code>	logical; whether to use ggrepel to avoid overplotting text labels. The old jitter argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>...</code>	Additional arguments. • in <code>fviz_ca_row()</code> and <code>fviz_ca_col()</code>: Additional arguments are passed to the functions <code>fviz()</code> and <code>ggpubr::ggpar()</code>. • in <code>fviz_ca_biplot()</code> and <code>fviz_ca()</code>: Additional arguments are passed to <code>fviz_ca_row()</code> and <code>fviz_ca_col()</code>.
<code>col.col, col.row</code>	color for column/row points. The default values are "red" and "blue", respectively. Can be a continuous variable or a factor variable. Allowed values also include "cos2", "contrib", "coord", "x", and "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")
<code>col.col.sup, col.row.sup</code>	colors for the supplementary column and row points, respectively.
<code>alpha.col, alpha.row</code>	controls the transparency of colors. The value ranges from 0 (total transparency) to 1 (no transparency). Default value is 1. Allowed values also include "cos2", "contrib", "coord", "x", and "y", as for the arguments <code>col.col</code> and <code>col.row</code> .
<code>select.col, select.row</code>	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 • union: logical. When several of name/cos2/contrib are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-cos2 ones.
label	a character vector specifying the elements to be labelled. Default value is "all". Allowed values are "all", "none", or a 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 "all", "none", or a combination of c("row", "row.sup", "col", "col.sup").
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 (M)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 ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

[get_ca](#), [fviz_pca](#), [fviz_mca](#). Online tutorial: [Correspondence Analysis \(CA\) in R: Contingency Tables, Biplot & Interpretation](#).

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)

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

# Asymmetric biplot, use arrows for columns
fviz_ca_biplot(res.ca, map = "rowprincipal",
  arrows = 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")

# 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))

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

# Control automatically the color of row points
# using the "cos2" or the contributions "contrib"
# cos2 = the quality of the rows on the factor map
# Change gradient color
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_ca_row(res.ca, col.row = "cos2",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
```

```

    repel = TRUE)

# You can also control the transparency
# of the color by the "cos2" or "contrib"
fviz_ca_row(res.ca, alpha.row="contrib")

# 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
# ++++++

# Control colors using their contributions
fviz_ca_col(res.ca, col.col = "contrib",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))

# 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))

```

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 $\text{ncol}(\text{data}) > 2$. An ellipse is drawn around each cluster. When `stand = TRUE`, the plotting data must remain finite after scaling.

Read more: [K-Means Clustering in R: Algorithm, Visualization & Interpretation](#).

Usage

```
fviz_cluster(
  object,
```

```

data = NULL,
choose.vars = NULL,
stand = TRUE,
axes = c(1, 2),
geom = c("point", "text"),
repel = FALSE,
show.clust.cent = TRUE,
ellipse = TRUE,
ellipse.type = "convex",
ellipse.level = 0.95,
ellipse.alpha = 0.2,
shape = NULL,
pointsize = 1.5,
labelsize = 12,
main = "Cluster plot",
xlab = NULL,
ylab = NULL,
outlier.color = "black",
outlier.shape = 19,
outlier.pointsize = pointsize,
outlier.labelsize = labelsize,
ggtheme = theme_grey(),
max.points = NULL,
sample.seed = 123,
...
)

```

Arguments

object	an object of class "partition" created by <code>pam()</code> , <code>clara()</code> , or <code>fanny()</code> [cluster]; "kmeans" [stats]; "dbscan" [fpc]; "Mclust" [mclust]; or "hkmeans" or "eclust" [factoextra]. A custom list may instead contain data plus either cluster or clustering. When the assignments and the data both carry complete, unique, matching names, the assignments are aligned to the data rows by name; otherwise they are used in their given (positional) order. The exception is a clustering fitted from dissimilarities and plotted with external data: complete, unique, non-matching name sets are rejected because positional use would attach clusters to the wrong observations.
data	the data used for clustering. It is required for kmeans and dbscan objects, and for partition or hcut objects fitted from dissimilarities when those objects do not retain the original observations.
choose.vars	a character vector containing variables to be considered for plotting.
stand	logical value; if TRUE, data is standardized before principal component analysis. If scaling produces NA values, <code>fviz_cluster()</code> stops with a package-level error.
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 <code>c("point", "text")</code> . Use "point" (to show only points); "text" to show only labels; <code>c("point", "text")</code> to show both types.
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>show.clust.cent</code>	logical; if TRUE, shows cluster centers
<code>ellipse</code>	logical value; if TRUE, draws outline around points of each cluster
<code>ellipse.type</code>	Character specifying frame type. Possible values are 'convex', 'confidence' or types supported by stat_ellipse including one of <code>c("t", "norm", "euclid")</code> .
<code>ellipse.level</code>	the size of the concentration ellipse in normal probability. Passed for <code>ggplot2::stat_ellipse</code> 's level. Ignored in 'convex'. Default value is 0.95.
<code>ellipse.alpha</code>	Alpha for frame specifying the transparency level of fill color. Use <code>alpha = 0</code> for no fill color.
<code>shape</code>	the shape of points.
<code>pointsize</code>	the size of points
<code>labelsize</code>	font size for the labels
<code>main</code>	plot main title.
<code>xlab, ylab</code>	character vector specifying x and y axis labels, respectively. Use <code>xlab = FALSE</code> and <code>ylab = FALSE</code> to hide xlab and ylab, respectively.
<code>outlier.pointsize, outlier.color, outlier.shape, outlier.labelsize</code>	arguments for customizing outliers, which can be detected only in DBSCAN clustering.
<code>ggtheme</code>	function, ggplot2 theme name. The default is set by each function's <code>ggtheme</code> argument; see the function usage for the actual default. Set <code>ggtheme = NULL</code> to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via <code>theme_set()</code> . Allowed values include ggplot2 official themes: <code>theme_gray()</code> , <code>theme_bw()</code> , <code>theme_minimal()</code> , <code>theme_classic()</code> , <code>theme_void()</code> , ...
<code>max.points</code>	integer or NULL. When the data has more than <code>max.points</code> observations, a random subset of that many <i>points</i> is drawn so the cluster plot stays readable instead of over-plotting. Only the drawn scatter/labels are thinned: the cluster frame (convex hull / ellipse) and centres are still computed on the full data, so a convex hull is not shrunk by dropping the extreme points a random draw tends to lose. The draw is stratified by cluster so every cluster keeps at least a minimum number of points, and a message reports how many are shown. Any DBSCAN/Mclust outliers are always drawn. The subset is reproducible (see <code>sample.seed</code>) and does not change the caller's random stream. NULL (default) draws every observation.
<code>sample.seed</code>	the random seed used to pick the <code>max.points</code> subset, for a reproducible figure. Ignored when <code>max.points</code> is NULL.
<code>...</code>	other arguments to be passed to the functions ggscatter and ggpar .

Value

a ggplot2 object.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[fviz_silhouette](#), [hcut](#), [hkmeans](#), [eclust](#), [fviz_dend](#). Online tutorial: [K-Means Clustering in R: Algorithm, Visualization & Interpretation](#).

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 = 10)

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

# Change the color palette and theme
fviz_cluster(km.res, iris[, -5],
  palette = "Set2", ggtheme = 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
# ++++++
requireNamespace("cluster", quietly = TRUE)
pam.res <- cluster::pam(iris.scaled, 3)
# Visualize pam clustering
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm")

# Hierarchical clustering
# ++++++
# Use hcut(), which computes hclust and cuts 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, ellipse.type = "convex")

## End(Not run)

```

fviz_contrib

Visualize the contributions of row/column elements

Description

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

Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret.](#)

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,
  xtckslab.rt = 45,
  ggtheme = theme_minimal(),
  display = c("bar", "heatmap"),
  ...
)

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

```

Arguments

<code>X</code>	an object of class PCA, CA, MCA, FAMD, MFA and HMFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package].
<code>choice</code>	allowed values are "row" and "col" for CA; "var" and "ind" for PCA or MCA; "var", "ind", "quanti.var", "quali.var" and "group" for FAMD, MFA and HMFA.
<code>axes</code>	a numeric vector specifying the dimension(s) of interest.
<code>fill</code>	a fill color for the bar plot.
<code>color</code>	an outline color for the bar plot.
<code>sort.val</code>	a string specifying whether the value should be sorted. Allowed values are "none" (no sorting), "asc" (for ascending) or "desc" (for descending).
<code>top</code>	a numeric value specifying the number of top elements to be shown.
<code>xtickslab.rt</code>	rotation angle for x axis tick labels. Default is 45 degrees.
<code>ggtheme</code>	function, ggplot2 theme name. The default is set by each function's ggtheme argument; see the function usage for the actual default. Set ggtheme = NULL to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via theme_set(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
<code>display</code>	how to display the values. "bar" (default) draws the usual barplot of the cos2/contribution summed over axes. "heatmap" draws a grid with one tile per element and dimension, filled by the per-dimension cos2/contribution and labelled with its value, so several dimensions can be read at once. With "heatmap", elements are ordered by their (unweighted) total over the requested axes and top keeps the leading ones; the bar-specific sort.val and color arguments are ignored, and fill sets the high end of the white-to-colour gradient.
<code>...</code>	other arguments to be passed to the function ggpar .
<code>sortcontrib</code>	see the argument sort.val

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 were 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

<https://www.datanovia.com/learn/>

See Also

[fviz_cos2](#), [get_pca](#). Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

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, top = 10 )

# 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)

# Heat-grid of contributions across several dimensions
fviz_contrib(res.pca, choice = "var", axes = 1:3, display = "heatmap")

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

## Not run:
# 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 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)
# Visualize variable category 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, top = 20)
# Visualize categorical variable category contributions on axes 1
fviz_contrib(res.mfa, choice = "quali.var", axes = 1)

## End(Not run)
```

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), Factor Analysis of Mixed Data (FAMD), Multiple Factor Analysis (MFA) and Hierarchical Multiple Factor Analysis (HMFA) functions.

Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

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,
    xtickslab.rt = 45,
    ggtheme = theme_minimal(),
    display = c("bar", "heatmap"),
    ...
  )

```

Arguments

<code>x</code>	an object of class PCA, CA, MCA, FAMD, MFA and HMFA [FactoMineR]; prcomp and princomp [stats]; dudi, pca, coa and acm [ade4]; ca [ca package].
<code>choice</code>	allowed values are "row" and "col" for CA; "var" and "ind" for PCA or MCA; "var", "ind", "quanti.var", "quali.var" and "group" for FAMD, MFA and HMFA.
<code>axes</code>	a numeric vector specifying the dimension(s) of interest.
<code>fill</code>	a fill color for the bar plot.
<code>color</code>	an outline color for the bar plot.
<code>sort.val</code>	a string specifying whether the value should be sorted. Allowed values are "none" (no sorting), "asc" (for ascending) or "desc" (for descending).
<code>top</code>	a numeric value specifying the number of top elements to be shown.
<code>xtickslab.rt</code>	rotation angle for x axis tick labels. Default is 45 degrees.
<code>ggtheme</code>	function, ggplot2 theme name. The default is set by each function's ggtheme argument; see the function usage for the actual default. Set ggtheme = NULL to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via theme_set(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
<code>display</code>	how to display the values. "bar" (default) draws the usual barplot of the cos2/contribution summed over axes. "heatmap" draws a grid with one tile per element and dimension, filled by the per-dimension cos2/contribution and labelled with its value, so several dimensions can be read at once. With "heatmap", elements are ordered by their (unweighted) total over the requested axes and top keeps the leading ones; the bar-specific sort.val and color arguments are ignored, and fill sets the high end of the white-to-colour gradient.
<code>...</code>	not used

Value

a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

`fviz_contrib`, `get_pca`. Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret.](#)

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, top = 10 )

# Change color
fviz_cos2(res.pca, choice="var", axes = 1,
          fill = "lightgray", color = "black")

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

# Heat-grid of cos2 across several dimensions
fviz_cos2(res.pca, choice = "var", axes = 1:4, display = "heatmap")

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

## Not run:
# Correspondence Analysis
# ++++++
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 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, top = 20)
# Visualize variable category 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
# Select the top 20
fviz_cos2(res.mfa, choice = "ind", axes = 1, top = 20)
# Visualize categorical variable category cos2 on axes 1
fviz_cos2(res.mfa, choice = "quali.var", axes = 1)

## End(Not run)
```

fviz_dend

Enhanced Visualization of Dendrogram

Description

Draws easily beautiful dendrograms using either R base plot or ggplot2. It also provides options for circular dendrograms and phylogenetic-style trees.

Read more: [Visualizing Dendrograms in R: Color, Zoom & Customize](#); to compare two trees see [Comparing Dendrograms in R: Tanglegrams & Correlation](#).

Usage

```
fviz_dend(
  x,
  k = NULL,
  h = NULL,
  k_colors = NULL,
  palette = NULL,
  show_labels = TRUE,
  color_labels_by_k = TRUE,
  match_coord_colors = FALSE,
  label_cols = NULL,
  labels_font = "plain",
  labels_track_height = NULL,
  repel = FALSE,
  lwd = 0.7,
  highlight = NULL,
```

```

highlight.col = "red",
highlight.lwd = NULL,
type = c("rectangle", "circular", "phylogenetic"),
phylo_layout = "layout.auto",
rect = FALSE,
rect_border = "gray",
rect_lty = 2,
rect_fill = FALSE,
lower_rect,
horiz = FALSE,
cex = 0.8,
main = "Cluster Dendrogram",
xlab = "",
ylab = "Height",
sub = NULL,
ggtheme = theme_classic(),
...
)

```

Arguments

x	an object of class dendrogram, hclust, agnes, diana, hcut, hkmeans or HCPC (FactoMineR).
k	the number of groups for cutting the tree.
h	a numeric value. Cut the dendrogram by cutting at height h. (k overrides h)
k_colors, palette	a vector containing colors to be used for the groups. It should contain k colors. Allowed values also include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickand-morty".
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.
match_coord_colors	logical value. Default is FALSE, where dendrogram colors follow the left-to-right leaf order. If TRUE, cluster colors are remapped to cluster-label order so they match <code>fviz_cluster()</code> and <code>fviz_silhouette()</code> for the same clustering.
label_cols	a vector containing the colors for labels.
labels_font	font face for the leaf labels of "rectangle"/"circular" dendrograms. One of "plain" (default), "bold", "italic" or "bold.italic". Default "plain" leaves labels unchanged.
labels_track_height	a positive numeric value for adjusting the room for the labels. Used only when type = "rectangle".

<code>repel</code>	logical value. Use <code>repel = TRUE</code> to avoid label overplotting when <code>type = "phylogenetic"</code> . The literal "phylogenetic" value is a historical API token retained for compatibility.
<code>lwd</code>	a numeric value specifying dendrogram branch and rectangle line width.
<code>highlight</code>	an optional character vector of leaf labels; the branches leading to those leaves are emphasized (thicker, and coloured <code>highlight.col</code>) while every other branch keeps its colour and width. <code>NULL</code> (default) highlights nothing. Has no effect for <code>type = "phylogenetic"</code> (that layout does not colour branch segments).
<code>highlight.col</code>	colour (name or hex) for the highlighted branches.
<code>highlight.lwd</code>	line width for the highlighted branches. <code>NULL</code> (default) uses $2 * lwd$ so the emphasis stands out by thickness regardless of colour; set <code>highlight.lwd = lwd</code> for colour-only emphasis.
<code>type</code>	type of plot. Allowed values are "rectangle", "circular", and the historical compatibility token "phylogenetic" for a phylogenetic-style tree.
<code>phylo_layout</code>	the layout used for phylogenetic-style trees. Default value is "layout.auto", which is kept as a compatibility alias for "layout_nicely". Allowed values include: <code>layout.auto</code> , <code>layout_nicely</code> , <code>layout_with_drl</code> , <code>layout_as_tree</code> , <code>layout.gem</code> , <code>layout_with_gem</code> , <code>layout.mds</code> , <code>layout_with_mds</code> and <code>layout_with_lgl</code> .
<code>rect</code>	logical value specifying whether to add a rectangle around groups. Used only when <code>k != NULL</code> .
<code>rect_border, rect_lty</code>	border color and line type for rectangles.
<code>rect_fill</code>	a logical value. If <code>TRUE</code> , fill the rectangle.
<code>lower_rect</code>	a value of how low should the lower part of the rectangle around clusters. Ignored when <code>rect = FALSE</code> .
<code>horiz</code>	a logical value. If <code>TRUE</code> , a horizontal dendrogram is drawn.
<code>cex</code>	size of labels
<code>main, xlab, ylab</code>	main and axis titles
<code>sub</code>	Plot subtitle. Default is <code>NULL</code> (no subtitle). Set to a character string to display a subtitle below the title, e.g. <code>sub = paste0("Method: ", "ward.D2")</code> .
<code>ggtheme</code>	function, ggplot2 theme name. Default value is <code>theme_classic()</code> . Allowed values include ggplot2 official themes: <code>theme_gray()</code> , <code>theme_bw()</code> , <code>theme_minimal()</code> , <code>theme_classic()</code> , <code>theme_void()</code> , ...
<code>...</code>	other arguments to be passed to the function <code>plot.dendrogram()</code>

Details

For branch styling beyond `highlight` - for example dashed branches or per-branch colours - pre-style a dendextend dendrogram and pass it to `fviz_dend()`, which honours its `set()` aesthetics:

```
library(dendextend)
dend <- as.dendrogram(hclust(dist(scale(USArrests))))
dend <- set(dend, "branches_lty", 2) # dashed branches
fviz_dend(dend)
```

Comparing two dendrograms. To compare two hierarchical clusterings of the same observations (e.g. different linkages), draw a *tanglegram* with `dendextend` (already a dependency): the two trees face each other, matched leaves are connected, and `entanglement()` measures agreement (0 = perfect, 1 = worst). This uses base graphics.

```
library(dendextend)
d1 <- as.dendrogram(hclust(dist(scale(USArrests)), "complete"))
d2 <- as.dendrogram(hclust(dist(scale(USArrests)), "average"))
d1 <- dendlist(d1, d2)
d1 <- untangle(d1, method = "step2side")          # reduce crossings
tanglegram(d1, common_subtrees_color_branches = TRUE)
entanglement(d1)                                # agreement, in [0, 1]
```

Both dendrograms must share the same leaf labels.

Value

an object of class `fviz_dend` which is a `ggplot` with the attributes "dendrogram" accessible using `attr(x, "dendrogram")`, where `x` is the result of `fviz_dend()`.

See Also

[tanglegram](#), [entanglement](#) for comparing two dendrograms (see **Details**). Online tutorials: [Visualizing Dendrograms in R: Color, Zoom & Customize](#) and [Comparing Dendrograms in R: Tanglegrams & Correlation](#).

Examples

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

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

# Default plot
fviz_dend(res.hc)

# Increase branch and rectangle line widths
fviz_dend(res.hc, lwd = 2)

# 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)

# 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)

# Phylogenetic-style tree layouts support both compatibility aliases and
# current igraph layout names
if (requireNamespace("igraph", quietly = TRUE)) {
  fviz_dend(res.hc, type = "phylogenetic", phylo_layout = "layout_nicely",
    show_labels = FALSE)
}

```

fviz_ellipses

Draw confidence ellipses around the categories

Description

Draw confidence ellipses around the categories

Usage

```

fviz_ellipses(
  X,
  habillage,
  axes = c(1, 2),
  addEllipses = TRUE,
  ellipse.type = "confidence",
  palette = NULL,
  pointsize = 1,
  geom = c("point", "text"),
  ggtheme = theme_bw(),
  ...
)

```

Arguments

X an object of class MCA, PCA or MFA.

habillage	a numeric vector of indexes of variables or a character vector of names of variables. Can be also a data frame containing grouping variables.
axes	a numeric vector specifying the axes of interest. Values must be positive integer indices within the available dimensions. Default values are 1:2 for axes 1 and 2.
addEllipses	logical value. If TRUE, draws ellipses around the individuals when habillage != "none".
ellipse.type	Character specifying frame type. Possible values are "convex", "confidence" or types supported by stat_ellipse() including one of c("t", "norm", "euclid") for plotting concentration ellipses. • "convex": plot convex hull of a set of points. • "confidence": plot confidence ellipses around group mean points as coord.ellipse() [in FactoMineR]. • "t": assumes a multivariate t-distribution. • "norm": assumes a multivariate normal distribution. • "euclid": draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless coord_fixed() is applied.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
pointsize	the size of points
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.
ggtheme	function, ggplot2 theme name. The default is set by each function's ggtheme argument; see the function usage for the actual default. Set ggtheme = NULL to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via theme_set() . Allowed values include ggplot2 of-ficial themes: theme_gray() , theme_bw() , theme_minimal() , theme_classic() , theme_void() , ...
...	Arguments to be passed to the functions ggpubr::ggscatter() & ggpubr::ggpar() .

Value

a ggplot

Author(s)Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Examples

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

fviz_ellipses(res.mca, 3:4, geom = "point",
              palette = "jco")
```

fviz_famd

Visualize Factor Analysis of Mixed Data

Description

Factor analysis of mixed data (FAMD) is a particular case of MFA, used to analyze a data set containing both quantitative and qualitative variables. `fviz_famd()` provides ggplot2-based elegant visualization of FAMD outputs from the R function: `FAMD` [FactoMineR].

- `fviz_famd_ind()`: Graph of individuals
- `fviz_famd_var()`: Graph of variables
- `fviz_famd()`: An alias of `fviz_famd_ind(res.famd)`

Read more: [Factor Analysis of Mixed Data \(FAMD\) in R: Compute, Visualize & Interpret.](#)

Usage

```
fviz_famd_ind(
  X,
  axes = c(1, 2),
  geom = c("point", "text"),
  repel = FALSE,
  habillage = "none",
  palette = NULL,
  addEllipses = FALSE,
  col.ind = "blue",
  col.ind.sup = "darkblue",
  alpha.ind = 1,
  shape.ind = 19,
  col.quali.var = "black",
  select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
```

```

    gradient.cols = NULL,
    ...
  )

fviz_famd_var(
  X,
  choice = c("var", "quanti.var", "quali.var", "quali.sup"),
  axes = c(1, 2),
  geom = c("point", "text"),
  repel = FALSE,
  col.var = "red",
  alpha.var = 1,
  shape.var = 17,
  col.var.sup = "darkgreen",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  ...
)

fviz_famd(X, ...)

```

Arguments

<code>X</code>	an object of class FAMD [FactoMineR].
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	a text specifying the geometry to be used for the graph. Allowed values are the combination of <code>c("point", "arrow", "text")</code> . Use "point" (to show only points); "text" to show only labels; <code>c("point", "text")</code> or <code>c("arrow", "text")</code> to show arrows and texts. Using <code>c("arrow", "text")</code> is sensible only for the graph of variables.
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>habillage</code>	an optional factor variable for coloring the observations by groups. Default value is "none". If <code>X</code> is a FAMD object from FactoMineR package, <code>habillage</code> can also specify the index of the factor variable in the data.
<code>palette</code>	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
<code>addEllipses</code>	logical value. If TRUE, draws ellipses around the individuals when <code>habillage != "none"</code> .
<code>col.ind, col.var</code>	color for individuals and variables, respectively. Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>col.ind.sup</code>	color for supplementary individuals
<code>alpha.ind, alpha.var</code>	controls the transparency of individuals and variables, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values also include "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 <code>habillage = "none"</code> .
<code>shape.ind, shape.var</code>	point shapes of individuals, variables, groups and axes
<code>col.quali.var</code>	color for qualitative variables in <code>fviz_famd_ind()</code> . Default is "black".
<code>select.ind, select.var</code>	a selection of individuals and 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 contributions are drawn • union: logical. When several of name/cos2/contrib are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-cos2 ones.
<code>gradient.cols</code>	vector of colors to use for n-colour gradient. Allowed values include brewer and ggsci color palettes.
<code>...</code>	Arguments to be passed to the function <code>fviz()</code>
<code>choice</code>	The graph to plot in <code>fviz_famd_var()</code> . Allowed values include one of <code>c("var", "quanti.var", "quali.var", "quali.sup")</code> .
<code>col.var.sup</code>	color for supplementary variables.

Value

a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[get_famd](#). Online tutorial: [Factor Analysis of Mixed Data \(FAMD\) in R: Compute, Visualize & Interpret](#).

Examples

```
# Compute FAMD
library("FactoMineR")
data(wine)
res.famd <- FAMD(wine[,c(1,2, 16, 22, 29, 28, 30,31)], graph = FALSE)
res.famd.sup <- FAMD(wine[,c(1,2, 16, 22, 29, 28, 30,31)],
                     sup.var = 2, graph = FALSE)

# Eigenvalues/variances of dimensions
fviz_screplot(res.famd)
# Graph of variables
fviz_famd_var(res.famd)
# Quantitative variables
fviz_famd_var(res.famd, "quanti.var", repel = TRUE, col.var = "black")
# Qualitative variables
fviz_famd_var(res.famd, "quali.var", col.var = "black")
# Supplementary qualitative variable categories
fviz_famd_var(res.famd.sup, "quali.sup", col.var = "darkgreen")
# Graph of individuals colored by cos2
fviz_famd_ind(res.famd, col.ind = "cos2",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE)
```

fviz_hmfa

Visualize Hierarchical Multiple Factor Analysis

Description

Hierarchical Multiple Factor Analysis (HMFA) is an extension of MFA, used in a situation where the data are organized into a hierarchical structure. `fviz_hmfa()` provides ggplot2-based elegant visualization of HMFA outputs from the R function: `HMFA` [FactoMineR].

- `fviz_hmfa_ind()`: Graph of individuals
- `fviz_hmfa_var()`: Graph of variables
- `fviz_hmfa_quali_biplot()`: Biplot of individuals and qualitative variables
- `fviz_hmfa()`: An alias of `fviz_hmfa_ind()`

Read more: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables.](#)

Usage

```
fviz_hmfa_ind(
  X,
  axes = c(1, 2),
```

```

    geom = c("point", "text"),
    repel = FALSE,
    habillage = "none",
    addEllipses = FALSE,
    shape.ind = 19,
    col.ind = "blue",
    col.ind.sup = "darkblue",
    alpha.ind = 1,
    select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
    partial = NULL,
    col.partial = "group",
    group.names = NULL,
    node.level = 1,
    ...
)

fviz_hmfa_var(
  X,
  choice = c("quanti.var", "quali.var", "group"),
  axes = c(1, 2),
  geom = c("point", "text"),
  repel = FALSE,
  col.var = "red",
  alpha.var = 1,
  shape.var = 17,
  col.var.sup = "darkgreen",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  ...
)

fviz_hmfa_quali_biplot(
  X,
  axes = c(1, 2),
  geom = c("point", "text"),
  repel = FALSE,
  habillage = "none",
  title = "Biplot of individuals and qualitative variables - HMFA",
  ...
)

fviz_hmfa(X, ...)

```

Arguments

<code>X</code>	an object of class HMFA [FactoMineR].
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	a text specifying the geometry to be used for the graph. Allowed values are the combination of <code>c("point", "arrow", "text")</code> . Use "point" (to show

	only points); "text" to show only labels; c("point", "text") or c("arrow", "text") to show arrows and texts. Using c("arrow", "text") is sensible only for the graph of variables.
repel	logical; whether to use ggrepel to avoid overplotting text labels. The old jitter argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
habillage	an optional factor variable for coloring the observations by groups. Default value is "none". If X is an HMFA 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".
shape.ind, shape.var	point shapes of individuals and variables, respectively.
col.ind, col.var	color for individuals, partial individuals and variables, respectively. Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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, partial individual and variable, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values also include "cos2", "contrib", "coord", "x", and "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".
select.ind, select.var	a selection of individuals and 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 $\text{cos2} > 0.6$ are drawn. if $\text{cos2} > 1$, 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 contributions are drawn • union: logical. When several of name/cos2/contrib are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-cos2 ones.
partial	list of the individuals for which the partial points should be drawn. (by default, partial = NULL and no partial points are drawn). Use partial = "all" to visualize partial points for all individuals.

<code>col.partial</code>	color for partial individuals. By default, points are colored according to the groups.
<code>group.names</code>	a vector containing the name of the groups (by default, NULL and the groups are named <code>group.1</code> , <code>group.2</code> , and so on).
<code>node.level</code>	a single number indicating the HMFA node level to plot.
<code>...</code>	Arguments to be passed to the function <code>fviz()</code> and <code>ggpubr::ggpar()</code>
<code>choice</code>	the graph to plot. Allowed values include one of <code>c("quanti.var", "quali.var", "group")</code> for plotting quantitative variables, qualitative variables and group of variables, respectively.
<code>col.var.sup</code>	color for supplementary variables.
<code>title</code>	the title of the graph

Value

a ggplot

Author(s)

Fabian Mundt <f.mundt@inventionate.de>

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

[get_hmfa](#), [fviz_mfa](#). Online tutorial: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables](#).

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
# ++++++
# 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")

# Color individuals by groups, add concentration ellipses
# Remove labels: label = "none".
```

```

# Change color palette to "jco". See ?ggpubr::ggpar
grp <- as.factor(wine[,1])
p <- fviz_hmfa_ind(res.hmfa, label="none", habillage=grp,
  addEllipses=TRUE, palette = "jco")
print(p)

# Graph of variables
# ++++++
# Quantitative variables
fviz_hmfa_var(res.hmfa, "quanti.var")
# Graph of categorical variable categories
fviz_hmfa_var(res.hmfa, "quali.var")
# Groups of variables (correlation square)
fviz_hmfa_var(res.hmfa, "group")

# Biplot of categorical variable categories and individuals
# ++++++
fviz_hmfa_quali_biplot(res.hmfa)

# Graph of partial individuals (starplot)
# ++++++
fviz_hmfa_ind(res.hmfa, partial = "all", col.partial = "black")

```

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 visualization of MCA outputs from the R functions: `MCA` [in `FactoMineR`], `acm` [in `ade4`], and `expoOutput/epMCA` [in `ExPosition`]. Read more: [Multiple Correspondence Analysis \(MCA\) in R: Compute, Visualize & Interpret](#).

- `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"),
geom.ind = geom,
repel = FALSE,
habillage = "none",
palette = NULL,
addEllipses = FALSE,
col.ind = "blue",
col.ind.sup = "darkblue",
alpha.ind = 1,
shape.ind = 19,
map = "symmetric",
select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
quanti.sup = FALSE,
col.quanti.sup = "#D55E00",
...
)

```

```

fviz_mca_var(
  X,
  choice = c("var.cat", "mca.cor", "var", "quanti.sup"),
  axes = c(1, 2),
  geom = c("point", "text"),
  geom.var = geom,
  repel = FALSE,
  col.var = "red",
  alpha.var = 1,
  shape.var = 17,
  col.quanti.sup = "blue",
  col.quali.sup = "darkgreen",
  map = "symmetric",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  ...
)

```

```

fviz_mca_biplot(
  X,
  axes = c(1, 2),
  geom = c("point", "text"),
  geom.ind = geom,
  geom.var = geom,
  repel = FALSE,
  label = "all",
  invisible = "none",
  habillage = "none",
  addEllipses = FALSE,
  palette = NULL,
  arrows = c(FALSE, FALSE),
  map = "symmetric",

```

```

    title = "MCA - Biplot",
    quanti.sup = FALSE,
    col.quanti.sup = "#D55E00",
    ...
)

fviz_mca(X, ...)

```

Arguments

<code>X</code>	an object of class MCA [FactoMineR], acm [ade4] and expoOutput/epMCA [ExPosition].
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	a text specifying the geometry to be used for the graph. Allowed values are the combination of <code>c("point", "arrow", "text")</code> . Use "point" (to show only points); "text" to show only labels; <code>c("point", "text")</code> or <code>c("arrow", "text")</code> to show arrows and texts. Using <code>c("arrow", "text")</code> is sensible only for the graph of variables.
<code>geom.ind, geom.var</code>	as <code>geom</code> but for individuals and variables, respectively. Default is <code>geom.ind = c("point", "text")</code> , <code>geom.var = c("point", "text")</code> .
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>habillage</code>	an optional factor variable for coloring the observations by groups. Default value is "none". If <code>X</code> is an MCA object from FactoMineR package, <code>habillage</code> can also specify the index of the factor variable in the data.
<code>palette</code>	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
<code>addEllipses</code>	logical value. If TRUE, draws ellipses around the individuals when <code>habillage != "none"</code> .
<code>col.ind, col.var</code>	color for individuals and variables, respectively. Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>col.ind.sup</code>	color for supplementary individuals
<code>alpha.ind, alpha.var</code>	controls the transparency of individual and variable colors, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value

is 1. Possible values also include "cos2", "contrib", "coord", "x", and "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"`.

<code>shape.ind, shape.var</code>	point shapes of individuals and variables.
<code>map</code>	character string specifying the map type. Allowed options include: "symmetric", "rowprincipal", "colprincipal", "sympbiplot", "rowgab", "colgab", "rowgreen" and "colgreen". See details
<code>select.ind, select.var</code>	a selection of individuals/variables to be drawn. Allowed values are NULL or a list containing the arguments <code>name</code>, <code>cos2</code> or <code>contrib</code>: • <code>name</code> is a character vector containing individuals/variables to be drawn • <code>cos2</code> if <code>cos2</code> is in $[0, 1]$, ex: 0.6, then individuals/variables with a <code>cos2</code> > 0.6 are drawn. if <code>cos2</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>cos2</code> are drawn. • <code>contrib</code> if <code>contrib</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>contrib</code> are drawn • <code>union</code>: logical. When several of <code>name/cos2/contrib</code> are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-<code>cos2</code> ones.
<code>quanti.sup</code>	logical. If TRUE, the supplementary quantitative variables of a FactoMineR MCA are overlaid on the individuals / biplot map as correlation arrows (see the Details section). Default FALSE leaves the map unchanged. Used by <code>fviz_mca_ind()</code> and <code>fviz_mca_biplot()</code> .
<code>col.quanti.sup, col.quali.sup</code>	a color for the quantitative/qualitative supplementary variables.
<code>...</code>	Additional arguments. • in <code>fviz_mca_ind()</code> and <code>fviz_mca_var()</code> (including <code>choice = "mca.cor"</code>): Additional arguments are passed to the functions <code>fviz()</code> and <code>ggpubr::ggpar()</code>. • in <code>fviz_mca_biplot()</code> and <code>fviz_mca()</code>: Additional arguments are passed to <code>fviz_mca_ind()</code> and <code>fviz_mca_var()</code>.
<code>choice</code>	the graph to plot. Allowed values include: i) "var" and "mca.cor" for plotting the correlation between variables and principal dimensions; ii) "var.cat" for variable categories and iii) "quanti.sup" for the supplementary quantitative variables.
<code>label</code>	a text specifying the elements to be labelled. Default value is "all". Allowed values are "all", "none", or a combination of <code>c("ind", "ind.sup", "var", "quali.sup", "quanti.sup")</code> . "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.
<code>invisible</code>	a text specifying the elements to be hidden on the plot. Default value is "none". Allowed values are "all", "none", or a combination of <code>c("ind", "ind.sup", "var", "quali.sup", "quanti.sup")</code> .

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 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. The argument "map" can be used to change the plot type. For more explanation, read the details section of fviz_ca documentation.

quanti.sup = TRUE overlays the supplementary quantitative variables on the individuals / biplot map. Each such variable is drawn as an arrow from the origin in the direction of its correlations with the shown dimensions (X\$quanti.sup\$coord), so the arrow points toward the region of the cloud where the variable takes larger values. Each arrow's length is proportional to the variable's absolute correlation with the shown dimensions (a correlation of 1 reaches about 80% of the individual-cloud extent), so a weak covariate draws a short arrow. The overlay labels are repelled by default (needs the ggrepel package). **Arrow lengths are relative** to the cloud, so compare directions and relative lengths rather than absolute sizes.

Value

a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[get_mca](#), [fviz_pca](#), [fviz_ca](#), [fviz_mfa](#), [fviz_hmfa](#). Online tutorial: [Multiple Correspondence Analysis \(MCA\) in R: Compute, Visualize & Interpret](#).

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: https://www.datanovia.com/learn/data-visualization/ggplot2/themes

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

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

## 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)

# Overlay supplementary quantitative variables as correlation arrows.
# Fit the MCA with quanti.sup, then set quanti.sup = TRUE when plotting.
res.mca2 <- MCA(poison, quanti.sup = 1:2, quali.sup = 3:4, graph = FALSE)
fviz_mca_ind(res.mca2, label = "none", habillage = poison$Vomiting,
  addEllipses = TRUE, quanti.sup = TRUE)

# Change group colors using RColorBrewer color palettes
# Read more: https://www.datanovia.com/learn/data-visualization/ggplot2/colors
p + scale_color_brewer(palette="Dark2") +
  scale_fill_brewer(palette="Dark2")

# Change group colors manually
# Read more: https://www.datanovia.com/learn/data-visualization/ggplot2/colors
p + scale_color_manual(values=c("#999999", "#E69F00"))+
  scale_fill_manual(values=c("#999999", "#E69F00"))

# 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")

# Control variable colors using their contributions
# use repel = TRUE to avoid overplotting
fviz_mca_var(res.mca, col.var = "contrib",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))

# 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)

## 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")

# 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)

# 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)

```

Description

Plots the classification, the uncertainty and the BIC values returned by the `Mclust()` function. Read more: [Model-Based Clustering in R \(Mclust\)](#).

Usage

```
fviz_mclust(
  object,
  what = c("classification", "uncertainty", "BIC"),
  ellipse.type = "norm",
  ellipse.level = 0.4,
  ggtheme = theme_classic(),
  ...
)

fviz_mclust_bic(
  object,
  model.names = NULL,
  shape = 19,
  color = "model",
  palette = NULL,
  legend = NULL,
  main = "Model selection",
  xlab = "Number of components",
  ylab = "BIC",
  ...
)
```

Arguments

<code>object</code>	an object of class <code>Mclust</code>
<code>what</code>	choose from one of the following three options: "classification" (default), "uncertainty" and "BIC".
<code>ellipse.type</code>	Character specifying frame type. Possible values are 'convex', 'confidence' or types supported by stat_ellipse including one of <code>c("t", "norm", "euclid")</code> .
<code>ellipse.level</code>	the size of the concentration ellipse in normal probability. Passed for <code>ggplot2::stat_ellipse</code> 's level. Ignored in 'convex'. Default value is 0.95.
<code>ggtheme</code>	function, ggplot2 theme name. The default is set by each function's <code>ggtheme</code> argument; see the function usage for the actual default. Set <code>ggtheme = NULL</code> to skip applying a ggpubr theme, so the plot keeps ggplot2 default theme or the theme set globally via <code>theme_set()</code> . Allowed values include ggplot2 official themes: <code>theme_gray()</code> , <code>theme_bw()</code> , <code>theme_minimal()</code> , <code>theme_classic()</code> , <code>theme_void()</code> , ...
<code>...</code>	other arguments to be passed to the functions fviz_cluster and ggpar .
<code>model.names</code>	one or more model names corresponding to models fit in object. The default is to plot the BIC for all of the models fit.

shape	point shape. To change point shape by model names use shape = "model".
color	point and line color.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
legend	character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). To remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
main	plot main title.
xlab	character vector specifying x axis labels. Use xlab = FALSE to hide xlab.
ylab	character vector specifying y axis labels. Use ylab = FALSE to hide ylab.

Value

A ggplot2 object.

Functions

- `fviz_mclust()`: Plots classification and uncertainty.
- `fviz_mclust_bic()`: Plots the BIC values.

See Also

[fviz_cluster](#). Online tutorial: [Model-Based Clustering in R \(Mclust\)](#).

Examples

```
if(requireNamespace("mclust", quietly = TRUE)){

# Compute model-based-clustering
library("mclust")
data("diabetes")
mc <- Mclust(diabetes[, -1])

# Visualize BIC values
fviz_mclust_bic(mc)

# Visualize classification
fviz_mclust(mc, "classification", geom = "point")
}
```

Description

Multiple factor analysis (MFA) is used to analyze a data set in which individuals are described by several sets of variables (quantitative and/or qualitative) structured into groups. `fviz_mfa()` provides ggplot2-based elegant visualization of MFA outputs from the R function: `MFA` [FactoMineR].

- `fviz_mfa_ind()`: Graph of individuals
- `fviz_mfa_var()`: Graph of variables
- `fviz_mfa_axes()`: Graph of partial axes
- `fviz_mfa()`: An alias of `fviz_mfa_ind(res.mfa, partial = "all")`
- `fviz_mfa_quali_biplot()`: Biplot of individuals and qualitative variables

Read more: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables](#).

Usage

```
fviz_mfa_ind(  
  X,  
  axes = c(1, 2),  
  geom = c("point", "text"),  
  repel = FALSE,  
  habillage = "none",  
  palette = NULL,  
  addEllipses = FALSE,  
  col.ind = "blue",  
  col.ind.sup = "darkblue",  
  alpha.ind = 1,  
  shape.ind = 19,  
  col.quali.var.sup = "black",  
  select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),  
  partial = NULL,  
  col.partial = "group",  
  ...  
)  
  
fviz_mfa_quali_biplot(  
  X,  
  axes = c(1, 2),  
  geom = c("point", "text"),  
  repel = FALSE,  
  title = "Biplot of individuals and qualitative variables - MFA",
```

```

    ...
  )

  fviz_mfa_var(
    X,
    choice = c("quanti.var", "group", "quali.var", "quali.sup"),
    axes = c(1, 2),
    geom = c("point", "text"),
    repel = FALSE,
    habillage = "none",
    col.var = "red",
    alpha.var = 1,
    shape.var = 17,
    col.var.sup = "darkgreen",
    palette = NULL,
    select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
    ...
  )

  fviz_mfa_axes(
    X,
    axes = c(1, 2),
    geom = c("arrow", "text"),
    col.axes = NULL,
    alpha.axes = 1,
    col.circle = "grey70",
    select.axes = list(name = NULL, contrib = NULL),
    repel = FALSE,
    ...
  )

  fviz_mfa(X, partial = "all", ...)

```

Arguments

<code>X</code>	an object of class MFA [FactoMineR].
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	a text specifying the geometry to be used for the graph. Allowed values are the combination of <code>c("point", "arrow", "text")</code> . Use <code>"point"</code> (to show only points); <code>"text"</code> to show only labels; <code>c("point", "text")</code> or <code>c("arrow", "text")</code> to show arrows and texts. Using <code>c("arrow", "text")</code> is sensible only for the graph of variables.
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>habillage</code>	an optional factor variable for coloring the observations by groups. Default value is <code>"none"</code> . If <code>X</code> is an MFA object from FactoMineR package, <code>habillage</code> can also specify the index of the factor variable in the data.

<code>palette</code>	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
<code>addEllipses</code>	logical value. If TRUE, draws ellipses around the individuals when <code>habillage != "none"</code> .
<code>col.ind, col.var, col.axes</code>	color for individuals, variables and <code>col.axes</code> respectively. Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>col.ind.sup</code>	color for supplementary individuals
<code>alpha.ind, alpha.var, alpha.axes</code>	controls the transparency of individual, variable, group and axes colors, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>shape.ind, shape.var</code>	point shapes of individuals, variables, groups and axes
<code>col.quali.var.sup</code>	color for supplementary qualitative variables. Default is "black".
<code>select.ind, select.var, select.axes</code>	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 contributions are drawn • union: logical. When several of name/cos2/contrib are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-cos2 ones.
<code>partial</code>	list of the individuals for which the partial points should be drawn. (by default, partial = NULL and no partial points are drawn). Use partial = "all" to visualize partial points for all individuals.

<code>col.partial</code>	color for partial individuals. By default, points are colored according to the groups.
<code>...</code>	Arguments to be passed to the function <code>fviz()</code>
<code>title</code>	the title of the graph
<code>choice</code>	the graph to plot. Allowed values include one of <code>c("quanti.var", "quali.var", "quali.sup", "group")</code> for plotting quantitative variables, qualitative variables, supplementary qualitative variables and group of variables, respectively.
<code>col.var.sup</code>	color for supplementary variables.
<code>col.circle</code>	a color for the correlation circle. Used only when X is a PCA output.

Value

a `ggplot2` plot

Author(s)

Fabian Mundt <f.mundt@inventionate.de>

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

[get_mfa](#). Online tutorial: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables](#).

Examples

```
# Compute 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)

# Eigenvalues/variances of dimensions
fviz_screplot(res.mfa)
# Group of variables
fviz_mfa_var(res.mfa, "group")
# Quantitative variables
fviz_mfa_var(res.mfa, "quanti.var", palette = "jco",
             col.var.sup = "violet", repel = TRUE)
# Graph of individuals colored by cos2
fviz_mfa_ind(res.mfa, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
# Partial individuals
fviz_mfa_ind(res.mfa, partial = rownames(wine)[1:3], col.partial = "black")
```

```

# Partial axes
fviz_mfa_axes(res.mfa)

# 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 of qualitative variables
fviz_mfa_var(res.mfa, "quali.var")
# Plot of supplementary qualitative variables
fviz_mfa_var(res.mfa, "quali.sup")

# 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)

```

fviz_nbclust

Determining 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()`: Determines and visualizes the optimal number of clusters using different methods: **within-cluster sum of squares**, **average silhouette**, and the **gap statistic**. Silhouette values are evaluated only for $k = 2, \dots, k_{\max}$ because average silhouette width is undefined for a one-cluster partition. The "silhouette" and "gap_stat" plots mark their optimum with a dashed guide line; set `mark_optimal = TRUE` to also mark the elbow on the "wss" plot, or `mark_optimal = FALSE` to omit the guide line for every method.
- `fviz_gap_stat()`: Visualizes the gap statistic generated by the function `clusGap()` [in cluster package]. The optimal number of clusters is specified using the `maxSE` method with `method = "firstSEmax"`.

For `method = "wss"`, `factoextra` computes the $k = 1$ baseline internally so helper functions such as `hcut()` and `hkmeans()` can keep rejecting direct $k = 1$ inputs.

Read more: [Determining the Optimal Number of Clusters in R](#).

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,
  ...,
  mark_optimal = NULL
)

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

Arguments

<code>x</code>	numeric matrix or data frame. In the function <code>fviz_nbclust()</code> , <code>x</code> can be the results of the function <code>NbClust()</code> . For <code>method = "silhouette"</code> or <code>"wss"</code> , <code>x</code> may also be a precomputed dissimilarity (an object of class <code>"dist"</code>); it is then passed directly to <code>FUNcluster</code> , which must be a dissimilarity-capable method (e.g. <code>cluster::pam</code> or <code>factoextra::hcut</code>). <code>method = "gap_stat"</code> still needs the raw data.
<code>FUNcluster</code>	a partitioning function which accepts as first argument a (data) matrix like <code>x</code> , second argument, say <code>k >= 2</code> , the number of clusters desired, and returns a list with a component named <code>cluster</code> which contains the grouping of observations. Allowed values include: <code>kmeans</code> , <code>cluster::pam</code> , <code>cluster::clara</code> , <code>cluster::fanny</code> , <code>hcut</code> , etc. In <code>method = "wss"</code> mode, <code>fviz_nbclust()</code> computes the <code>k = 1</code> baseline internally instead of calling <code>FUNcluster(x, 1, ...)</code> . This argument is not required when <code>x</code> is an output of the function <code>NbClust::NbClust()</code> .
<code>method</code>	the method to be used for estimating the optimal number of clusters. Possible values are <code>"silhouette"</code> (for average silhouette width), <code>"wss"</code> (for total within-cluster sum of squares), and <code>"gap_stat"</code> (for the gap statistic).
<code>diss</code>	<code>dist</code> object as produced by <code>dist()</code> , i.e.: <code>diss = dist(x, method = "euclidean")</code> . Used to compute the average silhouette width and within-cluster sum of squares. If <code>NULL</code> , <code>dist(x)</code> is computed with the default <code>method = "euclidean"</code>
<code>k.max</code>	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 the gap statistic.
verbose	logical value. If TRUE, progress information 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 is printed in <code>fviz_nbclust()</code> .
...	optionally further arguments: arguments for <code>FUNcluster()</code> in "wss"/"silhouette" modes; arguments for <code>clusGap()</code> in "gap_stat" mode. A <code>maxSE</code> list can also be supplied in "gap_stat" mode and is forwarded to <code>fviz_gap_stat()</code> .
mark_optimal	logical, or NULL (default). NULL keeps each method's standard marker: a dashed guide line at the estimated optimal number of clusters is drawn for method = "silhouette" (maximum average silhouette width) and method = "gap_stat" (the <code>maxSE</code> location), and omitted for method = "wss". Set TRUE to also mark the "wss" elbow (a maximum-distance heuristic that returns a candidate even when the data has no clear cluster structure; see Details), or FALSE to omit the guide line for every method.
gap_stat	an object of class "clusGap" returned by the function <code>clusGap()</code> [in cluster package]
maxSE	a list containing the parameters <code>method</code> and <code>SE.factor</code> used by <code>maxSE</code> to locate the gap statistic optimum. The default is <code>list(method = "firstSEmax", SE.factor = 1)</code> . Allowed methods include: • "globalmax": simply corresponds to the global maximum, i.e., <code>which.max(gap)</code> • "firstmax": gives the location of the first local maximum • "Tibs2001SEmax": uses the criterion, Tibshirani et al (2001) proposed: "the smallest k such that $\text{gap}(k) \geq \text{gap}(k+1) - s(k+1)$". It's also possible to use "the smallest k such that $\text{gap}(k) \geq \text{gap}(k+1) - \text{SE.factor} * s(k+1)$" where <code>SE.factor</code> is a numeric value which can be 1 (default), 2, 3, etc. • "firstSEmax": location of the first <code>f()</code> value which is not larger than the first local maximum minus <code>SE.factor * SE.f</code>, i.e, within an "f S.E." range of that maximum. • see <code>maxSE</code> for more options

Details

When `mark_optimal = TRUE`, the "wss" elbow is located with a deterministic chord-distance heuristic: both axes are rescaled to the unit interval $[0, 1]$ (so the choice does not depend on the magnitude of the within-cluster sum of squares) and the marked k is the one whose point lies farthest from the straight line joining the first and last points of the curve. This is related to the chord-distance idea used in knee detection, but it does not implement the full Kneedle sensitivity and local-maximum procedure of Satopää et al. (2011). The heuristic always returns a candidate, even for data with no clear cluster structure, which is why it is opt-in; the "silhouette" and "gap_stat" guide lines instead mark defined optima (the maximum average silhouette width and the `maxSE` location).

Value

Both `fviz_nbclust()` and `fviz_gap_stat()` return a `ggplot2` object.

Method selection for gap statistic

The default "firstSEmax" method returns the first value within SE.factor standard errors of the first local maximum. Other [maxSE](#) rules can be selected explicitly through maxSE.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

Satopää, V., Albrecht, J., Irwin, D. & Raghavan, B. (2011). Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. *2011 31st International Conference on Distributed Computing Systems Workshops*, 166-171. doi:[10.1109/ICDCSW.2011.20](#)

See Also

[fviz_cluster](#), [eclust](#). Online tutorial: [Determining the Optimal Number of Clusters in R](#).

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)

# Let factoextra mark the elbow automatically
fviz_nbclust(iris.scaled, kmeans, method = "wss", mark_optimal = TRUE)

# WSS with hierarchical clustering keeps the internal k = 1 baseline
fviz_nbclust(iris.scaled, hcut, method = "wss", hc_method = "complete")

# 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 = "firstSEmax")
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`], iii) `dudi.pca` [in `ade4`] and `epPCA` [ExPosition]. Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

- `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()`

Note that, `fviz_pca_xxx()` functions are wrapper around the core function `fviz()`, which is also a wrapper around the function `ggscatter()` [in `ggpubr`]. Therefore, further arguments, to be passed to the function `fviz()` and `ggscatter()`, can be specified in `fviz_pca_ind()` and `fviz_pca_var()`.

Usage

```
fviz_pca(X, ...)

fviz_pca_ind(
  X,
  axes = c(1, 2),
  geom = c("point", "text"),
  geom.ind = geom,
  repel = FALSE,
  habillage = "none",
```

```

    palette = NULL,
    addEllipses = FALSE,
    col.ind = "black",
    fill.ind = "white",
    col.ind.sup = "blue",
    alpha.ind = 1,
    shape.ind = NULL,
    select.ind = list(name = NULL, cos2 = NULL, contrib = NULL),
    max.points = NULL,
    sample.seed = 123,
    ...
)

fviz_pca_var(
  X,
  axes = c(1, 2),
  geom = c("arrow", "text"),
  geom.var = geom,
  repel = FALSE,
  col.var = "black",
  fill.var = "white",
  alpha.var = 1,
  col.quanti.sup = "blue",
  col.circle = "grey70",
  select.var = list(name = NULL, cos2 = NULL, contrib = NULL),
  ...
)

fviz_pca_biplot(
  X,
  axes = c(1, 2),
  geom = c("point", "text"),
  geom.ind = geom,
  geom.var = c("arrow", "text"),
  col.ind = "black",
  fill.ind = "white",
  col.var = "steelblue",
  fill.var = "white",
  gradient.cols = NULL,
  label = "all",
  invisible = "none",
  repel = FALSE,
  habillage = "none",
  palette = NULL,
  addEllipses = FALSE,
  shape.ind = NULL,
  title = "PCA - Biplot",
  biplot.type = c("auto", "form", "covariance"),

```

```
    ...
  )
```

Arguments

<code>X</code>	an object of class PCA [FactoMineR]; <code>prcomp</code> and <code>princomp</code> [stats]; <code>dudi</code> and <code>pca</code> [ade4]; <code>expoOutput/epPCA</code> [ExPosition].
<code>...</code>	Additional arguments. • in <code>fviz_pca_ind()</code> and <code>fviz_pca_var()</code>: Additional arguments are passed to the functions <code>fviz()</code> and <code>ggpubr::ggpar()</code>. • in <code>fviz_pca_biplot()</code> and <code>fviz_pca()</code>: Additional arguments are passed to <code>fviz_pca_ind()</code> and <code>fviz_pca_var()</code>.
<code>axes</code>	a numeric vector of length 2 specifying the dimensions to be plotted.
<code>geom</code>	a text specifying the geometry to be used for the graph. Allowed values are the combination of <code>c("point", "arrow", "text")</code> . Use "point" (to show only points); "text" to show only labels; <code>c("point", "text")</code> or <code>c("arrow", "text")</code> to show arrows and texts. Using <code>c("arrow", "text")</code> is sensible only for the graph of variables.
<code>geom.ind, geom.var</code>	as <code>geom</code> but for individuals and variables, respectively. Default is <code>geom.ind = c("point", "text")</code> , <code>geom.var = c("arrow", "text")</code> .
<code>repel</code>	logical; whether to use <code>ggrepel</code> to avoid overplotting text labels. The old <code>jitter</code> argument is kept for backward compatibility and is converted to <code>repel = TRUE</code> with a deprecation warning.
<code>habillage</code>	an optional factor variable for coloring the observations by groups. Default value is "none". If <code>X</code> is a PCA object from FactoMineR package, <code>habillage</code> can also specify the supplementary qualitative variable (by its index or name) to be used for coloring individuals by groups (see ?PCA in FactoMineR).
<code>palette</code>	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette .
<code>addEllipses</code>	logical value. If TRUE, draws ellipses around the individuals when <code>habillage != "none"</code> .
<code>col.ind, col.var</code>	color for individuals and variables, respectively. Can be a continuous variable or a factor variable. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>fill.ind, fill.var</code>	same as <code>col.ind</code> and <code>col.var</code> but for the fill color.

<code>col.ind.sup</code>	color for supplementary individuals
<code>alpha.ind, alpha.var</code>	controls the transparency of individual and variable colors, respectively. The value can vary from 0 (total transparency) to 1 (no transparency). Default value is 1. Possible values also include "cos2", "contrib", "coord", "x", and "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 <code>habillage = "none"</code> .
<code>shape.ind</code>	optional factor variable to map the point shape of individuals, independently of their color. This allows colouring individuals by one grouping variable (<code>col.ind/habillage</code>) and shaping them by another (e.g. <code>fviz_pca_ind(res, col.ind = group1, shape.ind = group2)</code>). Default NULL keeps the previous behaviour (shape follows the colour grouping). Use <code>+ labs(shape = , colour =)</code> to rename the legends.
<code>select.ind, select.var</code>	a selection of individuals/variables to be drawn. Allowed values are NULL or a list containing the arguments <code>name</code> , <code>cos2</code> or <code>contrib</code> : • <code>name</code>: is a character vector containing individuals/variables to be drawn • <code>cos2</code>: if <code>cos2</code> is in $[0, 1]$, ex: 0.6, then individuals/variables with a <code>cos2</code> > 0.6 are drawn. if <code>cos2</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>cos2</code> are drawn. • <code>contrib</code>: if <code>contrib</code> > 1, ex: 5, then the top 5 individuals/variables with the highest <code>contrib</code> are drawn • <code>union</code>: logical. When several of <code>name</code>/<code>cos2</code>/<code>contrib</code> are given, FALSE (default) combines them with AND (each condition further narrows the selection); TRUE combines them with OR (an element is kept if it matches any condition), e.g. named items <i>plus</i> the top-<code>cos2</code> ones.
<code>max.points</code>	integer or NULL. When the individual / row / column cloud has more than <code>max.points</code> points, a random subset of that many <i>points</i> is drawn so the plot stays readable (usable labels instead of an over-plotted cloud). Only the drawn points/labels are thinned: any ellipse (<code>addEllipses</code>) and the group mean point are still computed on the full data, so a convex / confidence frame is not shrunk or inflated by the draw. When points are coloured/split by a group (e.g. <code>habillage</code>), the draw is stratified so every group keeps at least a minimum number of points and none is decimated. When colour, fill or size is mapped to a continuous metric (e.g. <code>col.ind = "cos2"</code>), its scale and legend are pinned to the full-data range, so a point's colour, fill and size do not depend on how many points are drawn. A message reports how many points are shown. The subset is reproducible (see <code>sample.seed</code>) and does not change the caller's random stream. NULL (default) draws every point.
<code>sample.seed</code>	the random seed used to pick the <code>max.points</code> subset, for a reproducible figure. Ignored when <code>max.points</code> is NULL.
<code>col.quant.sup</code>	a color for the quantitative supplementary variables.
<code>col.circle</code>	a color for the correlation circle. Used only when X is a PCA output.
<code>gradient.cols</code>	vector of colors to use for n-colour gradient. Allowed values include brewer and ggsci color palettes.

label	a text specifying the elements to be labelled. Default value is "all". Allowed values are "all", "none", or a 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 "all", "none", or a combination of c("ind", "ind.sup", "quali", "var", "quanti.sup").
title	the title of the graph
biplot.type	type of biplot scaling for fviz_pca_biplot(). Options are: • "auto" (default): Uses range-based rescaling for visualization • "form": Gabriel form biplot scaling (equivalent to stats::biplot(..., scale = 0)), preserving the score geometry used to compare individuals. • "covariance": Gabriel covariance biplot scaling (equivalent to stats::biplot(..., scale = 1)), preserving the variable covariance geometry. Because both clouds share a single set of axes (unlike stats::biplot()'s dual axes), the individuals can appear compressed relative to the variable arrows; use "form" or "auto" if the individual cloud is the focus.

Note: "form" and "covariance" scaling requires prcomp or princomp objects.

Value

a ggplot

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[fviz_ca](#), [fviz_mca](#). Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

Examples

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

# Graph of individuals
# ++++++

# Default plot
# Use repel = TRUE to avoid overplotting (slow if many points)
fviz_pca_ind(res.pca, col.ind = "#00AFBB",
  repel = TRUE)
```

```

# 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 using ggtheme: https://www.datanovia.com/learn/data-visualization/ggplot2/themes

fviz_pca_ind(res.pca, col.ind="cos2", geom = "point",
  gradient.cols = c("white", "#2E9FDF", "#FC4E07" ))

# Color individuals by groups, add concentration ellipses
# Change group colors using RColorBrewer color palettes
# Read more: https://www.datanovia.com/learn/data-visualization/ggplot2/colors
# Remove labels: label = "none".
fviz_pca_ind(res.pca, label="none", habillage=iris$Species,
  addEllipses=TRUE, ellipse.level=0.95, palette = "Dark2")

# Change group colors manually
# Read more: https://www.datanovia.com/learn/data-visualization/ggplot2/colors
fviz_pca_ind(res.pca, label="none", habillage=iris$Species,
  addEllipses=TRUE, ellipse.level=0.95,
  palette = c("#999999", "#E69F00", "#56B4E9"))

# 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))

# By default several conditions combine with AND (intersection).
# Use union = TRUE for OR: keep named individuals PLUS the top-contrib ones.
fviz_pca_ind(res.pca,
  select.ind = list(name = c("23", "42"), contrib = 20, union = TRUE))

# Graph of variables
# ++++++

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

# Control variable colors using their contributions
fviz_pca_var(res.pca, col.var = "contrib",
  gradient.cols = c("white", "blue", "red"),
  ggtheme = theme_minimal())

# Biplot of individuals and variables
# ++++++

```

```
# 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,
               ggtheme = theme_minimal())

# Biplot scaling modes:
# Form biplot - focus on individual distances
fviz_pca_biplot(res.pca, biplot.type = "form",
               label = "var", habillage = iris$Species)
# Covariance biplot - focus on variable correlations
fviz_pca_biplot(res.pca, biplot.type = "covariance",
               label = "var", habillage = iris$Species)
```

fviz_silhouette

Visualize Silhouette Information from Clustering

Description

Silhouette (Si) analysis compares an observation's average dissimilarity within its assigned cluster with its average dissimilarity to the nearest alternative cluster. `fviz_silhouette()` provides a ggplot2-based visualization of silhouette information from i) the result of `silhouette()`, `pam()`, `clara()` and `fanny()` [in cluster package]; ii) `eclust()` and `hcut()` [in factoextra]. Results without silhouette information, such as one-cluster `eclust/hcut` objects, are rejected with a package-level error.

Read more: [Cluster Validation Statistics in R](#).

Usage

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

Arguments

<code>sil.obj</code>	an object of class <code>silhouette</code> , <code>pam</code> , <code>clara</code> , or <code>fanny</code> [cluster], or <code>eclust</code> or <code>hcut</code> [factoextra]. For <code>eclust</code> and <code>hcut</code> , silhouette information must be available, which requires at least two clusters.
<code>label</code>	logical value. If true, x axis tick labels are shown
<code>print.summary</code>	logical value. If TRUE, a summary of cluster silhouettes is printed by <code>fviz_silhouette()</code> .
<code>...</code>	other arguments to be passed to the function <code>ggpubr::ggpar()</code> .

Details

- Observations with a large silhouette S_i (almost 1) are very well clustered.
- A small S_i (around 0) means that the observation lies between two clusters.
- A negative S_i means that the observation is, on average, closer to another cluster than to its assigned cluster.
- Silhouette plots require at least two clusters and available silhouette widths.

Value

a ggplot2 object.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[fviz_cluster](#), [hcut](#), [hkmeans](#), [eclust](#), [fviz_dend](#). Online tutorial: [Cluster Validation Statistics in R](#).

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 = 2)

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

# Visualize silhouette information
requireNamespace("cluster", quietly = TRUE)
sil <- cluster::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]
## Not run:
# PAM clustering
# ++++++
```

```

requireNamespace("cluster", quietly = TRUE)
pam.res <- cluster::pam(iris.scaled, 3)
# Visualize pam clustering
fviz_cluster(pam.res, ellipse.type = "norm")+
  theme_minimal()
# Visualize silhouette 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 silhouette information
if (hc.cut$nbclust > 1) fviz_silhouette(hc.cut)

## End(Not run)

```

fviz_umap

Visualize a UMAP or t-SNE embedding

Description

fviz_umap() and fviz_tsne() draw a ggplot2 scatter plot of a two-dimensional **embedding** produced by UMAP or t-SNE, with factoextra's grouping, ellipse and palette styling. They accept the result of uwot::umap() / umap::umap() / Rtsne::Rtsne(), or a plain matrix / data frame of coordinates.

Unlike PCA, an embedding has **no eigenvalues**: the axes explain no fixed percentage of variance, so they are labelled UMAP1/UMAP2 (tSNE1/tSNE2) without a percentage, and there is deliberately no scree plot, no variable loadings, and no correlation circle - those are meaningless for an embedding. For methods that *do* have eigenvalues (PCA, CA, MCA, MDS) see [fviz_pca](#) and [as_factoextra_pca](#).

Read more: [UMAP in R: Nonlinear Dimension Reduction & Visualization](#) and [t-SNE in R: Visualize High-Dimensional Data](#).

Usage

```

fviz_umap(
  X,
  dims = c(1L, 2L),
  habillage = NULL,
  col.ind = NULL,
  geom = "point",
  label = FALSE,
  pointsize = 1.5,
  labelsize = 4,
  addEllipses = FALSE,

```

```

    ellipse.type = "convex",
    ellipse.level = 0.95,
    mean.point = FALSE,
    palette = NULL,
    gradient.cols = NULL,
    density = FALSE,
    facet.by = NULL,
    repel = TRUE,
    max.points = NULL,
    sample.seed = 123,
    legend.title = NULL,
    caption = NULL,
    ggtheme = theme_minimal(),
    ...
)

fviz_tsne(
  X,
  dims = c(1L, 2L),
  habillage = NULL,
  col.ind = NULL,
  geom = "point",
  label = FALSE,
  pointsize = 1.5,
  labelsize = 4,
  addEllipses = FALSE,
  ellipse.type = "convex",
  ellipse.level = 0.95,
  mean.point = FALSE,
  palette = NULL,
  gradient.cols = NULL,
  density = FALSE,
  facet.by = NULL,
  repel = TRUE,
  max.points = NULL,
  sample.seed = 123,
  legend.title = NULL,
  caption = NULL,
  ggtheme = theme_minimal(),
  ...
)

```

Arguments

<code>X</code>	an embedding: a <code>uwot::umap()</code> result (a matrix, or a list with <code>\$embedding</code>), an <code>Rtsne::Rtsne()</code> result (<code>\$Y</code>), a <code>umap::umap()</code> result (<code>\$layout</code>), or a numeric matrix / data frame of coordinates (one column per dimension).
<code>dims</code>	two distinct positive integer indices specifying which embedding dimensions to

	plot.
habillage	an optional factor / vector used to colour the points by group (discrete). Its length must equal the number of embedded observations.
col.ind	point colour: a factor / character vector (discrete groups) or a numeric vector to colour by a continuous feature value (as in a Seurat feature plot). A single colour name is also allowed. The eigenvalue metrics accepted by fviz_pca_ind ("cos2", "contrib", ...) are rejected here - an embedding has no such quantities.
geom	character: "point" and/or "text".
label	logical; if TRUE, label the points by row name.
pointsize, labelsize	point and label size.
addEllipses	logical; if TRUE, draw an outline around each group. Default FALSE. When TRUE the default ellipse.type is "convex" (a hull) rather than a normal/t confidence ellipse: a confidence ellipse assumes a metric that UMAP/t-SNE do not preserve.
ellipse.type, ellipse.level	ellipse type and confidence level. See fviz_cluster .
mean.point	logical; if TRUE, show group mean points. Default FALSE - a centroid in embedding space is not a feature-space mean.
palette, gradient.cols	discrete group palette / continuous colour gradient, passed to ggpar .
density	logical; if TRUE, overlay 2D density contour lines (a visual aid, not a probability readout).
facet.by	optional variable (length = number of observations) to facet by.
repel	logical; if TRUE, use ggrepel for labels.
max.points, sample.seed	passed to the downsampling used for large embeddings; see fviz_pca_ind .
legend.title	colour legend title. Defaults to "Groups" for a discrete colouring and "value" for a continuous feature.
caption	optional plot caption (e.g. an interpretation reminder). Off by default.
ggtheme, ...	passed to ggscatter / ggplot2.

Details

UMAP and t-SNE produce a low-dimensional embedding, not a variance decomposition. Local relationships are often more informative than global geometry, but both remain method- and parameter-dependent. In particular, the **size** of a cluster, the **distance** between well-separated clusters, and the amount of empty space between them are not quantitatively meaningful and change with perplexity or n_neighbors (Wattenberg et al., 2016). Similar embedding scatter plots are drawn by `Seurat::DimPlot()` / `FeaturePlot()` and `ggpca`.

Value

a ggplot2 object.

References

- McInnes, L., Healy, J. and Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426*.
- Wattenberg, M., Viégas, F. and Johnson, I. (2016). How to Use t-SNE Effectively. *Distill*. doi:10.23915/distill.00002.

See Also

[fviz_pca](#), [as_factoextra_pca](#), [fviz_cluster](#). Online tutorials: [UMAP in R: Nonlinear Dimension Reduction & Visualization](#) and [t-SNE in R: Visualize High-Dimensional Data](#).

Examples

```
if (requireNamespace("uwot", quietly = TRUE)) {
  set.seed(123)
  um <- uwot::umap(iris[, 1:4], n_neighbors = 15)
  fviz_umap(um, habillage = iris$Species, addEllipses = TRUE)
  # colour by a continuous feature value
  fviz_umap(um, col.ind = iris$Petal.Length)
}
```

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

Read more: [Correspondence Analysis \(CA\) in R: Contingency Tables, Biplot & Interpretation](#).

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

<https://www.datanovia.com/learn/>

See Also

[fviz_ca](#). Online tutorial: [Correspondence Analysis \(CA\) in R: Contingency Tables, Biplot & Interpretation](#).

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	<i>Assessing Clustering Tendency</i>
--------------------	--------------------------------------

Description

Before applying clustering methods, assess whether the data show non-random structure, a process known as **assessing clustering tendency**. `get_clust_tendency()` uses the Hopkins statistic and, optionally, an ordered dissimilarity image (ODI). Rows containing missing values are omitted, and the remaining matrix must be numeric and finite. The observed rows are sampled without replacement; artificial points are sampled uniformly within the observed bounding box. Zero-range columns are ignored for the Hopkins calculation when at least one column varies; an all-constant data set remains undefined. In the ODI, objects grouped by hierarchical clustering are displayed in consecutive order. For more details and interpretation, see [Assessing Clustering Tendency in R](#).

Usage

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

Arguments

<code>data</code>	a numeric data frame or matrix. Columns are variables and rows are samples. Computations are performed on complete rows. To calculate the Hopkins statistic for variables, transpose the data first.
<code>n</code>	a positive integer specifying the number of points selected from sample space and from the observed data. Must be smaller than the number of complete observations.
<code>graph</code>	logical value; if TRUE the ordered dissimilarity image (ODI) is shown.
<code>gradient</code>	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.
<code>seed</code>	an integer seed for reproducibility, or NULL to use the current RNG stream. When non-NULL, the function restores the caller RNG state on exit.

Details

Hopkins statistic: Values near 1 support a clustered tendency, values near 0.5 are consistent with complete spatial randomness, and values near 0 indicate regular spacing. The statistic uses the formula from Cross and Jain (1982), with exponent $d = D$, where D is the number of varying columns. Redundant constant columns therefore do not change the statistic. A Beta(n, n) comparison is an approximation under ideal complete-spatial-randomness assumptions, not an unconditional finite-sample distribution for every data set.

Note on interpretation: This function returns the Hopkins statistic H where values close to 1 indicate clusterable data. Some other packages (e.g., `performance::check_clusterstructure`) return $1-H$, where values close to 0 indicate clusterability. Always check the documentation of the specific implementation you are using.

Breaking change: `factoextra` uses the corrected Hopkins statistic formula (Wright 2022). Results differ from legacy `factoextra` and a one-time warning is emitted. Set `options(factoextra.warn_hopkins = FALSE)` to silence the warning.

For large datasets, nearest-neighbor distances are computed with a low-memory fallback when the full pairwise matrix would exceed `getOption("factoextra.hopkins.max_matrix_cells", 2e7)` cells. Only the sampled row's own occurrence is excluded from its nearest-neighbor search; a duplicated row therefore remains a valid zero-distance neighbor.

VAT (Visual Assessment of Cluster Tendency): The VAT displays clustering tendency by showing square-shaped dark (or colored) blocks along the diagonal in a VAT image.

Value

A list containing:

<code>hopkins_stat</code>	The Hopkins statistic.
<code>plot</code>	The ordered dissimilarity image generated by <code>fviz_dist()</code> , or NULL when <code>graph = FALSE</code> .

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

`fviz_dist`. Online tutorial: [Assessing Clustering Tendency in R](#).

Examples

```
data(iris)

# Silence the one-time compatibility warning in examples
old_hopkins_warn <- getOption("factoextra.warn_hopkins")
options(factoextra.warn_hopkins = FALSE)

# 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)
options(factoextra.warn_hopkins = old_hopkins_warn)
```

get_famd

Extract the results for individuals and variables - FAMD

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals and variables from Factor Analysis of Mixed Data (FAMD) outputs.

- `get_famd()`: Extract the results for variables and individuals
- `get_famd_ind()`: Extract the results for individuals only
- `get_famd_var()`: Extract the results for quantitative and qualitative variables only

Read more: [Factor Analysis of Mixed Data \(FAMD\) in R: Compute, Visualize & Interpret.](#)

Usage

```
get_famd(
  res.famd,
  element = c("ind", "var", "quanti.var", "quali.var", "quali.sup")
)

get_famd_ind(res.famd)

get_famd_var(
  res.famd,
  element = c("var", "quanti.var", "quali.var", "quali.sup")
)
```

Arguments

<code>res.famd</code>	an object of class FAMD [FactoMineR].
<code>element</code>	the element to subset from the output. Possible values are "ind", "var", "quanti.var", "quali.var" or "quali.sup".

Value

a list of matrices containing the results for the active individuals and variables, including:

coord	coordinates of individuals/variables.
cos2	cos2 values representing the quality of representation on the factor map.
contrib	contributions of individuals / variables to the principal components.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[fviz_famd](#). Online tutorial: [Factor Analysis of Mixed Data \(FAMD\) in R: Compute, Visualize & Interpret](#).

Examples

```
# Compute FAMD
library("FactoMineR")
data(wine)
res.famd <- FAMD(wine[,c(1,2, 16, 22, 29, 28, 30,31)], graph = FALSE)
res.famd.sup <- FAMD(wine[,c(1,2, 16, 22, 29, 28, 30,31)],
                    sup.var = 2, graph = FALSE)

# Extract the results for qualitative variable categories
quali.var <- get_famd_var(res.famd, "quali.var")
print(quali.var)
head(quali.var$coord) # coordinates of qualitative variables

# Extract the results for supplementary qualitative variable categories
quali.sup <- get_famd_var(res.famd.sup, "quali.sup")
print(quali.sup)
head(quali.sup$coord) # coordinates of supplementary qualitative variables

# Extract the results for quantitative variables
quanti.var <- get_famd_var(res.famd, "quanti.var")
print(quanti.var)
head(quanti.var$coord) # coordinates

# Extract the results for individuals
ind <- get_famd_ind(res.famd)
print(ind)
head(ind$coord) # coordinates of individuals
```

get_hmfa	<i>Extract the results for individuals/variables/group/partial axes - HMFA</i>
----------	--

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals/quantitative variables/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()`: Extract the results for quantitative variables, qualitative variables, and groups
- `get_hmfa_partial()`: Extract the results for partial.node.

Read more: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables.](#)

Usage

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

get_hmfa_ind(res.hmfa)

get_hmfa_var(res.hmfa, element = c("quanti.var", "quali.var", "group"))

get_hmfa_partial(res.hmfa)
```

Arguments

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

Value

a list of matrices containing the results for the active individuals, variables, groups, and partial nodes, including:

<code>coord</code>	coordinates
<code>cos2</code>	cos2
<code>contrib</code>	contributions

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Fabian Mundt <f.mundt@inventionate.de>

See Also

[fviz_hmfa](#), [fviz_mfa](#). Online tutorial: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables](#).

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_var(res.hmfa, "quali.var")
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

For FactoMineR MCA results, `get_mca()` and `get_mca_var()` also support `element = "quanti.sup"` for quantitative supplementary variables and report a clean package-level error when that result is absent.

Read more: [Multiple Correspondence Analysis \(MCA\) in R: Compute, Visualize & Interpret](#).

Usage

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

get_mca_var(res.mca, element = c("var", "mca.cor", "quanti.sup"))

get_mca_ind(res.mca)
```

Arguments

<code>res.mca</code>	an object of class MCA [FactoMineR], acm [ade4], expoOutput/epMCA [Ex-Position].
<code>element</code>	the element to subset from the output. Possible values are "var" for variables, "ind" for individuals, "mca.cor" for correlation between variables and principal dimensions, and "quanti.sup" for quantitative supplementary variables in FactoMineR MCA results.

Value

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

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

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

[fviz_mca](#). Online tutorial: [Multiple Correspondence Analysis \(MCA\) in R: Compute, Visualize & Interpret](#).

Examples

```
# Multiple Correspondence Analysis
# ++++++
# Install and load FactoMineR to compute MCA
# install.packages("FactoMineR")
library("FactoMineR")
data(poison)
res.mca <- MCA(poison, quanti.sup = 1:2, 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
quanti.sup <- get_mca(res.mca, "quanti.sup")
head(quanti.sup$coord) # coordinates of quantitative supplementary variables
```

get_mfa

Extract the results for individuals/variables/group/partial axes - MFA

Description

Extract all the results (coordinates, squared cosine and contributions) for the active individuals/quantitative variables/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()`: Extract the results for quantitative variables, qualitative variables, and groups
- `get_mfa_partial_axes()`: Extract the results for partial axes only

Read more: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables.](#)

Usage

```

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

get_mfa_ind(res.mfa)

get_mfa_var(
  res.mfa,
  element = c("quanti.var", "quali.var", "quali.sup", "group")
)

get_mfa_partial_axes(res.mfa)

```

Arguments

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

Value

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

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

See Also

[fviz_mfa](#). Online tutorial: [Multiple Factor Analysis \(MFA\) in R: Analyze Groups of Variables](#).

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_var(res.mfa, "quali.var")
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_mfa(res.mfa, "quali.sup") # Results for supplementary qualitative variable categories
```

get_pca

Extract the results for individuals/variables - PCA

Description

Extract all the results (coordinates, squared cosines, and 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

Read more: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret.](#)

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 or princomp [stats]; factoextra_pca; pca, dudi, between, or within [ade4]; or expoOutput/epPCA [ExPosition].
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. For an adapted recipe/workflow object this is NULL when the metric cannot be recovered. For a rank-truncated prcomp object, individual cos2 is the quality within the retained component subspace because the discarded row inertia is not stored in the fitted object.
contrib	contributions of the individuals/variables; contributions to each nonzero-inertia axis sum to 100 percent, while a zero-inertia axis contains zeros
cor	loading-times-component-standard-deviation coordinates for PCA objects from stats; these equal variable-component correlations when the input variables were standardized. Returned for variables; for an adapted recipe/workflow object it is NULL when correlations cannot be recovered.

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

References

<https://www.datanovia.com/learn/>

See Also

[fviz_pca](#). Online tutorial: [Principal Component Analysis \(PCA\) in R: Compute, Visualize & Interpret](#).

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 cuts the tree into k clusters. It also accepts correlation-based distance measures such as "pearson", "spearman" and "kendall". Direct calls require $k \geq 2$; helper-level one-cluster handling is implemented in callers such as eclust() and fviz_nbclust().

Read more: [Hierarchical Clustering in R: Dendrograms, Agglomerative & Divisive](#).

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	a single integer specifying the number of clusters to be generated. Must be at least 2 and smaller than the number of observations.
isdiss	logical value specifying whether x is already a dissimilarity matrix. If TRUE, x must inherit from class "dist" and contain only finite values.
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. If scaling produces NA values, hcut() stops with a package-level error.
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 assignment of observations after cutting the tree
- nbclust: the number of clusters
- silinfo: the silhouette information of observations (available when $k > 1$)
- size: the size of clusters
- data: a matrix containing the original or the standardized data (if stand = TRUE)

See Also

[fviz_dend](#), [hkmeans](#), [eclust](#). Online tutorial: [Hierarchical Clustering in R: Dendrograms, Agglomerative & Divisive](#).

Examples

```
data(USArrests)

# Compute hierarchical clustering and cut into 4 clusters
```

```

res <- hcut(USArrests, k = 4, stand = TRUE)

# Cluster assignments 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. Read more: [Hierarchical K-Means Clustering in R: A Hybrid for Stable Clusters](#).

- `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

<code>x</code>	a numeric matrix, data frame or vector
<code>k</code>	a single integer specifying the number of clusters to be generated. Must be at least 2 and smaller than <code>nrow(x)</code> .
<code>hc.metric</code>	the distance measure to be used. Possible values are "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski" (see <code>?dist</code>).
<code>hc.method</code>	the agglomeration method to be used. Possible values include "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid" (see <code>?hclust</code>).
<code>iter.max</code>	the maximum number of iterations allowed for k-means.
<code>km.algorithm</code>	the algorithm to be used for kmeans (see <code>?kmeans</code>).
<code>...</code>	others arguments to be passed to the function <code>plot.hclust()</code> ; (see <code>?plot.hclust</code>)
<code>hkmeans</code>	an object of class <code>hkmeans</code> (returned by the function <code>hkmeans()</code>)
<code>rect.col</code>	Vector with border colors for the rectangles around clusters in dendrogram

Details

The procedure is as follows:

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. Optimize the clustering.

This means that the final optimized partitioning obtained at step 4 might be different from the initial partitioning obtained at step 2. Consider mainly the result displayed by `fviz_cluster()`.

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()`

See Also

[eclust](#), [hcut](#), [fviz_dend](#). Online tutorial: [Hierarchical K-Means Clustering in R: A Hybrid for Stable Clusters](#).

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, ellipse.type = "norm", ellipse.level = 0.68)

```

housetasks	<i>House tasks contingency table</i>
------------	--------------------------------------

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()
```

map_factominer_legacy_names

Map legacy FactoMineR category names to current labels

Description

Map legacy FactoMineR category names to current labels

Usage

```
map_factominer_legacy_names(
  X,
  names,
  element = c("quali.var", "quali.sup", "var"),
  quiet = FALSE
)
```

Arguments

X	a FactoMineR object (MCA, MFA, FAMD, HMFA).
names	character vector of category labels.
element	element to map. Use "var" for MCA categories or "quali.var" for MFA/FAMD/HMFA qualitative categories. "quali.sup" maps supplementary qualitative categories when available.
quiet	if TRUE, suppress warnings.

Value

Character vector of mapped labels.

Examples

```
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  data(poison)
  res.mca <- FactoMineR::MCA(poison, quanti.sup = 1:2, quali.sup = 3:4, graph = FALSE)
  map <- factominer_category_map(res.mca, element = "var")
  map_factominer_legacy_names(res.mca, map$legacy_underscore[1:3], element = "var", quiet = TRUE)
}
```

multishapes	<i>A dataset containing clusters of multiple shapes</i>
-------------	---

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 observation.

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	<i>Poison</i>
--------	---------------

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	<i>Print method for an object of class factoextra</i>
------------------	---

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>

Examples

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

theme_factoextra	<i>A clean publication theme for factoextra plots</i>
------------------	---

Description

A ggplot2 theme tuned for the dimension-reduction scatter plots and biplots produced by the `fviz_*`() functions: a light, unobtrusive coordinate grid (so points can be read against the axes), a clean background, and de-emphasized axis lines. Pass it through the `ggtheme` argument, e.g. `fviz_pca_ind(res, ggtheme = theme_factoextra())`, or add it to any returned plot with `+ theme_factoextra()`.

Read more: [ggplot2 Themes in R: Customize the Look](#).

Usage

```
theme_factoextra(base_size = 12, base_family = "")
```

Arguments

<code>base_size</code>	base font size, in points.
<code>base_family</code>	base font family.

Details

The `fviz_*` functions have different default themes (`fviz_pca_*` use `theme_minimal()`, `fviz_cluster()` uses `theme_grey()`, `fviz_dend()` uses `theme_classic()`, ...). For one consistent look across several plots in the same document, pass `ggtheme = theme_factoextra()` to each call (see the last example) rather than relying on `ggplot2::theme_set()`, which the `fviz_*` functions override with their explicit `ggtheme` default.

Value

A ggplot2 theme object.

See Also

[factoextra_palette](#). Online tutorial: [ggplot2 Themes in R: Customize the Look](#).

Examples

```
data(iris)
res.pca <- prcomp(iris[, 1:4], scale. = TRUE)
fviz_pca_ind(res.pca, habillage = iris$Species,
             palette = factoextra_palette("okabe"),
             ggtheme = theme_factoextra())

# or add it to an existing plot
fviz_pca_var(res.pca) + theme_factoextra()

# a consistent look across the family: pass the same ggtheme to each call
```

```
km <- kmeans(scale(iris[, 1:4]), 3)
fviz_pca_ind(res.pca, ggtheme = theme_factoextra())
fviz_cluster(km, data = iris[, 1:4], ggtheme = theme_factoextra())
```

Index

as_factoextra_pca, 3, 80, 83

clara, 78

clusGap, 68, 70

coord.ellipse, 23, 47

coord_fixed, 23, 47

decathlon2, 6

deprecated, 7

dist, 9, 9, 10

eclust, 10, 35, 71, 78, 79, 97, 99

eigenvalue, 13

entanglement, 45

facto_summarize, 18

factoextra_palette, 16, 104

factominer_category_map, 18

fanny, 78

fviz, 21, 72

fviz_add, 26

fviz_ca, 15, 27, 59, 76, 84

fviz_ca_biplot (fviz_ca), 27

fviz_ca_col (fviz_ca), 27

fviz_ca_row (fviz_ca), 27

fviz_cluster, 12, 32, 43, 62, 63, 71, 79, 82, 83

fviz_contrib, 3, 36, 41

fviz_cos2, 3, 38, 39

fviz_dend, 12, 35, 42, 79, 97, 99

fviz_dist, 86

fviz_dist (dist), 9

fviz_eig, 3

fviz_eig (eigenvalue), 13

fviz_ellipses, 46

fviz_famd, 48, 88

fviz_famd_ind (fviz_famd), 48

fviz_famd_var (fviz_famd), 48

fviz_gap_stat (fviz_nbclust), 68

fviz_hmfa, 15, 51, 59, 90

fviz_hmfa_group (deprecated), 7

fviz_hmfa_ind (fviz_hmfa), 51

fviz_hmfa_ind_starplot (deprecated), 7

fviz_hmfa_quali_biplot (fviz_hmfa), 51

fviz_hmfa_quali_var (deprecated), 7

fviz_hmfa_quanti_var (deprecated), 7

fviz_hmfa_var (fviz_hmfa), 51

fviz_mca, 15, 31, 55, 76, 91

fviz_mca_biplot (fviz_mca), 55

fviz_mca_ind (fviz_mca), 55

fviz_mca_var (fviz_mca), 55

fviz_mclust, 61

fviz_mclust_bic (fviz_mclust), 61

fviz_mfa, 15, 54, 59, 64, 90, 93

fviz_mfa_axes (fviz_mfa), 64

fviz_mfa_group (deprecated), 7

fviz_mfa_ind (fviz_mfa), 64

fviz_mfa_ind_starplot (deprecated), 7

fviz_mfa_quali_biplot (fviz_mfa), 64

fviz_mfa_quali_var (deprecated), 7

fviz_mfa_quanti_var (deprecated), 7

fviz_mfa_var (fviz_mfa), 64

fviz_nbclust, 68

fviz_pca, 3, 15, 31, 59, 72, 80, 83, 95

fviz_pca_biplot (fviz_pca), 72

fviz_pca_contrib (fviz_contrib), 36

fviz_pca_ind, 72, 82

fviz_pca_ind (fviz_pca), 72

fviz_pca_var, 72

fviz_pca_var (fviz_pca), 72

fviz_sceeplot (eigenvalue), 13

fviz_silhouette, 12, 35, 43, 78

fviz_tsne, 5

fviz_tsne (fviz_umap), 80

fviz_umap, 5, 80

get_ca, 31, 83

get_ca_col (get_ca), 83

get_ca_row (get_ca), 83

get_clust_tendency, 85

`get_dist`(`dist`), 9
`get_eig`(`eigenvalue`), 13
`get_eigenvalue`(`eigenvalue`), 13
`get_famd`, 50, 87
`get_famd_ind`(`get_famd`), 87
`get_famd_var`(`get_famd`), 87
`get_hmfa`, 54, 89
`get_hmfa_group`(`deprecated`), 7
`get_hmfa_ind`(`get_hmfa`), 89
`get_hmfa_partial`(`get_hmfa`), 89
`get_hmfa_quali_var`(`deprecated`), 7
`get_hmfa_quanti_var`(`deprecated`), 7
`get_hmfa_var`(`get_hmfa`), 89
`get_mca`, 59, 90
`get_mca_ind`(`get_mca`), 90
`get_mca_var`(`get_mca`), 90
`get_mfa`, 67, 92
`get_mfa_group`(`deprecated`), 7
`get_mfa_ind`(`get_mfa`), 92
`get_mfa_partial_axes`(`get_mfa`), 92
`get_mfa_quali_var`(`deprecated`), 7
`get_mfa_quanti_var`(`deprecated`), 7
`get_mfa_var`(`get_mfa`), 92
`get_pca`, 38, 41, 94
`get_pca_ind`(`get_pca`), 94
`get_pca_var`(`get_pca`), 94
`ggpar`, 15, 34, 37, 62, 82
`ggscatter`, 34, 72, 82

`hcut`, 35, 78, 79, 96, 99
`hkmeans`, 35, 79, 97, 98
`hkmeans_tree`(`hkmeans`), 98
`housetasks`, 100

`layout.auto`, 44
`layout.gem`, 44
`layout.mds`, 44
`layout_as_tree`, 44
`layout_nicely`, 44
`layout_with_drl`, 44
`layout_with_gem`, 44
`layout_with_lgl`, 44
`layout_with_mds`, 44

`map_factominer_legacy_names`, 101
`maxSE`, 11, 68, 70, 71
`MCA`, 58
`multishapes`, 102

`palette`, 47, 49, 57, 63, 66, 74

`pam`, 78
`poison`, 102
`print.factoextra`, 103
`print.hkmeans`(`hkmeans`), 98

`silhouette`, 78
`stat_ellipse`, 23, 34, 47, 62

`tanglegram`, 45
`theme_factoextra`, 17, 104