Package: exploreit (via r-universe)

October 30, 2026

Type Package

Version 1.0.3

Title 'SciViews::R' - Exploratory Data Analysis

Description Multivariate analysis and data exploration for the 'SciViews::R' dialect.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0)

Imports ape (>= 5.7.1), broom (>= 1.0.4), ca (>= 0.71.1), chart (>= 1.5.0), data.table (>= 1.15.4), factoextra (>= 1.0.7), FactoMineR (>= 2.8), fastcluster (>= 1.2.3), generics (>= 0.1.3), ggdendro (>= 0.1.23), ggfortify (>= 0.4.16), ggplot2 (>= 3.4.2), ggrepel (>= 0.9.3), graphics (>= 4.2.0), grDevices (>= 4.2.0), grid (>= 4.2.0), lattice (>= 0.21.8), MASS (>= 7.3.58.3), rlang (>= 1.1.1), SciViews (>= 1.6.0), stats (>= 4.2.0), svFlow (>= 1.2.0), tibble (>= 3.2.1), utils (>= 4.2.0), vegan (>= 2.6.4),

Suggests cluster (>= 2.1.4), collapse (>= 2.0.12), data.io (>= 1.5.0), dplyr (>= 1.1.4), dtplyr (>= 1.3.1), forcats (>= 1.0.0), fs (>= 1.6.1), svBase (>= 1.4.0), svFast (>= 0.1.0), svTidy (>= 0.1.0), svMisc (>= 1.4.0), helpai (>= 0.1.0), knitr (>= 1.42), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

Remotes SciViews/data.io, SciViews/chart, SciViews/SciViews, SciViews/svBase, SciViews/svFlow, SciViews/svMisc, SciViews/svFast, SciViews/svTidy, SciViews/helpai

License MIT + file LICENSE

URL https://github.com/SciViews/exploreit, https://www.sciviews.org/exploreit/, https://sciviews.r-universe.dev/exploreit

BugReports https://github.com/SciViews/exploreit/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libxslt-dev libssl-dev libnode-dev zlib1g-dev

Repository https://sciviews.r-universe.dev

Date/Publication 2025-08-29 05:06:07 UTC

RemoteUrl https://github.com/SciViews/exploreit

RemoteRef HEAD

RemoteSha 1511066403e9cd9178783d4500e3517549506302

Contents

exploreit-package 2
as.dissimilarity 3
ca 4
circle 6
cluster 7
dissimilarity 10
geom_dendroline 12
k_means 13
mds 16
mfa 18
pca 21
scale 23

Index **25**

`exploreit-package` *'SciViews::R' - Exploratory Data Analysis*

Description

Multivariate analysis and data exploration for 'SciViews::R'. PCA, CA, MFA, K-Means clustering, hierarchical clustering, MDS...

Important functions

- `pca()` for Principal Component Analysis (PCA)
- `ca()` for Correspondence Analysis (CA)
- `mfa()` for Multiple Factor Analysis (MFA)
- `k_means()` for K-Means clustering
- dissimilarity for computing dissimilarity (distance) matrices
- `cluster()` for Hierarchical clustering
- `mds()` for metric and Non-metric MultiDimensional Scaling (MDS, NMDS)

Author(s)

Maintainer: Philippe Grosjean <phgrosjean@sciviews.org> (ORCID)

See Also

Useful links:

- https://github.com/SciViews/exploreit
- https://www.sciviews.org/exploreit/
- https://sciviews.r-universe.dev/exploreit
- Report bugs at https://github.com/SciViews/exploreit/issues

---

as.dissimilarity *Convert a dist or matrix object into a Dissimilarity object*

Description

Create a `Dissimilarity` matrix from an existing distance matrix as `dist` object (e.g., from `stats::dist()`, or `vegan::vegdist()`), or from a similarly shaped `matrix`object.

Usage

```
as.dissimilarity(x, ...)

as_dissimilarity(x, ...)

## S3 method for class 'matrix'
as.dissimilarity(x, ...)

## S3 method for class 'dist'
as.dissimilarity(x, ...)

## S3 method for class 'Dissimilarity'
as.dissimilarity(x, ...)
```

Arguments

x	An object to coerce into a Dissimilarity object.
...	Further argument passed to the coercion method.

Value

A Dissimilarity object.

See Also

dissimilarity()

Examples

```
SciViews::R
iris <- read("iris", package = "datasets")
iris_num <- iris[, -5] # Only numeric columns from iris
# Construct a dist object
iris_dist <- dist(iris_num)
class(iris_dist)
# Convert it into a Dissimilarity object
iris_dis <- as.dissimilarity(iris_dist)
class(iris_dis)
```

---

ca *Correspondence Analysis (CA)*

Description

ca() is a reexport of the function from the {ca} package, it offers a ca(formula, data, ...) interface. It is supplemented here with various chart() types.

Usage

```
ca(obj, ...)

## S3 method for class 'ca'
autoplot(
  object,
  choices = 1L:2L,
  type = c("screeplot", "altscreeplot", "biplot"),
  col = "black",
  fill = "gray",
  aspect.ratio = 1,
  repel = FALSE,
  ...
)
```

```
## S3 method for class 'ca'
chart(
  data,
  choices = 1L:2L,
  ...,
  type = c("screeplot", "altscreeplot", "biplot"),
  env = parent.frame()
)
```

Arguments

obj	A formula or a data frame with numeric columns, or a matrix, or a table or xtabs two-way contingency table, see ca::ca(). The formula version allows to specify two categorical variables from a data frame as ~f1 + f2. The other versions analyze a two-way contingency table crossing two factors.
...	Further arguments from ca::ca() or for plot.
object	A **pcomp** object
choices	Vector of two positive integers. The two axes to plot, by
type	The type of plot to produce: "screeplot" or "altscreeplot" for two versions of the screeplot, or "biplot" for the CA biplot.
col	The color for the points representing the observations, black by default.
fill	The color to fill bars, gray by default
aspect.ratio	height/width of the plot, 1 by default (for plots where the ratio height / width does matter)
repel	Logical. Should repel be used to rearrange points labels? FALSEby default
data	Idem
env	The environment where to evaluate code, parent.frame() by default, which should not be changed unless you really know what you are doing!

Value

pca() produces a **ca** object.

Examples

```
library(chart)
data(caith, package = "MASS")
caith # A two-way contingency table
class(caith) # in a data frame
caith_ca <- ca(caith)
summary(caith_ca)

chart$scree(caith_ca)
chart$altscree(caith_ca)

chart$biplot(caith_ca)
```

`circle` *Draw circles in a plot*

Description

Add a circle in a base R plot, given its center (x and y coordinates) and its diameter. In {exploreit}, this function can be used to cut a circular dendrogram, see example.

Usage

```
circle(x = 0, y = 0, d = 1, col = 0, lwd = 1, lty = 1, ...)
```

Arguments

`x`	The x coordinate of the center of the circle.
`y`	The y coordinate of the center of the circle.
`d`	The diameter of the circle.
`col`	The color of the border of the circle.
`lwd`	The width of the circle border.
`lty`	The line type to use to draw the circle.
`...`	More arguments passed to `symbols()`.

Value

This function returns `NULL`. It is invoked for it side effect of adding a circle in a base R plot.

See Also

`symbols()`

Examples

```
plot(x = 0:2, y = 0:2)
circle(x = 1, y = 1, d = 1, col = "red", lwd = 2, lty = 2)
```

`cluster` *Hierarchical Clustering Analysis*

Description

Hierarchical clustering is an agglomerative method that uses a dissimilarity matrix to group individuals. It is represented by a dendrogram that can be cut at a certain level to form the final clusters.

Usage

```
cluster(x, ...)

## Default S3 method:
cluster(x, ...)

## S3 method for class 'dist'
cluster(x, method = "complete", fun = NULL, ...)

## S3 method for class 'Cluster'
str(object, max.level = NA, digits.d = 3L, ...)

## S3 method for class 'Cluster'
labels(object, ...)

## S3 method for class 'Cluster'
nobs(object, ...)

## S3 method for class 'Cluster'
predict(object, k = NULL, h = NULL, ...)

## S3 method for class 'Cluster'
augment(x, data, k = NULL, h = NULL, ...)

## S3 method for class 'Cluster'
plot(
  x,
  y,
  labels = TRUE,
  hang = -1,
  check = TRUE,
  type = "vertical",
  lab = "Height",
  ...
)

## S3 method for class 'Cluster'
autoplot(
```

```
  object,
  labels = TRUE,
  type = "vertical",
  circ.text.size = 3,
  theme = theme_sciviews(),
  xlab = "",
  ylab = "Height",
  ...
)

## S3 method for class 'Cluster'
chart(data, ..., type = NULL, env = parent.frame())
```

Arguments

x	A Dissimilarity object.
...	Further arguments for the methods (see their respective manpages).
method	The agglomeration method used. "complete" by default. Other options depend on the function fun = used. For the default one, you can also use "single", "average", "mcquitty", "ward.D", "ward.D2", "centroid", or "median".
fun	The function to use to do the calculation. By default, it is fastcluster::hclust(), an fast and memory-optimized version of the default R function stats::hclust(). You can also use flashClust::hclust(), cluster::agnes(), cluster::diana(), as well as, any other function that returns an hclustobject, or something convertible to hclust with as.hclust(). The default (NULL) means that the **fastcluster** implementation is used.
object	A cluster object.
max.level	The maximum level to present.
digits.d	The number of digits to print.
k	The number of clusters to get.
h	The height where the dendrogram should be cut (give either k = or h = , but not both at the same time).
data	The original dataset
y	Do not use it.
labels	Should we show the labels (TRUE by default).
hang	The fraction of the plot height at which labels should hang below (by default, -1 meaning labels are all placed at the extreme of the plot).
check	The validity of the cluster object is verified first to avoid crashing R. You can put it at FALSE to speed up computation if you are really sure your object is valid.
type	The type of dendrogram, by default, "vertical". It could also be "horizontal" (more readable when there are many observations), or "circular" (even more readable with many observations, but more difficult to chose the cutting level).
lab	The label of the y axis (vertical) or x axis (horizontal), by default "Height".

circ.text.size	Size of the text for a circular dendrogram
theme	The ggplot2 theme to use, by default, it is theme_sciviews().
xlab	Label of the x axis (nothing by default)
ylab	Label of the y axis, by default "Height".
env	The environment where to evaluate formulas. If you don't understand this, it means you should not touch it!

Value

A Cluster object inheriting from hclust. Specific methods are: str() (compact display of the object content), labels() (get the labels for the observations), nobs() (number of observations), predict() (get the clusters, given a cutting level), augment() (add the groups to the original data frame or tibble), plot() (create a dendrogram as base R plot), autoplot() (create a dendrogram as a ggplot2), and chart() (create a dendrogram as a chart variant of a ggplot2).

See Also

dissimilarity(), stats::hclust(), fastcluster::hclust()

Examples

```
SciViews::R
iris <- read("iris", package = "datasets")
iris_num <- iris[, -5] # Only numeric columns from iris
# Cluster the 150 flowers
iris_dis <- dissimilarity(iris_num, method = "euclidean", scale = TRUE)
(iris_clust <- cluster(iris_dis, method = "complete"))
str(iris_clust) # More useful
str(iris_clust, max.level = 3L) # Only the top of the dendrogram

# Dendrogram with base R graphics
plot(iris_clust)
plot(iris_clust, labels = FALSE, hang = 0.1)
abline(h = 3.5, col = "red")
# Horizontal dendrogram
plot(iris_clust, type = "horizontal", labels = FALSE)
abline(v = 3.5, col = "red")
# Circular dendrogram
plot(iris_clust, type = "circular", labels = FALSE)
circle(d = 3.5, col = "red")

# Chart version of the dendrogram
chart(iris_clust) +
  geom_dendroline(h = 3.5, color = "red")
# Horizontal dendrogram and without labels
chart$horizontal(iris_clust, labels = FALSE) +
  geom_dendroline(h = 3.5, color = "red")
# Circular dendrogram with labels
chart$circ(iris_clust, circ.text.size = 3) + # Abbreviate type and change size
  geom_dendroline(h = 3.5, color = "red")
```

```
# Get the clusters
predict(iris_clust, h = 3.5)
# Four clusters
predict(iris_clust, k = 4)
# Add the clusters to the data (.fitted column added)
augment(data = iris, iris_clust, k = 4)
```

`dissimilarity` *Calculate a dissimilarity matrix*

Description

Compute a distance matrix from all pairs of columns or rows in a data frame, using a unified SciViews::R formula interface.

Usage

```
dissimilarity(
  data,
  formula = ~.,
  subset = NULL,
  method = "euclidean",
  scale = FALSE,
  rownames.col = getOption("SciViews.dtx.rownames", default = ".rownames"),
  transpose = FALSE,
  fun = NULL,
  ...
)

## S3 method for class 'Dissimilarity'
print(x, digits.d = 3L, rownames.lab = "labels", ...)

## S3 method for class 'Dissimilarity'
labels(object, ...)

## S3 method for class 'Dissimilarity'
nobs(object, ...)

## S3 method for class 'Dissimilarity'
autoplot(
  object,
  order = TRUE,
  show.labels = TRUE,
  lab.size = NULL,
  gradient = list(low = "blue", mid = "white", high = "red"),
  ...
```

```
)

## S3 method for class 'Dissimilarity'
chart(
  data,
  order = TRUE,
  show.labels = TRUE,
  lab.size = NULL,
  gradient = list(low = "blue", mid = "white", high = "red"),
  ...,
  type = NULL,
  env = parent.frame()
)
```

Arguments

data	A data.frame, tibble or matrix.
formula	A right-side only formula (~ . . .) indicating which columns to keep in the data. The default one (~ .) keeps all columns.
subset	An expression indicating which rows to keep from data.
method	The distance (dissimilarity) method to use. By default, it is "euclidean", but it can also be "maximum", "binary", "minkowski" from stats::dist(), or "bray", "manhattan", "canberra", "clark", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horm", "mountfort", "raup", "binomial", "chao", "cao", "mahalanobis", "chisq", or "chord" from vegan::vegdist(), or any other distance from the function you provide in fun =.
scale	Do we scale (mean = 0, standard deviation = 1) the data before calculating the distance (FALSE by default)?
rownames.col	In case the data object does not have row names (a tibble for instance), which column should be used for name of the rows?
transpose	Do we transpose data first (to calculate distance between columns instead of rows)? By default, not (FALSE).
fun	A function that does the calculation and return a dist-like object (similar to what stats::dist()) provides. If NULL (by default), stats::dist() or vegan::vegdist() is used, depending on method =. Note that both functions calculate "canberra" differently, and in this case, it is the vegan::vegdist() version that is used by default. Other compatible functions: vegan::designdist(), cluster::daisy(), factoextra::get_dist(), and probably more.
...	Further parameters passed to the fun = (see its man page) for dissimilarity(), or further arguments passed to methods.
x, object	A Dissimilarity object
digits.d	Number of digits to print, by default, 3.
rownames.lab	The name of the column containing the labels, by default "labels".
order	Do we reorder the lines and columns according to their resemblance (TRUE by default)?

show.labels	Are the labels displayed on the axes (TRUE by default)?
lab.size	Force the size of the labels (NULL by default for automatic size).
gradient	The palette of color to use in the plot.
type	The type of plot. For the moment, only one plot is possible and the default value (NULL) should not be changed.
env	The environment where to evaluate the formula. If you don't understand this, you probably don't have to touch this arguments.

Value

An S3 object of class c("Dissimilarity", "dist"), thus inheriting from dist. A Dissimilarity object is better displayed (specific print() method), and has also dedicated methods labels() (get line and column labels), nobs() (get number of observations, that is, number of lines or columns), autoplot() (generate a ggplot2 from the matrix) and chart() (generate a chart version of the ggplot2).

See Also

stats::dist(), vegan::vegdist(), vegan::designdist(), cluster::daisy(), factoextra::get_dist()

Examples

```
SciViews::R
iris <- read("iris", package = "datasets")
iris_num <- iris[, -5] # Only numeric columns from iris
# Compare the 150 flowers and nicely print the result
dissimilarity(iris_num, method = "manhattan")
# Compare the measurements by transposing and scaling them first
iris_dist <- dissimilarity(iris_num, method = "euclidean",
  scale = TRUE, transpose = TRUE)
iris_dist
class(iris_dist)
labels(iris_dist)
nobs(iris_dist)
# specific plots
autoplot(iris_dist)
chart(iris_dist, gradient = list(low = "green", mid = "white", high = "red"))
```

---

geom_dendroline *Draw a line to cut a dendrogram*

Description

Add a line (horizontal, vertical, or circular, depending on the dendrogram type) at height h to depict where it is cut into groups.

Usage

```
geom_dendroline(h, ...)
```

Arguments

h	The height to cut the dendrogram.
...	Further arguments passed to geom_hline() (this is really a convenience function that builds on it).

See Also

geom_hline()

Examples

```
SciViews::R
iris <- read("iris", package = "datasets")
iris_num <- iris[, -5]
iris_num %>.%
  dissimilarity(.) %>.%
  cluster(.) ->
  iris_cluster
chart(iris_cluster) +
  geom_dendroline(h = 3, color = "red")
```

---

k_means *K-means clustering*

Description

Perform a k-means clustering analysis using the stats::kmeans() function in {stats} but creating a **k_means** object that possibly embeds the original data with the analysis for a richer set of methods.

Usage

```
k_means(
  x,
  k,
  centers = k,
  iter.max = 10L,
  nstart = 1L,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  trace = FALSE,
  keep.data = TRUE
)

profile_k(x, fun = kmeans, method = "wss", k.max = NULL, ...)
```

```
## S3 method for class 'kmeans'
augment(x, data, ...)

## S3 method for class 'k_means'
predict(object, ...)

## S3 method for class 'k_means'
plot(
  x,
  y,
  data = x$data,
  choices = 1L:2L,
  col = NULL,
  c.shape = 8,
  c.size = 3,
  ...
)

## S3 method for class 'k_means'
autoplot(
  object,
  data = object$data,
  choices = 1L:2L,
  alpha = 1,
  c.shape = 8,
  c.size = 3,
  theme = NULL,
  use.chart = FALSE,
  ...
)

## S3 method for class 'k_means'
chart(data, ..., type = NULL, env = parent.frame())
```

Arguments

x	A data frame or a matrix with numeric data
k	The number of clusters to create, or a set of initial cluster centers. If a number, a random set of initial centers are computed first.
centers	Idem (centers is synonym to k)
iter.max	Maximum number of iterations (10 by default)
nstart	If k is a number, how many random sets should be chosen?
algorithm	The algorithm to use. May be abbreviated. See stats::kmeans() for more details about available algorithms.
trace	Logical or integer. Should process be traced. Higher value produces more tracing information.

keep.data	Do we keep the data in the object? If TRUE (by default), a richer set of methods could be applied to the resulting object, but it takes more space in memory. Use FALSE if you want to save RAM.
fun	The kmeans clustering function to use, kmeans() by default.
method	The method used in profile_k(): "wss" (by default, total within sum of square), "silhouette" (average silhouette width) or "gap_stat" (gap statistics).
k.max	Maximum number of clusters to consider (at least two). If not provided, a reasonable default is calculated.
...	Other arguments transmitted to factoextra::fviz_nbclust().
data	The original data frame
object	The *k_means** object
y	Not used
choices	The axes (variables) to plot (first and second by default)
col	Color to use
c.shape	The shape to represent cluster centers
c.size	The size of the shape representing cluster centers
alpha	Semi-transparency to apply to points
theme	The ggplot theme to apply to the plot
use.chart	If TRUE use chart(), otherwise, use ggplot().
type	Not used here
env	Not used here

Value

k_means() creates an object of classes **k_means** and **kmeans**. profile_k() is used for its side-effect of creating a plot that should help to chose the best value for k.

Examples

```
data(iris, package = "datasets")
iris_num <- iris[, -5] # Only numerical variables
library(chart)

# Profile k is to be taken only asx a (useful) indication!
profile_k(iris_num) # 2, maybe 3 clusters
iris_k2 <- k_means(iris_num, k = 2)
chart(iris_k2)

iris_k3 <- k_means(iris_num, k = 3, nstart = 20L) # Several random starts
chart(iris_k3)

# Get clusters and compare with Species
iris3 <- augment(iris_k3, iris) # Use predict() to just get clusters
head(iris3)
table(iris3$.cluster, iris3$Species) # setosa OK, the other are mixed a bit
```

mds *Multidimensional scaling or principal coordinates analysis*

Description

Perform a PCoA (‘type = "metric"). or other forms of MDS.

Usage

```
mds(
  dist,
  k = 2,
 type = c("metric", "nonmetric", "cmdscale", "wcmdscale", "sammon", "isoMDS", "monoMDS",
    "metaMDS"),
  p = 2,
  ...
)

## S3 method for class 'mds'
plot(x, y, ...)

## S3 method for class 'mds'
autoplot(object, labels, col, ...)

## S3 method for class 'mds'
chart(data, labels, col, ..., type = NULL, env = parent.frame())

shepard(dist, mds, p = 2)

## S3 method for class 'shepard'
plot(
  x,
  y,
  l.col = "red",
  l.lwd = 1,
  xlab = "Observed Dissimilarity",
  ylab = "Ordination Distance",
  ...
)

## S3 method for class 'shepard'
autoplot(
  object,
  alpha = 0.5,
  l.col = "red",
  l.lwd = 1,
  xlab = "Observed Dissimilarity",
```

```
  ylab = "Ordination Distance",
  ...
)

## S3 method for class 'shepard'
chart(
  data,
  alpha = 0.5,
  l.col = "red",
  l.lwd = 1,
  xlab = "Observed Dissimilarity",
  ylab = "Ordination Distance",
  ...,
  type = NULL,
  env = parent.frame()
)

## S3 method for class 'mds'
augment(x, data, ...)

## S3 method for class 'mds'
glance(x, ...)
```

Arguments

dist	A **dist** object from stats::dist() or other compatible functions like vegan::vegdist(), or a **Dissimilarity** object, see dissimilarity().
k	The dimensions of the space for the representation, usually k = 2 (by default). It should be possible to use also k = 3 with extra care and custom plots.
type	Not used
p	For types "nonmetric", "metaMDS", "isoMDS", "monoMDS" and "sammon", a Shepard plot is also precalculated. pis the power for Minkowski distance in the configuration scale. By default, p = 2. Leave it like that if you don't understand what it means see MASS::Shepard().
...	More arguments (see respective types or functions)
x	Idem
y	Not used
object	An **mds** object
labels	Points labels on the plot (optional)
col	Points color (optional)
data	A data frame to augment with columns from the MDS analysis
env	Not used
mds	Idem
l.col	Color of the line in the Shepard's plot (red by default)
l.lwd	Width of the line in the Shepard"s plot (1 by default)

xlab	Label for the X axis (a default value exists)
ylab	Idem for the Y axis
alpha	Alpha transparency for points (0.5 by default, meaning 50% transparency)

Value

A **mds** object, which is a list containing all components from the corresponding function, plus possibly Shepard if the Shepard plot is precalculated.

Examples

```
library(chart)
data(iris, package = "datasets")
iris_num <- iris[, -5] # Only numeric columns
iris_dis <- dissimilarity(iris_num, method = "euclidean")

# Metric MDS
iris_mds <- mds$metric(iris_dis)
chart(iris_mds, labels = 1:nrow(iris), col = iris$Species)

# Non-metric MDS
iris_nmds <- mds$nonmetric(iris_dis)
chart(iris_nmds, labels = 1:nrow(iris), col = iris$Species)
glance(iris_nmds) # Good R^2
iris_sh <- shepard(iris_dis, iris_nmds)
chart(iris_sh) # Excellent matching + linear -> metric MDS is OK here
```

---

mfa *Multiple Factor Analysis (MFA)*

Description

Analyze several groups of variables at once with supplementary groups of variables or individuals. Each group can be numeric, factor or contingency tables. Missing values are replaced by the column mean and missing values for factors are treated as an additional level. This is a formula interface to the FactoMineR::MFA() function.

Usage

```
mfa(data, formula, nd = 5, suprow = NA, ..., graph = FALSE)

## S3 method for class 'MFA'
autoplot(
  object,
  type = c("screeplot", "altscreeplot", "loadings", "scores", "groups", "axes",
    "contingency", "ellipses"),
  choices = 1L:2L,
  name = deparse(substitute(object)),
```

```
  col = "black",
  fill = "gray",
  title,
  ...,
  env
)

## S3 method for class 'MFA'
chart(
  data,
  choices = 1L:2L,
  name = deparse(substitute(data)),
  ...,
  type = NULL,
  env = parent.frame()
)
```

Arguments

`data`	A data frame
`formula`	A formula that specifies the variables groups to consider (see details)
`nd`	Number of dimensions kept in the results (by default, 5)
`suprow`	A vector indicating the row indices for the supplemental individuals
`...`	Additional arguments to `FactoMineR::MFA()` or to the plot
`graph`	If `TRUE`a graph is displayed (`FALSE` by default)
`object`	An **MFA** object
`type`	The type of plot to produce: `"screeplot"` or `"altscreeplot"` for two versions of the screeplot, `"loadings"`, `"scores"`, `"groups"`, `"axes"`, `"contingency"` or `"ellipses"` for the different views of the MFA.
`choices`	Vector of two positive integers. The two axes to plot, by default first and second axes.
`name`	The name of the object (automatically defined by default)
`col`	The color for the points representing the observations, black by default.
`fill`	The color to fill bars, gray by default
`title`	The title of the plot (optional, a reasonable default is used)
`env`	The environment where to evaluate code, `parent.frame()` by default, which should not be changed unless you really know what you are doing!

Details

The formula presents how the different columns of the data frame are grouped and indicates the kind of sub-table they are and the name we give to them in the analysis. So, a component of the formula for one group is `n * kind %as% name` where `n` is the number of columns belonging to this group, starting at column 1 for first group, `kind` is `std` for numeric variables to be standardized and used as a PCA, `num` for numerical variables to use as they are also as a PCA, `cnt` for counts

in a contingency table to be treated as a CA and `fct` for classical factors (categorical variables). Finally, `name` is a (short) name you use to identify this group. The kind may be omitted and it will be `std` by default. If `%as% name` is omitted, a generic name (group1, group2, group3, ...) is used. The complete formula is the addition of the different groups to include in the analysis and the subtraction of the supplementary groups not included in the analysis, like `~n1*std %as% gr1 - n2*fct %as% gr2 + n3*num %as% gr3`, with groups "gr1" and "gr3" included in the analysis and group "gr2" as supplemental. The total `n1 + n2 + n3` must equal the number of columns in the data frame.

Value

An **MFA** object

Note

The symbols for the groups are different in `mfa()` and `FactoMineR::MFA()`). To avoid further confusion, the symbols use three letters here:

- `std` is the same as `s` in `MFA()`: "standardized" and is the default
- `num` stands here for "numeric", thus continuous variables `c` in `MFA()`
- `cnt` stands for "contingency" table and matches `f` in `MFA()`
- `fct` stands for "factor", thus qualitative variables `n` in `MFA()`

Examples

```
# Same example as in {FactoMineR}
library(chart)
data(wine, package = "FactoMineR")
wine_mfa <- mfa(data = wine,
  ~ -2*fct %as% orig +5 %as% olf + 3 %as% vis + 10 %as% olfag + 9 %as% gust - 2 %as% ens)
wine_mfa
summary(wine_mfa)

chart$scree(wine_mfa)
chart$altscree(wine_mfa)

chart$loadings(wine_mfa)
chart$scores(wine_mfa)
chart$groups(wine_mfa)

chart$axes(wine_mfa)
# No contingency group! chart$contingency(wine_mfa)
chart$ellipses(wine_mfa)
```

pca *Principal Component Analysis (PCA)*

Description

Principal Component Analysis (PCA)

Usage

```
pca(x, ...)

## S3 method for class 'pcomp'
autoplot(
  object,
 type = c("screeplot", "altscreeplot", "loadings", "correlations", "scores", "biplot"),
  choices = 1L:2L,
  name = deparse(substitute(object)),
  ar.length = 0.1,
  circle.col = "gray",
  col = "black",
  fill = "gray",
  scale = 1,
  aspect.ratio = 1,
  repel = FALSE,
  labels,
  title,
  xlab,
  ylab,
  ...
)

## S3 method for class 'pcomp'
chart(
  data,
  choices = 1L:2L,
  name = deparse(substitute(data)),
  ...,
  type = NULL,
  env = parent.frame()
)

## S3 method for class 'princomp'
augment(x, data = NULL, newdata, ...)

## S3 method for class 'princomp'
tidy(x, matrix = "u", ...)
```

```
as.prcomp(x, ...)

## Default S3 method:
as.prcomp(x, ...)

## S3 method for class 'prcomp'
as.prcomp(x, ...)

## S3 method for class 'princomp'
as.prcomp(x, ...)
```

Arguments

x	A formula or a data frame with numeric columns, for as.prcomp(), an object to coerce into **prcomp**.
...	For pca(), further arguments passed to SciViews::pcomp(), notably, data= associated with a formula, subset=(optional), na.action=, method= that can be "svd" or "eigen". See SciViews::pcomp() for more details on these arguments.
object	A **pcomp** object
type	The type of plot to produce: "screeplot" or "altscreeplot" for two versions of the screeplot, "loadings", "correlations", or "scores" for the different views of the PCA, or a combined "biplot".
choices	Vector of two positive integers. The two axes to plot, by default first and second axes.
name	The name of the object (automatically defined by default)
ar.length	The length of the arrow head on the plot, 0.1 by default
circle.col	The color of the circle on the plot, gray by default
col	The color for the points representing the observations, black by default.
fill	The color to fill bars, gray by default
scale	The scale to apply for annotations, 1 by default
aspect.ratio	height/width of the plot, 1 by default (for plots where the ratio height / width does matter)
repel	Logical. Should repel be used to rearrange points labels? FALSEby default
labels	The label of the points (optional)
title	The title of the plot (optional, a reasonable default is used)
xlab	The label for the X axis. Automatically defined if not provided
ylab	Idem for the Y axis
data	The original data frame used for the PCA
env	The environment where to evaluate code, parent.frame() by default, which should not be changed unless you really know what you are doing!
newdata	A data frame with similar structure to data and new observations
matrix	Indicate which component should be be tidied. See broom::tidy.prcomp()

Value

pca() produces a **pcomp** object.

Examples

```
library(chart)
library(ggplot2)
data(iris, package = "datasets")
iris_num <- iris[, -5] # Only numeric columns
iris_pca <- pca(data = iris_num, ~ .)
summary(iris_pca)
chart$scree(iris_pca) # OK to keep 2 components
chart$altscree(iris_pca) # Different presentation

chart$loadings(iris_pca, choices = c(1L, 2L))
chart$scores(iris_pca, choices = c(1L, 2L), aspect.ratio = 3/5)
# or better:
chart$scores(iris_pca, choices = c(1L, 2L), labels = iris$Species,
  aspect.ratio = 3/5) +
  stat_ellipse()

# biplot
chart$biplot(iris_pca)
```

---

scale *Scale a data frame (data.frame, data.table or tibble's tbl_df)*

Description

Center or scale all variables in a data frame. This takes a data frame and return an object of the same class.

Usage

```
## S3 method for class 'data.frame'
scale(x, center = TRUE, scale = TRUE)

## S3 method for class 'tbl_df'
scale(x, center = TRUE, scale = TRUE)

## S3 method for class 'data.table'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	A data frame
center	Are the columns centered (mean = 0)?
scale	Are the column scaled (standard deviation = 1)?

Value

An object of the same class as x.

Examples

```
data(trees, package = "datasets")
colMeans(trees)
trees2 <- scale(trees)
head(trees2)
class(trees2)
colMeans(trees2)
```

Index

as.dissimilarity, 3
as.hclust(), *8*
as.prcomp *(*pca*)*, 21
as_dissimilarity *(*as.dissimilarity*)*, 3
augment(), *9*
augment.Cluster *(*cluster*)*, 7
augment.kmeans *(*k_means*)*, 13
augment.mds *(*mds*)*, 16
augment.princomp *(*pca*)*, 21
autoplot(), *9*
autoplot.ca *(*ca*)*, 4
autoplot.Cluster *(*cluster*)*, 7
autoplot.Dissimilarity *(*dissimilarity*)*, 10
autoplot.k_means *(*k_means*)*, 13
autoplot.mds *(*mds*)*, 16
autoplot.MFA *(*mfa*)*, 18
autoplot.pcomp *(*pca*)*, 21
autoplot.shepard *(*mds*)*, 16

broom::tidy.prcomp(), *22*

ca, 4
ca(), *3*
ca::ca(), *5*
chart(), *9*, *15*
chart.ca *(*ca*)*, 4
chart.Cluster *(*cluster*)*, 7
chart.Dissimilarity *(*dissimilarity*)*, 10
chart.k_means *(*k_means*)*, 13
chart.mds *(*mds*)*, 16
chart.MFA *(*mfa*)*, 18
chart.pcomp *(*pca*)*, 21
chart.shepard *(*mds*)*, 16
circle, 6
cluster, 7
cluster(), *3*
cluster::agnes(), *8*
cluster::daisy(), *11*, *12*
cluster::diana(), *8*

dissimilarity, *3*, 10
dissimilarity(), *4*, *9*, *11*, *17*

exploreit *(*exploreit-package*)*, 2
exploreit-package, 2

factoextra::fviz_nbclust(), *15*
factoextra::get_dist(), *11*, *12*
FactoMineR::MFA(), *18–20*
fastcluster::hclust(), *8*, *9*
flashClust::hclust(), *8*

geom_dendroline, 12
geom_hline(), *13*
ggplot(), *15*
glance.mds *(*mds*)*, 16

k_means, 13
k_means(), *3*, *15*

labels(), *9*
labels.Cluster *(*cluster*)*, 7
labels.Dissimilarity *(*dissimilarity*)*, 10

MASS::Shepard(), *17*
mds, 16
mds(), *3*
mfa, 18
mfa(), *3*, *20*

nobs(), *9*
nobs.Cluster *(*cluster*)*, 7
nobs.Dissimilarity *(*dissimilarity*)*, 10

pca, 21
pca(), *3*, *5*, *23*
plot(), *9*
plot.Cluster *(*cluster*)*, 7
plot.k_means *(*k_means*)*, 13
plot.mds *(*mds*)*, 16
plot.shepard *(*mds*)*, 16

predict(), *9*
predict.Cluster *(*cluster*)*, 7
predict.k_means *(*k_means*)*, 13
print.Dissimilarity *(*dissimilarity*)*, 10
profile_k *(*k_means*)*, 13
profile_k(), *15*

scale, 23
SciViews::pcomp(), *22*
shepard *(*mds*)*, 16
stats::dist(), *3*, *11*, *12*, *17*
stats::hclust(), *8*, *9*
stats::kmeans(), *13*, *14*
str(), *9*
str.Cluster *(*cluster*)*, 7
symbols(), *6*

theme_sciviews(), *9*
tidy.princomp *(*pca*)*, 21

vegan::designdist(), *11*, *12*
vegan::vegdist(), *3*, *11*, *12*, *17*