Package ‘ggfacto’

September 22, 2026

Title Graphs for Correspondence Analysis

Version 0.4.0

Description Readable, complete and pretty graphs for multiple correspondence analysis, correspondence analysis and principal component analysis made with 'FactoMineR'. They can be rendered as interactive 'HTML' plots, showing useful information at mouse hover. The interest is not mainly visual but statistical. It helps the reader to keep in mind the data contained in the cross-table or Burt table while reading the correspondence analysis, thus preventing over-interpretation. Most graphs are made with 'ggplot2', which means that you can use the + syntax to manually add as many graphical pieces you want, or change theme elements. 3D graphs are made with 'plotly'.

URL https://bricenocenti.github.io/ggfacto/, https://github.com/BriceNocenti/ggfacto

BugReports https://github.com/BriceNocenti/ggfacto/issues

License GPL (>= 3)

Encoding UTF-8

Imports FactoMineR (>= 2.0.0), ggiraph (>= 0.9.6), htmltools (>= 0.5.0), ggplot2 (>= 4.0.0), dplyr (>= 1.0.0), forcats (>= 0.5.0), purrr (>= 0.3.0), rlang (>= 0.4.10), tibble (>= 3.0.0), tidyr (>= 1.0.0), tidyselect (>= 1.1.0), vctrs (>= 0.3.0), ggrepel (>= 0.9.0), fastcluster (>= 1.2.3), gdtools (>= 0.5.0), tabxplor (>= 2.0.1), withr (>= 2.0.0), magrittr (>= 1.5.0), grDevices (>= 4.0.0), graphics (>= 4.0.0), stats (>= 4.0.0), scales (>= 1.1.0)

Suggests GDAtools (>= 2.0), htmlwidgets (>= 1.4.0), knitr (>= 1.30), rmarkdown (>= 2.0), plotly (>= 4.1.0), testthat (>= 3.0.0)

Depends R (>= 4.3.0)

Config/testthat/edition 3

Config/Needs/website pkgdown, BriceNocenti/txtheme

Config/potools/style explicit

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Brice Nocenti [aut, cre],
Francois Husson [ctb, cph] (the cut rule and tree plot of hierarchical_clust(), after 'FactoMineR' HCPC()),
Guillaume Le Ray [ctb, cph] (the cut rule and tree plot of hierarchical_clust(), after 'FactoMineR' HCPC()),
Quentin Molto [ctb, cph] (the cut rule and tree plot of hierarchical_clust(), after 'FactoMineR' HCPC())

Maintainer Brice Nocenti <brice.nocenti@gmail.com>

Repository CRAN

Date/Publication 2026-09-22 17:30:02 UTC

Contents

axis_coord . . . 3
benzecri_mrv . . . 4
clust_tab . . . 4
correspondence_analysis . . . 6
eigenvalues . . . 7
ggca . . . 8
ggfacto . . . 11
ggfacto_plot . . . 13
ggfacto_summary . . . 14
ggfacto_widget . . . 15
ggi . . . 16
ggmca . . . 17
ggmca_3d . . . 23
ggmca_initial_dims . . . 24
ggmca_with_base_ref . . . 25
ggpca . . . 26
ggpca_3d . . . 29
ggpca_cor_circle . . . 32
ggsave2 . . . 33
hierarchical_clust . . . 34
interpret . . . 36
is_in_analysis . . . 38
material_colors_dark . . . 39
material_colors_light . . . 39
mean_sd_tab . . . 40
multiple_correspondence_analysis . . . 41
name_axes . . . 42
principal_component_analysis . . . 43
theme_facto . . . 45

Index **47**

axis_coord *Coordinates of the Individuals on the Axes of an Analysis*

Description

The coordinates of each individual on the axes of a principal component analysis or of a multiple correspondence analysis, to write into the data frame with `dplyr::mutate()`, like `hierarchical_clust`:

‘data <- data |> mutate(axe1 = axis_coord(res, 1))‘

An analysis made with `multiple_correspondence_analysis` is computed on the distinct answer profiles, so its ‘indcoord‘ has one row per profile: ‘axis_coord()‘ gives each individual the coordinate of its profile. In a correspondence analysis, each individual takes the coordinate of its level.

Usage

```
axis_coord(res, axes = 1, margin = "rows")
```

Arguments

`res`	An analysis made with `multiple_correspondence_analysis`, `principal_component_analysis` or `correspondence_analysis` (or with `FactoMineR::MCA()`, `PCA()` or `CA()`, or `GDAtools::speMCA()` or `csMCA()`).
`axes`	The axes. Several axes give a data frame, which `mutate()` writes as several columns, named after ‘axes‘ when it has names (‘c(axe1 = 1, axe2 = 2)‘), else ‘axis1‘, ‘axis2‘...
`margin`	For a correspondence analysis, the variable whose levels give the coordinates: ‘"rows"‘, the default, or ‘"columns"‘.

Value

One value per row of the data frame: inside `dplyr::mutate()`, of the data frame being written, with ‘NA‘ on the rows the analysis did not use (when it was made on a subset of the population); outside, of the data frame the analysis started from. For a correspondence analysis outside `mutate()`, one value per level, named after it.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)

tea <- tea |>
  dplyr::mutate(axe1 = axis_coord(res.mca, 1),
                axis_coord(res.mca, c(axe2 = 2, axe3 = 3)))

# How much of the first axis does age explain?
summary(stats::lm(axe1 ~ age_Q, data = tea))$r.squared
```

benzecri_mrv — *Benzecri's modified rate of variance*

Description

Benzecri's modified rate of variance

Usage

```
benzecri_mrv(res.mca, fmt = FALSE)
```

Arguments

`res.mca`	A multiple correspondence analysis, made with `multiple_correspondence_analysis` (or `FactoMineR::MCA()`, `GDAtools::speMCA()` or `csMCA()`).
`fmt`	By default, the result is given as a numeric vector. Set to 'TRUE' to have a **tabxplor** `link[tabxplor]{fmt}` vector instead.

Value

A numeric vector (or fmt vector with 'fmt = TRUE').

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
benzecri_mrv(res.mca)
```

clust_tab — *Describe Clusters with One Table*

Description

One table describing every cluster: each variable's levels down the page, the clusters across it, and a colour saying at a glance which levels a cluster is made of. Give it the analysis, the data frame and the clusters, in the order of `ggmca`: the active variables, the weights and the rows the analysis was made on are its own. Numeric variables come in as mean rows, coloured by their difference to the mean in standard deviations; the last two rows give each cluster's share of the population and its size (under the table when every row is a mean, as in a principal component analysis).

Usage

```
clust_tab(
  res,
  data,
  clust,
  row_vars,
  pct = "col",
  excl = NA,
  color = "difference",
  row_tot = "% of population",
  cleannames = TRUE,
  ...,
  wt
)
```

Arguments

res	The analysis the clusters were made on, with multiple_correspondence_analysis or principal_component_analysis (or FactoMineR::MCA() or PCA(), or GDAtools::speMCA() or csMCA()). For a correspondence analysis, cross the clusters with the other variable of the table with tabxplor::tab() instead.
data	The data frame, with the clusters. The whole data frame will do when the analysis was made on a subset of it: only the rows the analysis used are described.
clust	The variable with the clusters, typically made with hierarchical_clust, as a bare name or a string.
row_vars	<tidy-select> The variables to describe the clusters with: by default, the active variables of the analysis. Numeric ones become mean rows, unless ‘shape‘ (passed on to tabxplor::tab()) cuts them into levels, e.g. ‘shape = "sd_bands"‘, or ‘shape = c(AGE = "quintiles")‘ for that one.
pct	‘"col"‘ (default) reads each cluster as a distribution: of the people in this cluster, what percentage are in this level. ‘"row"‘ reads each level as a distribution across clusters.
excl	The levels not to show, matched exactly by name; their individuals still count in the percentages. ‘NA‘, the default, hides the missing values (and the levels named ‘<VAR>.NA‘); ‘excl = NULL‘ shows every level.
color	The colour measure, see tab. With ‘"difference"‘ (the default), percentages are coloured by their difference with the whole population, and means by their difference in standard deviations — but not both in one table, which a single ladder cannot grade: there, means stay uncoloured, and ‘"ratio"‘ colours every row.
row_tot	The name of the row giving each cluster’s share of the population.
cleannames	Set to FALSE to keep the level and cluster names as they are, prefix numbers like "1-" and text in parentheses included.
...	Additional arguments to pass to tab.
wt	Not used: the table is weighted with the weights of the analysis. It is there for the former form, ‘clust_tab(data, row_vars, clust, wt)‘, still read as HCPC_tab.

Value

A `tabxplor` table — see [ggfacto_summary] for how it prints.

See Also

[ggfacto_summary], [hierarchical_clust()], [interpret()].

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, nb_clust = 6))

# ONE option decides how every tabxplor table prints, an interpretation table included.
# In a script it goes once, at the top, beside the library() calls.
options(tabxplor.print = "html")

# The clusters, by the active variables
clust_tab(res.mca, tea, clust)

# ... and by other variables
clust_tab(res.mca, tea, clust, row_vars = c(sex, SPC, age), pct = "row")

# A principal component analysis: the means of each cluster
res.pca <- principal_component_analysis(mtcars, 1:7)
cars <- mtcars |>
  dplyr::mutate(clust = hierarchical_clust(res.pca, ncp = 2, nb_clust = 3))
clust_tab(res.pca, cars, clust)
```

---

correspondence_analysis

Correspondence Analysis of a Crosstab

Description

Makes the correspondence analysis of a crosstab with `FactoMineR::CA`: first make the table with `tabxplor::tab()`, then analyse it. The analysis reads the (weighted) counts of the table, whatever it displays, without its Total rows and columns. To leave out some rows or columns, filter the table before, with `dplyr::filter()` and `dplyr::select()`.

Supplementary variables are given in the table itself: in a ‘tab()‘ of several row variables, or several column variables, the first row variable and the first column variable make the active table, and the other variables are supplementary, placed on the axes without taking part in them. ‘tab(data, c(relig, marital), c(partyid, race))‘ analyses ‘relig‘ by ‘partyid‘, and places the levels of ‘marital‘ by their profile over ‘partyid‘, and the levels of ‘race‘ by their profile over ‘relig‘.

Usage

```
correspondence_analysis(table, ncp = Inf, ...)
```

Arguments

`table`	A crosstab made with `tabxplor::tab()`, with one or several row variables and one or several column variables. A matrix or a `table` of counts works too, with its supplementary rows and columns given as in `FactoMineR::CA()` (‘row.sup‘, ‘col.sup‘).
`ncp`	The number of axes to keep. All of them by default.
`...`	Additional arguments to pass to `CA`.

Value

A ‘CA‘ object from **FactoMineR**, which remembers the names of the two active variables, so that `interpret` can print them, and, in ‘source‘, the variable and the name of every row and column of the table.

Examples

```
gss <- forcats::gss_cat |>
  dplyr::filter(!relig %in% c("No answer", "Don't know", "Not applicable"),
                !partyid %in% c("No answer", "Don't know"))
crosstab <- tabxplor::tab(gss, relig, partyid)
res.ca <- correspondence_analysis(crosstab)
interpret(res.ca)                           # the eigenvalues, then the axes

ggfacto(res.ca)                             # the graph
ggfacto(res.ca, interactive = TRUE)         # hover: the profile of each level

# the size of the deviations, which the graph does not show
tabxplor::tab(gss, relig, partyid, pct = "row", color = "contrib")

# marital (rows) and race (columns) are supplementary
res.ca2 <- tabxplor::tab(gss, c(relig, marital), c(partyid, race)) |>
  correspondence_analysis()
ggfacto(res.ca2)
```

`eigenvalues` *The Eigenvalues of an Analysis*

Description

The table of the eigenvalues of the axes, the one `interpret` prints under its table: the variance of each axis, its percentage and the cumulated percentage, and for a multiple correspondence analysis Benzecri's modified rate, which corrects the raw percentages. It is read to choose how many axes to interpret.

Usage

```
eigenvalues(res, n_axes = 8L, color = TRUE, lang = NULL)
```

Arguments

res	An analysis made with multiple_correspondence_analysis, correspondence_analysis or principal_component_analysis (or with FactoMineR::MCA(), CA() or PCA(), or GDAtools::speMCA() or csMCA()).
n_axes	How many axes to print. When some are left out, a last row states how many the cloud has.
color	Set to FALSE to draw no data bar behind the percentages.
lang	NULL (the session's language), "en" or "fr".

Value

A **tabxplor** table, printed as options(tabxplor.print) says.

See Also

[interpret()], [benzecri_mrv()].

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
eigenvalues(res.mca)
```

---

ggca — *Readable and Interactive Graph for Simple Correspondence Analysis*

Description

A readable, complete and beautiful graph of a correspondence analysis. Hovering a level shows its profile — its distribution over the levels of the other variable, each percentage coloured by its difference from the average profile, as in tabxplor::tab(color = "diff") — so the graph is read with the table it draws. The supplementary variables of the table (see correspondence_analysis) are drawn as black italic text, and the clusters of one margin, made with hierarchical_clust, can colour its levels. It is a **ggplot2** graph, to which elements can be added with '+'; pass it to ggi for the interactive version. ggfacto is the same graph, for any analysis.

Usage

```
ggca(
  res.ca,
  axes = c(1, 2),
  show_sup = TRUE,
  xlim,
  ylim,
  out_lims_move = FALSE,
  type = c("points", "text", "labels"),
```

```
  text_repel = TRUE,
  uppercase = "col",
  tooltips = c("row", "col"),
  rowtips_subtitle,
  coltips_subtitle,
  rowcolor_numbers,
  colcolor_numbers,
  cleannames = TRUE,
  filter,
  title,
  text_size = 3.5,
  dist_labels = c("auto", 0.12),
  right_margin = 0,
  size_scale_max = NULL,
  use_theme = TRUE,
  clust,
  color_groups = "^.{0}",
  clust_color_groups = "^.+$",
  keep_levels,
  discard_levels,
  axes_names = NULL,
  axes_reverse = NULL,
  actives_in_bold = TRUE,
  sup_in_italic = TRUE,
  shift_colors = 0,
  colornames_recode,
  scale_color_light = material_colors_light(),
  scale_color_dark = material_colors_dark(),
  get_data = FALSE,
  lang = NULL
)
```

Arguments

res.ca	An analysis made with correspondence_analysis or FactoMineR::CA.
axes	The axes to print, as a numeric vector of length 2.
show_sup	Set to FALSE to leave the supplementary rows and columns out.
xlim, ylim	Horizontal and vertical axes limits, as double vectors of length 2.
out_lims_move	When TRUE, the levels outside xlim or ylim are moved to the edges of the graph rather than left out.
type	How the levels are printed: "points" (coloured points with their names), "text" (coloured names) or "labels" (coloured labels).
text_repel	By default, labels are moved so that they do not overlap. Set to FALSE to print each label exactly at its point.
uppercase	Print the levels of the column variables ("col", the default), of the row variables ("row"), of both or none (NULL) in uppercase.

`tooltips` The tooltips to build: `"row"` for the profiles of the row levels, `"col"` for those of the column levels, both by default.

`rowtips_subtitle`, `coltips_subtitle`, `rowcolor_numbers`, `colcolor_numbers`, `filter`
Deprecated. A tooltip is headed by its variables' names; `color_groups = "^.{2}"` replaces `rowcolor_numbers = 2`, and `discard_levels` replaces `filter`.

`cleannames` Set to `TRUE` to clean levels names, by removing prefix numbers like `"1-"`, and text in parentheses.

`title` The title of the graph.

`text_size` Size of text.

`dist_labels` When `type = "points"`, the distance of the names from the points.

`right_margin` A margin at the right, in cm.

`size_scale_max` The size of the largest point. By default, computed from the spread of the levels' weights.

`use_theme` By default, a specific `ggplot2` theme is used. Set to `FALSE` to customize your own `theme`.

`clust` The clusters of the levels of one margin, as `hierarchical_clust` returns them outside `mutate()`: 'clust = hierarchical_clust(res.ca, ncp = 2, nb_clust = 4)'. The levels of a cluster take its colour, and the cluster is drawn at their barycentre.

`color_groups` One colour per variable by default. A regex matched against each level name makes colour groups within the variables (`"^.{1}"`: upon their first character).

`clust_color_groups`
Color groups for the clusters.

`keep_levels`, `discard_levels`
Regexes (or vectors of them) of the levels to keep, or to leave out.

`axes_names` Names of all the axes (not just the two selected ones), as a character vector.

`axes_reverse` '1' to invert left and right, '2' to invert up and down, '1:2' for both.

`actives_in_bold`
Set the active levels in bold font.

`sup_in_italic` Set the supplementary levels in italics. They are drawn as black text.

`shift_colors` Change the colors of the variables.

`colornames_recode`
A named character vector, in `forcats::fct_recode()` style, to rename the colour groups (printed with 'options(ggfacto.verbose = TRUE)').

`scale_color_light`, `scale_color_dark`
The colours of the points and of the names.

`get_data` Returns the data frames the graph is drawn from, instead of the graph.

`lang` `NULL` (the session's language), `"en"` or `"fr"`.

Value

A `ggplot` object, to which elements can be added with `+`. Sending it through `ggi` draws the interactive graph.

Examples

```
gss <- forcats::gss_cat |>
  dplyr::filter(!relig %in% c("No answer", "Don't know", "Not applicable"),
                !partyid %in% c("No answer", "Don't know"))
res.ca <- correspondence_analysis(tabxplor::tab(gss, relig, partyid))
ggca(res.ca) |>
  ggi()

# the clusters of the religions, drawn among them
ggca(res.ca, clust = hierarchical_clust(res.ca, ncp = 2, nb_clust = 4))
```

---

ggfacto *The Graph of an Analysis*

Description

The graph of a multiple correspondence analysis, a correspondence analysis or a principal component analysis, in the plane of two axes — one verb for the three, as `interpret` is their one table:

- a **multiple correspondence analysis** draws its active levels, and behind them the cloud of its individuals as answer profiles (see `ggmca`);
- a **correspondence analysis** draws the levels of its two variables, and of the supplementary variables its table holds (see `ggca`);
- a **principal component analysis** draws its biplot: the cloud of the individuals with the variables' arrows rescaled onto it (see `ggpca`); 'profiles = FALSE' draws its circle of correlations (see `ggpca_cor_circle`).

Hovering a point shows the data behind it: a level's crosstabs, coloured by their deviations from the mean, an individual's answers or values, a supplementary level's percentages or means. Supplementary variables and clusters are added from the data frame, for an MCA or a PCA, and from the table, for a CA.

Usage

```
ggfacto(
  res,
  data,
  sup_vars,
  clust,
  axes = c(1, 2),
  axes_reverse = NULL,
  type,
  profiles = TRUE,
  active_tables,
  ellipses = NULL,
  title,
```

```
  xlim,
  ylim,
  text_size = 3.5,
  size_scale_max = NULL,
  lang = NULL,
  interactive = FALSE,
  ...
)
```

Arguments

res	An analysis made with multiple_correspondence_analysis, correspondence_analysis or principal_component_analysis (or with FactoMineR::MCA(), CA() or PCA(), or GDAtools::speMCA() or csMCA()).
data	The data frame the analysis was made on, in which to find the supplementary variables and the clusters, for an MCA or a PCA: the whole data frame, even when the analysis was made on a subset of it. A CA reads its table instead.
sup_vars	<tidy-select> The supplementary variables, as in 'tab()': 'sup_vars = c(SEX, AGE)'. For a CA, they are the table's other variables: 'tab(data, c(relig, marital), c(partyid, race))'.
clust	The clusters, made with hierarchical_clust: for an MCA or a PCA, the column of 'data' that holds them ('clust = cah'); for a CA, the clusters of the levels of one margin ('clust = hierarchical_clust(res, ncp = 2, nb_clust = 4)').
axes	The axes to draw, as a numeric vector of length 2.
axes_reverse	'1' to invert left and right, '2' to invert up and down, '1:2' for both.
type	How the levels are drawn: "text", "labels" or "points", and "facets" for one graph per level of the first supplementary variable (MCA, PCA). By default "text", and "points" for a CA.
profiles	Should the cloud of the individuals be drawn? As answer profiles for an MCA, as a biplot for a PCA. By default, yes; 'FALSE' draws the levels alone, and a PCA's circle of correlations when nothing else is asked for.
active_tables	The crosstabs in the tooltips of an MCA: see ggmca.
ellipses	A number between 0 and 1 draws a concentration ellipse around the individuals of each level of the first supplementary variable: 0.5 holds half of them.
title	The title of the graph.
xlim, ylim	Horizontal and vertical limits, as numeric vectors of length 2.
text_size	Size of text.
size_scale_max	The size of the largest point. By default, computed from the spread of the weights of the points drawn.
lang	NULL (the session's language), "en" or "fr": the language of the tooltips and of the axis titles.
interactive	Set to TRUE to get the interactive graph at once, as ggi makes it. By default, a **ggplot2** graph, to which elements can be added with '+', before passing it to ggi.

... Further arguments of the analysis's own graph function, which document them: ggmca (such as 'tooltip_vars', 'max_profiles', 'color_groups'), ggca ('show_sup', 'uppercase', 'tooltips') or ggpca.

Details

An argument that an analysis does not take stops with an explanation: 'data', 'sup_vars', 'profiles', 'active_tables' and 'ellipses' for a CA, 'active_tables' for a PCA.

Value

A ggplot object, or an html widget with 'interactive = TRUE'.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, nb_clust = 5))
ggfacto(res.mca, tea, sup_vars = SPC, clust = clust, interactive = TRUE)

gss <- forcats::gss_cat |>
  dplyr::filter(!relig %in% c("No answer", "Don't know", "Not applicable"),
                !partyid %in% c("No answer", "Don't know"))
res.ca <- correspondence_analysis(tabxplor::tab(gss, c(relig, marital), partyid))
ggfacto(res.ca, interactive = TRUE)

cars <- mtcars
cars$cyl <- factor(cars$cyl)
res.pca <- principal_component_analysis(cars, c(mpg, disp, hp, drat, wt, qsec))
ggfacto(res.pca, cars, sup_vars = cyl, ellipses = 0.5)
```

---

ggfacto_plot *Graphs knitted at their own aspect ratio*

Description

Every graph **ggfacto** draws with **ggplot2** (ggfacto, ggmca, ggca, ggpca, ggpca_cor_circle...) carries the class `ggfacto_plot`, and knows the ratio of its axes: an axis twice as long as the other is drawn twice as long, so the cloud keeps the scale it is interpreted with. In a **knitr** or Quarto document, such a graph is drawn at the chunk's `fig.width` and at the height this ratio gives: no `fig.height` to compute by hand. Captions (`fig.cap`), alignment, `out.width` and cross-references work as for any figure.

Details

A chunk that sets its own `fig.height` or `fig.asp` (different from the document's default) keeps it. The class changes nothing else: the object is still a ggplot, to which **ggplot2** elements can be added with `+`, and ggi still makes it interactive.

ggfacto_summary *The interpretation tables, and how they print*

Description

interpret (for the three analyses), clust_tab and mean_sd_tab all return **one** tabxplor table, so it can be piped, filtered and exported like any other. What differs is only how it is *shown*:

options(tabxplor.print = "html") draws it with tab_html: the Viewer pane in RStudio/Positron, a real html table when knitted. The default, "console", prints the plain tabxplor grid. It is the same option that governs an ordinary crosstab — one to set, once, at the top of a script — and it is read at print time, so it can be set after the table is built. For a text file or a language model, pipe the table into tab_md explicitly. An html summary of *axes* carries no hover tooltip — every figure one would reveal already has a column of its own — while clust_tab, being a crosstab of percentages, keeps them: the count behind each one is worth hovering for.

An analysis-of-axes summary carries the **eigenvalues** as a subordinate table (set_footer_tabs), which every medium renders under it: the percentage of variance of each axis, its cumulated percentage, and for an MCA Benzecri's modified rate — the numbers the rule for choosing how many axes to interpret is read on.

eig = FALSE leaves them out, for a document that shows them already or prints the summary several times to comment it column by column; n_axes says how many of them to print. When some axes are left out — by n_axes, or because ncp truncated the analysis — a final row states how many the cloud has (... of 27). A table showing every axis carries no such row. The Total row is always the whole cloud: 100 % and the total inertia.

min_contrib moves the threshold: NULL (the default) keeps the points contributing more than the mean — Le Roux and Rouanet's rule — 0 keeps them all, and a number keeps what contributes at least that many percent. The summary row's label follows it, so it can never name a set it does not total — and in a correspondence analysis, where each axis carries two such rows, it leads with the margin's own name (Rows: above mean ctr, or the name vars gave it). color = FALSE builds the table with no colour measure at all.

lang is NULL (the session's language), "en" or "fr": it translates what a reader reads as prose — the axis heading, the summary row's label, the words the colour legend uses and the glossary lines under it. **Column names are never translated**: they are the tibble's own names, and a name that changed with the language could not be indexed. These words are fixed when the table is *built*, so an export asking for the other language (tab_md(lang =)) gets tabxplor's grammar translated and ggfacto's nouns as they were written: **build the table in the language you will print it in**.

complete = TRUE widens an MCA or CA summary: each side of the axis gains the point's **coordinate** (its sign says which pole, its size how far out) and its **cos2** (the share of the point's own variance the axis holds), plus the *spread* between the two sides. Neither is coloured there: a coordinate in axis standard deviations has no conventional cut-off, and an MCA cloud has so many axes that every cos2 is small — the 50 % / 75 % rule a interpret table of a PCA reads does not transfer. Both are read by comparing the points shown; only the contribution carries an absolute threshold.

Usage

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

Arguments

x	A table returned by one of the functions of [ggfacto_summary].
...	Passed to tab_html, or to the console print method.

Value

x invisibly (or the rendered object, for html).

The footer

The **colour legend is tabxplor's, saying ggfacto's nouns** — a factorial axis has no chi-squared, so set_legend_words re-states what the ladder grades and nothing else. It is therefore built at render, in the language and the palette of the call that prints it, with its coloured swatches, in all five media. **Nothing to suppress**: a call written by hand is just `interpret(res.mca) |> tab_md(css = FALSE, print = FALSE)`. Under it, one plain line names each statistic the colours do *not* grade.

After a dplyr verb

The subclass is not carried by dplyr (only a table's `tabxplor` attributes are), so a summary that has been through `mutate()` prints as an ordinary `tabxplor` table — the eigenvalues still render under it, and the format is the same `options(tabxplor.print)` either way. What is lost is only the hover policy and the margin names.

See Also

[interpret()], [clust_tab()], [mean_sd_tab()].

---

ggfacto_widget *Widgets written to their own file*

Description

Every interactive graph **ggfacto** returns – ggi (and so `ggfacto(interactive = TRUE)`), ggmca_3d, ggpca_3d – carries the class `ggfacto_widget`. In a **knitr** document, setting

Details

```
options(ggfacto.widget_dir = "auto")
```

makes such a widget write itself to `widget_<chunk label>.html` and put an `<iframe>` in the document instead of several megabytes of inline JSON. Use `"auto"` to follow the chunk's `fig.path`, so the files travel with the document exactly like its figures do, or give a directory path of your own.

This matters for books: **bookdown** merges every chapter into one markdown file and hands it to a single **pandoc** call, and a multi-megabyte raw HTML block on one line is what that reader handles worst.

The chunk must have a label, since the label names the file. Widgets in a chunk with `results = "hide"` are left alone: nothing would reference the file.

The frames share one copy of the JavaScript, which the document itself ships. Option "ggfacto.widget_lib_dir" says where, relative to the document: the default "libs" is what a **bookdown** book uses, while a plain rmarkdown::html_document with self_contained = FALSE puts them in "<name>_files". If it is wrong the frames say so rather than coming out blank.

Unset (the default), the option changes nothing and the widget is embedded inline.

---

ggi *Make a graph interactive*

Description

Make a graph interactive

Usage

```
ggi(
  plot = ggplot2::last_plot(),
  width = NULL,
  height = NULL,
  keep_ratio = TRUE,
  savewidget = FALSE,
  dir = NULL,
  name = "Plot",
  replace = FALSE,
  open = rlang::is_interactive(),
  ...,
  iframe,
  pixel_width
)
```

Arguments

plot	The graph, made with ggfacto (or ggmca, ggca, ggpca, ggpca_cor_circle), with or without **ggplot2** elements added with +. A graph that is already interactive is returned as it is.
width	The width in centimeters. Default to printing device's size.
height	The height in centimeters. Default to printing device's size.
keep_ratio	By default, the height is forced based of the relative size of the MCA's axes. Set to FALSE to avoid this behavior.
savewidget	Should the html widget be saved on disk ? The file is standalone: one single .html carrying its own JavaScript, ready to be sent to someone, or shown in an <iframe>: the graph fills the window or the frame it is opened in.
dir	If saved as file, the directory in which to save the html widget. Default to temporary directory. Set global option "ggfacto.export_dir" with link[base:options](options) to change default directory.

name	The name of the file to save.
replace	Replace file ? By default, number added to find a new name.
open	Should the resulting file be opened at once ?
...	Additional arguments to pass to girafe and dsvg. The widget embeds the Liberation Sans font alone; font_set (see gdtools::font_set()) embeds others.
iframe, pixel_width	Deprecated and ignored: the widget sizes itself.

Value

An html plot, of class ggfacto_widget. In a **knitr** document, setting option "ggfacto.widget_dir" writes it to its own file and embeds an <iframe> instead of the widget itself: see ggfacto_widget.

---

ggmca *Readable and Interactive graph for multiple correspondence analysis*

Description

A readable, complete and beautiful graph for multiple correspondence analysis made with multiple_correspondence_anal ggfacto is the same graph, for any analysis. Interactive tooltips, appearing when hovering near points with mouse, allow to keep in mind many important data (tables of active variables, and additional chosen variables) while reading the graph. Profiles of answers (from the graph of "individuals") are drawn in the back, and can be coloured by the clusters of hierarchical_clust. Since it is made in the spirit of ggplot2, it is possible to change theme or add another plot elements with +. Then, interactive tooltips won't appear until you pass the result through ggi. Step-by-step functions : use ggmca_data to get the data frames with every parameter in a MCA printing, then modify, and pass to ggmca_plot to draw the graph.

Usage

```
ggmca(
  res.mca,
  data,
  sup_vars,
  active_tables = "active",
  tooltip_vars_1lv,
  tooltip_vars,
  axes = c(1, 2),
  axes_names = NULL,
  axes_reverse = NULL,
  type = c("text", "labels", "points", "active_vars_only", "facets"),
  color_groups = "^.{0}",
  clust_color_groups = "^.+$",
  keep_levels,
  discard_levels,
  cleannames = TRUE,
```

```
  profiles = TRUE,
  profiles_tooltip_discard = "^Pas |^Non |^Not |^No ",
  clust,
  max_profiles = 2000,
  alpha_profiles = 0.7,
  color_profiles = TRUE,
  base_profiles_color = "#bbbbbb",
  text_repel = TRUE,
  title,
  actives_in_bold = NULL,
  sup_in_italic = TRUE,
  ellipses = NULL,
  xlim,
  ylim,
  out_lims_move = FALSE,
  shift_colors = 0,
  colornames_recode,
  scale_color_light = material_colors_light(),
  scale_color_dark = material_colors_dark(),
  text_size = 3.5,
  size_scale_max = NULL,
  dist_labels = c("auto", 0.04),
  right_margin = 0,
  use_theme = TRUE,
  get_data = FALSE,
  lang = NULL,
  dat,
  cah,
  cah_color_groups
)

ggmca_data(
  res.mca,
  data,
  sup_vars,
  active_tables = "active",
  tooltip_vars_1lv,
  tooltip_vars,
  color_groups = "^.{0}",
  clust_color_groups = "^.+$",
  keep_levels,
  discard_levels,
  cleannames = TRUE,
  profiles = TRUE,
  profiles_tooltip_discard = "^Pas |^Non |^Not |^No ",
  clust,
  max_profiles = 2000,
  lang = NULL,
```

```
  dat,
  cah,
  cah_color_groups
)

ggmca_plot(
  plot_data,
  axes = c(1, 2),
  axes_names = NULL,
  axes_reverse = NULL,
  type = c("text", "points", "labels", "active_vars_only", "facets"),
  text_repel = TRUE,
  title,
  ellipses = NULL,
  actives_in_bold = NULL,
  sup_in_italic = TRUE,
  xlim,
  ylim,
  out_lims_move = FALSE,
  color_profiles = TRUE,
  base_profiles_color = "#bbbbbb",
  alpha_profiles = 0.7,
  shift_colors = 0,
  colornames_recode,
  scale_color_light = material_colors_light(),
  scale_color_dark = material_colors_dark(),
  text_size = 3.5,
  size_scale_max = NULL,
  dist_labels = c("auto", 0.04),
  right_margin = 0,
  use_theme = TRUE,
  get_data = FALSE,
  data
)
```

Arguments

res.mca	An object created with multiple_correspondence_analysis or FactoMineR::MCA.
data	The data frame the analysis was made on, in which to find the supplementary variables and the clusters: the whole data frame, even when the analysis was made on a subset of it with multiple_correspondence_analysis. Only needed with ‘sup_vars‘, ‘clust‘ or the tooltip variables.
sup_vars	<tidy-select> The supplementary variables to draw, as in ‘tab()‘: ‘sup_vars = c(SEXE, AGE)‘ (strings work too). They need not be given to the analysis before.
active_tables	The coloured crosstabs shown in the tooltips. ‘"active"‘, the default, crosses each active variable with the others: it is the Burt table the analysis was computed from, so a level at the edge of the cloud shows many colours and one

near the centre few. ‘"sup"‘ crosses each supplementary variable with the active ones, ‘c("active", "sup")‘ does both, and ‘NULL‘ none. Percentages are coloured blue when over-represented and red when under-represented, as in `tabxplor::tab(color = "diff")`.

`tooltip_vars_1lv`
: <tidy-select> Variables whose first level (a factor), or weighted mean (a number), is added at the top of the tooltips.

`tooltip_vars`
: <tidy-select> Variables whose levels are all added at the bottom of the tooltips.

`axes`
: The axes to print, as a numeric vector of length 2.

`axes_names`
: Names of all the axes (not just the two selected ones), as a character vector.

`axes_reverse`
: Possibility to reserve the coordinates of the axes by providing a numeric vector : ‘1‘ to invert left and right ; ‘2‘ to invert up and down ; ‘1:2‘ to invert both.

`type`
: Determines the way `sup_vars` are printed.
  - `"text"` : colored text
  - `"points"` : colored points with text legends
  - `"labels"` : colored labels
  - `"active_vars_only"` : the active levels alone, and the answer profiles
  - `"facets"` : one graph of profiles of answer for each levels of the first `sup_vars` (‘profiles = TRUE‘ is not needed). A different color is used for each.

`color_groups`
: By default, there is one color group for all the levels of each ‘sup_vars‘. It is possible to color ‘sup_vars‘ with groups created upon their levels, with a regex matched against each level name (the groups are printed in the console with ‘options(ggfacto.verbose = TRUE)‘). For exemple, ‘color_groups = "^."‘ makes the groups upon the first character of each levels (uselful when their begin by numbers). `color_groups = "^.{3}"` upon the first three characters. `color_groups = "NB.+$"` takes anything between the ‘"NB"‘ and the end of levels names, etc.

`clust_color_groups`
: Color groups for the ‘clust‘ variable (the clusters).

`keep_levels`
: A regex, or a vector of them, matching the supplementary levels to keep: the others are discarded.

`discard_levels`
: A regex, or a vector of them, matching the supplementary levels to discard.

`cleannames`
: Set to `TRUE` to clean levels names, by removing prefix numbers like `"1-"`, and text in parentheses.

`profiles`
: By default, the answer profiles are drawn in the back of the graph, as light-grey points whose tooltips give their answers to the active variables. With `clust`, each profile takes the colour of its cluster, and to hover near one point lights all the points of its cluster. ‘FALSE‘ draws the levels alone.

`profiles_tooltip_discard`
: A regex pattern to remove useless levels among interactive tooltips for profiles of answers (ex. : levels expressing "no" answers).

`clust`
: The variable of ‘data‘ holding the clusters, typically made with `hierarchical_clust`, as a bare name (‘clust = cah_culture‘) or a string. The clusters are drawn as a supplementary variable, and the answer profiles of one cluster are coloured alike and linked at mouse hover (unless ‘profiles = FALSE‘).

max_profiles
: The maximum number of profiles points to print, the heaviest first. Default to 2000.

alpha_profiles
: The alpha (transparency, between 0 and 1) for profiles of answer.

color_profiles
: By default, if `clust` is provided, profiles are colored based on clust levels (HCPC clusters). Set do `FALSE` to avoid this behaviour. You can also give a character vector with only some of the levels of the ‘clust‘ variable .

base_profiles_color
: The base color for answers profiles. Default to gray. Set to ‘NULL‘ to discard profiles. With ‘color_profiles‘, set to ‘NULL‘ to discard the non-colored profiles.

text_repel
: By default, labels are moved so that they do not overlap, with `ggrepel::geom_text_repel`. Set to `FALSE` to print each label exactly at its point, which is faster to draw.

title
: The title of the graph.

actives_in_bold
: Set to ‘TRUE‘ to set active variables in bold font (and sup variables in plain).

sup_in_italic
: Set the supplementary levels in italics, as in every graph of the package. ‘FALSE‘ sets them upright.

ellipses
: Set to a number between 0 and 1 to draw a concentration ellipse for each level of the first `sup_vars`. `0.95` draw ellipses containing 95 individuals of each category. `0.5` draw median-ellipses, containing half the individuals of each category. Every individual counts, whether or not ‘profiles = TRUE‘.

xlim, ylim
: Horizontal and vertical axes limits, as double vectors of length 2.

out_lims_move
: When `TRUE`, the levels out of `xlim` or `ylim` are not removed, but moved to the edges of the graph.

shift_colors
: Change colors of the `sup_vars` points.

colornames_recode
: A named character vector with `fct_recode` style to rename the colour groups. They are printed in the console with ‘options(ggfacto.verbose = TRUE)‘.

scale_color_light
: A scale color for sup vars points

scale_color_dark
: A scale color for sup vars texts

text_size
: Size of text.

size_scale_max
: The size of the largest point. By default, computed from the spread of the weights of the points drawn, so that the median answer profile stays visible.

dist_labels
: When `type = points`, the distance of labels from points.

right_margin
: A margin at the right, in cm. Useful to read tooltips over points placed at the right of the graph without formatting problems.

use_theme
: By default, a specific `ggplot2` theme is used. Set to `FALSE` to customize your own `theme`.

get_data
: Returns the data frame to create the plot instead of the plot itself.

lang
: `NULL` (the session’s language), `"en"` or `"fr"`: the language of the tooltips and of the axis titles.

dat — Deprecated former name of ‘data‘. Still accepted, with a warning; use ‘data‘ instead.

cah, cah_color_groups — Deprecated former names of ‘clust‘ and ‘clust_color_groups‘.

plot_data — A list of data frames made with ggmca_data.

Value

A ggplot object to be printed in the ‘RStudio‘ Plots pane. Possibility to add other gg objects with +. Sending the result through ggi will draw the interactive graph in the Viewer pane using girafe.

A list to pass to ggmca_plot: ‘vars_data‘ (one row per level), ‘ind_data‘ (one row per answer profile, with ‘profiles = TRUE‘), ‘individuals‘ (one row per individual, with ‘sup_vars‘), ‘res‘, ‘clust‘ and ‘lang‘.

A ggplot object.

Functions

- ggmca_data(): get the data frames with all parameters to print a MCA graph
- ggmca_plot(): draws a plot model — the one ggmca_data() returns, and the one the correspondence and principal component graphs build internally.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)

# Interactive graph for multiple correspondence analysis :
ggmca(res.mca, tea, sup_vars = SPC, ylim = c(NA, 1.2)) |>
  ggi() # to make the graph interactive

# Hover a level: its crosstabs with every other active variable, the Burt table the analysis
#  was computed from. Points near the middle show few colours, points at the edges plenty.
ggmca(res.mca, ylim = c(NA, 1.2)) |>
  ggi()

# Graph with colored clusters (hierarchical clustering on the first three axes)
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, nb_clust = 6))
ggmca(res.mca, tea, clust = clust)

# Concentration ellipses for each levels of a supplementary variable :
ggmca(res.mca, tea, sup_vars = SPC, ylim = c(NA, 1.2),
  ellipses = 0.5, profiles = TRUE)

# Graph of profiles of answer for each levels of a supplementary variable :
ggmca(res.mca, tea, sup_vars = SPC, ylim = c(NA, 1.2),
  type = "facets", ellipses = 0.5, profiles = TRUE)
```

ggmca_3d *Interactive 3D Plot for Multiple Correspondence Analyses (plotly::)*

Description

Interactive 3D Plot for Multiple Correspondence Analyses (plotly::)

Usage

```
ggmca_3d(
  res.mca,
  data,
  clust,
  axes = 1:3,
  base_zoom = 1,
  remove_buttons = FALSE,
  cone_size = 0.15,
  view = "All",
  camera_view,
  aspectratio_from_eig = FALSE,
  title,
  ind_name.size = 10,
  max_point_size = 30,
  ...,
  dat,
  cah
)
```

Arguments

res.mca	An object created with multiple_correspondence_analysis or FactoMineR::MCA.
data	The data frame the analysis was made on, in which to find the clusters.
clust	The variable of ‘data‘ holding the clusters, typically made with hierarchical_clust, as a bare name or a string: the answer profiles of one cluster are coloured alike and linked at mouse hover.
axes	The axes to print, as a numeric vector of length 3.
base_zoom	The base level of zoom.
remove_buttons	Set to TRUE to remove buttons to change view.
cone_size	The size of the conic arrow at the end of each axe.
view	The starting point of view (in 3D) : • "Plane 1-2" : Axes 1 and 2. • "Plane 1-3" : Axes 1 and 3. • "Plane 2-3" : Axes 2 and 3. • "All" : A 3D perspective with Axes 1, 2, 3.

camera_view	Possibility to add a (replace ‘view‘)
aspectratio_from_eig	Set to ‘TRUE‘ to modify axes length based on eigenvalues.
title	The title of the graph.
ind_name.size	The size of the names of individuals.
max_point_size	The size of the biggest point.
...	Additional arguments to pass to ggmca.
dat	Deprecated former name of ‘data‘. Still accepted, with a warning; use ‘data‘ instead.
cah	Deprecated former name of ‘clust‘.

Value

A plotly html interactive 3d (or 2d) graph.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
ggmca_3d(res.mca)

# 3D graph with colored clusters
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, nb_clust = 6))
ggmca_3d(res.mca, tea, clust = clust)
```

---

ggmca_initial_dims — *Plot Initial Dimensions (Active Variables) of Multiple Correspondence Analysis*

Description

This function mostly have an educational value : it shows the initial dimensions of the Multiple Correspondence Analysis (active variables) in their initial reference frame. It shows the n dimensional space before the analysis is done. To see initial dimensions axes in the space built by the analysis (principal axes), use ggmca_with_base_ref.

Usage

```
ggmca_initial_dims(
  res.mca,
  data,
  proj_just = c(1.5, 2),
  cleannames = TRUE,
  keep = NULL
)
```

Arguments

res.mca	An object created with multiple_correspondence_analysis or FactoMineR::MCA.
data	The data frame the analysis was made on. Optional: the analysis keeps its levels in the data's order. It is accepted so that every 'ggmca_*' function takes '(res.mca, data)'.
proj_just	Horizontal justification of text of the coordinates on axes, as a character vector of length 2 (x and y).
cleannames	Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parentheses.
keep	A character vector of the name of active variables to keep.

Value

A ggplot object to be printed in the 'RStudio' Plots pane. Possibility to add other gg objects with +. Sending the result through ggi will draw the interactive graph in the Viewer pane using girafe.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
ggmca_initial_dims(res.mca, tea)
```

---

ggmca_with_base_ref *Plot Initial Dimensions (Active Variables) on a Multiple Correspondence Analyses*

Description

This function mostly have an educational value : it shows the initial dimensions of the Multiple Correspondence Analysis (active variables) in the space built by the analysis (principal axes). To see initial dimensions in their initial reference frame, use ggmca_initial_dims.

Usage

```
ggmca_with_base_ref(res.mca, data, axes = c(1, 2), keep = NULL)
```

Arguments

res.mca	An object created with FactoMineR::MCA.
data	The data the analysis was made on. Optional: this graph draws only active variables, which are read from 'res.mca', so it changes nothing. It is accepted so that every 'ggmca_*' function takes '(res.mca, data)'.
axes	The axes to print, as a numeric vector of length 2.
keep	A character vector of the name of active variables to keep.

Value

A ggplot object to be printed in the ‘RStudio‘ Plots pane. Possibility to add other gg objects with +. Sending the result through ggi will draw the interactive graph in the Viewer pane using girafe.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
ggmca_with_base_ref(res.mca)

# It is more readable to select just a few active variables
lv2_vars <- dplyr::select(tea[1:18], where(~ nlevels(.) == 2)) |> names()
ggmca_with_base_ref(res.mca, keep = lv2_vars)

lv3_vars <- dplyr::select(tea[1:18], where(~ nlevels(.) == 3)) |> names()
ggmca_with_base_ref(res.mca, keep = lv3_vars)

lv4_vars <- dplyr::select(tea[1:18], where(~ nlevels(.) == 4)) |> names()
ggmca_with_base_ref(res.mca, keep = lv4_vars)

lv6_vars <- dplyr::select(tea[1:18], where(~ nlevels(.) == 6)) |> names()
ggmca_with_base_ref(res.mca, keep = lv6_vars)
```

---

ggpca — *Readable and Interactive Graph of the Individuals of a Principal Component Analysis*

Description

The biplot of a principal component analysis: the cloud of its individuals in the plane of two axes, the levels of supplementary variables at the barycentre of their individuals, the clusters of hierarchical_clust coloured, and the active variables as arrows, the circle of correlations rescaled onto the cloud. Only the DIRECTION of an arrow reads there: an individual lies towards the variables it scores high on; the length of an arrow and the distance between an arrow and an individual mean nothing, and ggpca_cor_circle draws the correlations at their own scale. Hovering an individual shows its value on each active variable; hovering a supplementary level shows the mean of each active variable among its individuals, coloured by its standardized difference from the population's (blue above, red below), as clust_tab does. ggfacto draws it for a principal component analysis, unless ‘profiles = FALSE‘ leaves nothing but the circle.

Usage

```
ggpca(
  res.pca,
  data,
  sup_vars,
  axes = c(1, 2),
```

```
  axes_names = NULL,
  axes_reverse = NULL,
  type = c("text", "labels", "points", "facets"),
  color_groups = "^.{0}",
  clust_color_groups = "^.+$",
  keep_levels,
  discard_levels,
  cleannames = TRUE,
  profiles = TRUE,
  clust,
  max_profiles = 2000,
  alpha_profiles = 0.7,
  color_profiles = TRUE,
  base_profiles_color = "#bbbbbb",
  text_repel = TRUE,
  title,
  sup_in_italic = TRUE,
  ellipses = NULL,
  xlim,
  ylim,
  out_lims_move = FALSE,
  shift_colors = 0,
  colornames_recode,
  scale_color_light = material_colors_light(),
  scale_color_dark = material_colors_dark(),
  text_size = 3.5,
  size_scale_max = NULL,
  dist_labels = c("auto", 0.04),
  right_margin = 0,
  use_theme = TRUE,
  get_data = FALSE,
  lang = NULL,
  variables = TRUE
)
```

Arguments

res.pca	An analysis made with principal_component_analysis or FactoMineR::PCA().
data	The data frame the analysis was made on, in which to find the supplementary variables and the clusters: the whole data frame, even when the analysis was made on a subset of it. Only needed with ‘sup_vars‘ or ‘clust‘.
sup_vars	<tidy-select> The supplementary variables, as in ‘tab()‘: each level is drawn at the weighted barycentre of its individuals.
axes	The axes to print, as a numeric vector of length 2.
axes_names	Names of all the axes (not just the two selected ones), as a character vector.
axes_reverse	Possibility to reserve the coordinates of the axes by providing a numeric vector : ‘1‘ to invert left and right ; ‘2‘ to invert up and down ; ‘1:2‘ to invert both.

type — Determines the way sup_vars are printed: "text", "labels", "points", or "facets" (one graph of the individuals of each level of the first sup_vars).

color_groups — By default, there is one color group for all the levels of each ‘sup_vars‘. It is possible to color ‘sup_vars‘ with groups created upon their levels, with a regex matched against each level name (the groups are printed in the console with ‘options(ggfacto.verbose = TRUE)‘). For exemple, ‘color_groups = "^."‘ makes the groups upon the first character of each levels (uselful when their begin by numbers). color_groups = "^.{3}" upon the first three characters. color_groups = "NB.+$" takes anything between the ‘"NB"‘ and the end of levels names, etc.

clust_color_groups — Color groups for the ‘clust‘ variable (the clusters).

keep_levels — A regex, or a vector of them, matching the supplementary levels to keep: the others are discarded.

discard_levels — A regex, or a vector of them, matching the supplementary levels to discard.

cleannames — Set to TRUE to clean levels names, by removing prefix numbers like "1-", and text in parentheses.

profiles — By default the individuals are drawn: ‘FALSE‘ draws the supplementary levels and the variables alone.

clust — The variable of ‘data‘ holding the clusters, typically made with hierarchical_clust: the individuals of one cluster are coloured alike and linked at mouse hover.

max_profiles — The maximum number of individuals to draw: the heaviest first, and, among individuals of equal weight, an evenly spread sample.

alpha_profiles — The alpha (transparency, between 0 and 1) for profiles of answer.

color_profiles — By default, if clust is provided, profiles are colored based on clust levels (HCPC clusters). Set do FALSE to avoid this behaviour. You can also give a character vector with only some of the levels of the ‘clust‘ variable .

base_profiles_color — The base color for answers profiles. Default to gray. Set to ‘NULL‘ to discard profiles. With ‘color_profiles‘, set to ‘NULL‘ to discard the non-colored profiles.

text_repel — By default, labels are moved so that they do not overlap, with ggrepel::geom_text_repel. Set to FALSE to print each label exactly at its point, which is faster to draw.

title — The title of the graph.

sup_in_italic — Set the supplementary levels in italics, as in every graph of the package. ‘FALSE‘ sets them upright.

ellipses — Set to a number between 0 and 1 to draw a concentration ellipse for each level of the first sup_vars. 0.95 draw ellipses containing 95 individuals of each category. 0.5 draw median-ellipses, containing half the individuals of each category. Every individual counts, whether or not ‘profiles = TRUE‘.

xlim, ylim — Horizontal and vertical axes limits, as double vectors of length 2.

out_lims_move — When TRUE, the levels out of xlim or ylim are not removed, but moved to the edges of the graph.

shift_colors — Change colors of the sup_vars points.

colornames_recode
: A named character vector with fct_recode style to rename the colour groups. They are printed in the console with 'options(ggfacto.verbose = TRUE)'.

scale_color_light
: A scale color for sup vars points

scale_color_dark
: A scale color for sup vars texts

text_size
: Size of text.

size_scale_max
: The size of the largest point. By default, computed from the spread of the weights of the points drawn, so that the median answer profile stays visible.

dist_labels
: When type = points, the distance of labels from points.

right_margin
: A margin at the right, in cm. Useful to read tooltips over points placed at the right of the graph without formatting problems.

use_theme
: By default, a specific ggplot2 theme is used. Set to FALSE to customize your own theme.

get_data
: Returns the data frame to create the plot instead of the plot itself.

lang
: NULL (the session's language), "en" or "fr": the language of the tooltips and of the axis titles.

variables
: By default the active variables are drawn as arrows. Set to 'FALSE' to draw the individuals alone.

Value

A ggplot object, to which elements can be added with +. Sending it through ggi draws the interactive graph.

Examples

```
cars <- mtcars
cars$cyl <- factor(cars$cyl)
res.pca <- principal_component_analysis(cars, c(mpg, disp, hp, drat, wt, qsec))
ggpca(res.pca, cars, sup_vars = cyl) |>
  ggi()

cars <- cars |>
  dplyr::mutate(clust = hierarchical_clust(res.pca, ncp = 2, nb_clust = 3))
ggpca(res.pca, cars, clust = clust)
```

---

ggpca_3d *Interactive 3D Plot for Principal Component Analyses (plotly::)*

Description

Interactive 3D Plot for Principal Component Analyses (plotly::)

Usage

```
ggpca_3d(
  res.pca,
  axes = c(1, 2, 3),
  princ_axes_print = -3:3,
  base_axe_n_breaks = 10,
  ind.size = 4,
  ind_name.size = 3,
  title,
  center = TRUE,
  var_names_on = "var",
  base_zoom = 1,
  remove_buttons = FALSE,
  cone_size = 0.33,
  view = "All",
  type = c("var", "ind", "main_plan", "projections"),
  camera_view,
  aspectratio_from_eig = FALSE,
  always_make_ind_tooltips = FALSE,
  var_color = "#4D4D4D",
  max_ind = 500,
  max_ind_seed
)
```

Arguments

`res.pca` The result of `FactoMineR::PCA`.

`axes` The axes to print, as a numeric vector of length 3 (or 2).

`princ_axes_print` The breaks of the principal axes.

`base_axe_n_breaks` The number of breaks in initial variables axes.

`ind.size` The size of the points of individuals.

`ind_name.size` The size of the names of individuals.

`title` Plot title.

`center` By default the plot is centered on the central point. Set to ‘FALSE‘ to center on the origin of all variables (zero coordinates).

`var_names_on` By default ‘"var"‘ the names of variables are drawn upon the initial axes. Set to ‘"cor"‘ to draw them upon correlation vectors instead.

`base_zoom` The base level of zoom.

`remove_buttons` Set to TRUE to remove buttons to change view.

`cone_size` The size of the conic arrow at the end of each axe.

`view` The starting point of view (in 3D) :

- `"Plane 1-2"` : Axes 1 and 2.

- "Plane 1-3" : Axes 1 and 3.
- "Plane 2-3" : Axes 2 and 3.
- "All" : A 3D perspective with Axes 1, 2, 3.

type — Which elements of the graph to print, among : #'

- "var" : initial variables axes, with breaks
- "cor" : normalized correlation vectors (length = 1)
- "cor_sphere" : a 3D sphere of standard deviation 1
- "ind" : points of individuals
- "ind_name" : names of individuals
- "main_plan" : the plan 1-2.
- "projections" : projections of mean point on initial variables
- "V" : vectors of the V transition matrix
- "vs" : vectors of the matrix of singular values

camera_view — Possibility to add a (replace 'view')

aspectratio_from_eig — Set to 'TRUE' to modify axes length based on eigenvalues.

always_make_ind_tooltips — Set to 'TRUE' to add interactive toolips for individuals.

var_color — The color of the initial variables/dimensions

max_ind — The maximun number of individuals to print.

max_ind_seed — The random seed used to sample individuals.

Value

A plotly html interactive 2d or 3d graph.

Examples

```
data(mtcars, package = "datasets")
mtcars <- mtcars[1:7] |> dplyr::rename(weight = wt)
res.pca <- FactoMineR::PCA(mtcars, graph = FALSE)

# Variables and individuals
ggpca_3d(res.pca)

# Circle of correlation 3D
ggpca_3d(res.pca, type = c("cor", "cor_sphere"),
         var_names_on = "cor", base_zoom = 0.6,
         princ_axes_print = -1:1, view = "All"
)
```

ggpca_cor_circle *Correlation Circle Plot for Principal Component Analysis*

Description

The active variables of a principal component analysis as arrows in the circle of correlations: the coordinate of a variable on an axis is its correlation with it. Hovering a variable shows its coordinates, and its projections on the two axes. ggfacto draws it for a principal component analysis with ‘profiles = FALSE‘ and nothing else asked for; by default, it draws the same arrows over the cloud of individuals.

Usage

```
ggpca_cor_circle(
  res.pca,
  axes = c(1, 2),
  proj = FALSE,
  interactive = FALSE,
  text_size = 3.5,
  lang = NULL,
  axes_names = NULL,
  axes_reverse = NULL,
  title,
  xlim,
  ylim
)
```

Arguments

res.pca	An analysis made with principal_component_analysis or FactoMineR::PCA.
axes	The axes to print, as a numeric vector of length 2.
proj	Set to ‘TRUE‘ to print projections of vectors over the two axes.
interactive	Set to ‘TRUE‘ to get the interactive graph at once, as ggi would make it. By default, a ggplot, to which elements can be added with ‘+‘ before passing it to ggi.
text_size	Size of the text.
lang	NULL (the session’s language), "en" or "fr".
axes_names	Names of all the axes, as a character vector.
axes_reverse	‘1‘ to invert left and right, ‘2‘ to invert up and down, ‘1:2‘ for both.
title	The title of the graph.
xlim, ylim	Horizontal and vertical limits, as numeric vectors of length 2.

Value

A ggplot, or an html widget with ‘interactive = TRUE‘.

Examples

```
data(mtcars, package = "datasets")
mtcars <- mtcars[1:7] |> dplyr::rename(weight = wt)
res.pca <- principal_component_analysis(mtcars, 1:7)
ggpca_cor_circle(res.pca)
ggpca_cor_circle(res.pca) |> ggi()  # interactive
```

---

ggsave2 *Save a plot as image*

Description

Save a plot as image

Usage

```
ggsave2(
  plot = ggplot2::last_plot(),
  dir = NULL,
  name = "Plot",
  xt = "png",
  dpi = 600,
  width = 21,
  height,
  scale = 1,
  replace = FALSE,
  open = rlang::is_interactive()
)
```

Arguments

plot	The plot, created with **ggplot2**.
dir	If saved as file, the directory in which to save the html widget. Default to temporary directory. Set global option "ggfacto.export_dir" with link[base:options]{options} to change default directory.
name	The name of the file to save.
xt	The extension name, when saving as image (interactive graph will always be .html).
dpi	The resolution.
width	The width in centimeters.
height	The height in centimeters. By default, width/1.41.
scale	Fixed ratio between horizontal and vertical axes.
replace	Replace file ? By default, number added to find a new name.
open	Should the resulting file be opened at once ?

Value

Creates a file, and opens it in ‘RStudio‘ viewer, as a side effect.

---

hierarchical_clust *Hierarchical Clustering on the Axes of an Analysis*

Description

Clusters the individuals of a principal component analysis or of a multiple correspondence analysis, or the levels of one margin of a correspondence analysis, on the first ‘ncp‘ axes. The clusters are those of `FactoMineR::HCPC`: Ward’s hierarchical clustering, cut into ‘nb_clust‘ clusters, then consolidated by k-means. Use it inside `dplyr::mutate()` to add them to the data frame:

‘data <- data |> mutate(clust = hierarchical_clust(res, ncp = 3, nb_clust = 6))‘

To choose the number of clusters, look at the tree first: ‘hierarchical_clust(res, ncp = 3)‘. The tree is built once: cutting it again, into another number of clusters or with names, is instant. Name the clusters with ‘names‘, then describe them with `clust_tab` and draw them with `ggfacto(clust = )`.

Usage

```
hierarchical_clust(
  res,
  ncp,
  nb_clust = -1,
  tree = nb_clust == -1,
  consol = TRUE,
  margin = "rows",
  names = NULL
)
```

Arguments

`res`	An analysis made with `multiple_correspondence_analysis`, `principal_component_analysis` or `correspondence_analysis` (or with `FactoMineR::MCA()`, `PCA()` or `CA()`, or `GDAtools::speMCA()` or `csMCA()`, whose subcloud alone is clustered).
`ncp`	The number of axes to cluster on: the first ones, those worth interpreting (see the eigenvalues under `interpret`). There is no default: on every axis, the clustering would follow noise.
`nb_clust`	The number of clusters. With ‘-1‘, the default, the tree is cut where the gain in between-cluster inertia drops the most. Given ‘names‘, it is the number of names.
`tree`	Should the clustering tree be drawn, to choose the number of clusters? By default, only when ‘nb_clust = -1‘. Its leaves are the distinct points of the cloud: the answer profiles of a multiple correspondence analysis, the levels of a correspondence analysis. Each bar of the inertia gains carries the number of clusters

it makes, and the title the share of the inertia of the ‘ncp‘ axes that lies between the clusters drawn.

`consol` The k-means consolidation, which moves each individual to its nearest cluster once the tree is cut. ‘TRUE‘, the default, is `FactoMineR::HCPC`’s k-means, which counts every individual once, weights or not. ‘"weighted"‘ counts each by its weight (the survey weights, or a level’s count in a correspondence analysis) with a simpler k-means, Lloyd’s, which can place a few individuals differently even without weights. ‘FALSE‘ keeps the clusters as the tree cuts them.

`margin` For a correspondence analysis, the levels to cluster: ‘"rows"‘, the default, or ‘"columns"‘.

`names` The names of the clusters, in the order the levels should take: either ‘c("Name 1", "Name 2", ...)‘, for clusters 1, 2, etc., or ‘c("Name" = 2, "Other name" = 1, ...)‘, as in `forcats::fct_recode()`. Clusters are numbered along the first axis, so the names belong to one cut of one tree: write them after looking at that cut.

Details

The tree is kept in memory for the session, under a key made of everything it is built from — the coordinates on the ‘ncp‘ axes, the weights, the answer profiles — so a tree made from other data is never reused. The last 20 trees are kept; ‘options(ggfacto.clust_cache = 50)‘ keeps more, and ‘options(ggfacto.clust_cache = 0)‘ none.

Value

A factor with the clusters, ‘"1"‘, ‘"2"‘, etc. or their ‘names‘, numbered along the first axis and returned invisibly: a bare call only draws the tree. Inside `dplyr::mutate()`, it has one value per row of that data frame, and ‘NA‘ on the rows the analysis did not use (when it was made on a subset of the population). Outside `mutate()`, it has one value per row of the data frame the analysis started from. For a correspondence analysis, each row of the data frame gets the cluster of its level (‘NA‘ for a level outside the table); outside `mutate()`, there is one value per level, named after it.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)

# The tree, to choose the number of clusters, then the clusters, written into the data frame
hierarchical_clust(res.mca, ncp = 3)
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, nb_clust = 6))

clust_tab(res.mca, tea, clust)
ggfacto(res.mca, tea, clust = clust)

# Named, in the order of your choice (the tree is not built again)
tea <- tea |>
  dplyr::mutate(clust = hierarchical_clust(res.mca, ncp = 3, names = c(
    "Cluster A" = 1, "Cluster B" = 2, "Cluster C" = 3, "Cluster E" = 5, "Cluster D" = 4,
    "Cluster F" = 6
  )))
```

```
# On a subset of the population: the other rows get NA
res.mca_young <- tea |>
  dplyr::filter(age < 30) |>
  multiple_correspondence_analysis(1:18)
tea <- tea |>
  dplyr::mutate(clust_young = hierarchical_clust(res.mca_young, ncp = 3, nb_clust = 4))

# A correspondence analysis clusters the levels of one margin: each individual gets the
# cluster of its level
res.ca <- forcats::gss_cat |>
  tabxplor::tab(relig, partyid) |>
  correspondence_analysis()
gss <- forcats::gss_cat |>
  dplyr::mutate(relig_clust = hierarchical_clust(res.ca, ncp = 2, nb_clust = 4))
```

`interpret` *Interpret the Axes of an Analysis*

Description

One table to read the axes of a factorial analysis, whatever the analysis:

- a **multiple correspondence analysis**: per axis, the active levels contributing more than the mean contribution, the positive side facing the negative one, and the spread between the two sides in percent of each question's contribution (Brigitte Le Roux and Henri Rouanet, *Geometric data analysis*, Kluwer, 2004; Brigitte Le Roux, *Analyse geometrique des donnees multidimensionnelles*, Dunod, 2014);
- a **correspondence analysis**: the same for the row points and the column points, each margin against its own mean contribution, since each sums to 100 different number of points;
- a **principal component analysis**: each active variable's mean and spread, then, per axis, its coordinate — which under 'scale.unit' IS its correlation with the axis —, its contribution and its cos2.

The eigenvalues of the axes travel under the table, with Benzecri's modified rate for an MCA. 'mca_interpret()' and 'pca_interpret()' are the same tables, for one analysis each.

Usage

```
interpret(res, ...)

mca_interpret(
  res.mca,
  axes = 1:5,
  complete = FALSE,
  min_contrib = NULL,
  color = TRUE,
  eig = TRUE,
```

```
  n_axes = 8L,
  lang = NULL,
  type = NULL,
  spread = NULL
)

pca_interpret(
  res.pca,
  axes = 1:3,
  color = TRUE,
  eig = TRUE,
  n_axes = 8L,
  lang = NULL
)
```

Arguments

res	An analysis made with multiple_correspondence_analysis, correspondence_analysis or principal_component_analysis (or with FactoMineR::MCA(), CA() or PCA(), or GDAtools::speMCA() or csMCA()).
...	The arguments below. A correspondence analysis takes one more, ‘vars‘: the two margins’ names, as in ‘vars = c("CSER", "PR2017")‘. By default, the names correspondence_analysis kept; after a bare FactoMineR::CA(), which keeps none, the table says “Rows” and “Columns”.
res.mca, res.pca	The analysis, for ‘mca_interpret()‘ and ‘pca_interpret()‘.
axes	The axes to interpret, as an integer vector. By default, the first five of an MCA, two of a CA, three of a PCA.
complete	For an MCA or a CA, set to TRUE for the fuller summary: each side of the axis gains the point’s coordinate and its cos2, and the table gains the spread between the two sides.
min_contrib	For an MCA or a CA, the contribution threshold, in percent. NULL (the default) is the mean contribution of the point’s own set; 0 keeps every point.
color	Set to FALSE to build the table with no colour measure, and no data bar under the eigenvalues.
eig	The eigenvalues travel under the table. Set to FALSE in a document that already shows them, or that prints the summary several times to comment it column by column.
n_axes	How many axes the eigenvalue table prints. When some are left out, an ellipsis row states how many the cloud has.
lang	NULL (the session’s language), "en" or "fr".
type	Deprecated. The output format is now options(tabxplor.print), or an explicit tab_md / tab_html call — see [ggfacto_summary].
spread	Deprecated. Folded into complete.

Value

A tabxplor table — see [ggfacto_summary] for how it prints.

See Also

[ggfacto_summary], [benzecri_mrv()].

Examples

```
# ONE option decides how every tabxplor table prints, an interpretation table included.
# In a script it goes once, at the top, beside the library() calls.
options(tabxplor.print = "html")

data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
interpret(res.mca)
interpret(res.mca, axes = 1:2, complete = TRUE)

# a correspondence analysis draws the STRUCTURE of a crosstab's deviations and says nothing of
# their size, so the crosstab is asked for beside it, never instead of it:
crosstab <- tabxplor::tab(forcats::gss_cat, race, marital)
interpret(correspondence_analysis(crosstab))
tabxplor::tab(forcats::gss_cat, race, marital, pct = "row", color = "contrib", test = TRUE)

cars <- dplyr::rename(mtcars[1:7], weight = wt)
interpret(principal_component_analysis(cars, 1:7))
```

---

is_in_analysis *Which Rows an Analysis Was Made On*

Description

‘TRUE‘ for each row of the data frame the analysis was made on, ‘FALSE‘ for the others: the rows filtered out (with the pipe or with ‘filter‘), those with a weight of 0, and the supplementary individuals of a principal component analysis. Use it to describe the analysed population:

‘data |> dplyr::filter(is_in_analysis(res)) |> tabxplor::tab(SEXE, AGE)‘

Usage

```
is_in_analysis(res)
```

Arguments

res	An analysis made with multiple_correspondence_analysis or principal_component_analysis (or with FactoMineR::MCA() or PCA(), or GDAtools::speMCA() or csMCA()).

Value

A logical vector, one value per row: inside `dplyr::filter()` or `dplyr::mutate()`, of the data frame being read; outside, of the data frame the analysis started from.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18, filter = age < 30)

tea |>
  dplyr::filter(is_in_analysis(res.mca)) |>
  tabxplor::tab(sex, SPC)
```

`material_colors_dark` *Dark Material palette for MCA level names*

Description

Dark Material palette for MCA level names

Usage

```
material_colors_dark()
```

Value

A character vector of color codes, with color names.

Examples

```
material_colors_dark()
```

`material_colors_light` *Light Material palette for MCA points*

Description

Light Material palette for MCA points

Usage

```
material_colors_light()
```

Value

A character vector of color codes, with color names.

Examples

```
material_colors_light()
```

`mean_sd_tab` *Simple Mean and SD Summary (deprecated)*

Description

One row per numeric variable: its base, its mean, its standard deviation, and its coefficient of variation — the standard deviation as a percentage of the mean, which is what lets two variables measured in different units be compared for how dispersed they are.

Deprecated: `interpret` now opens with the same three figures, taken from the analysis itself, so the description and the interpretation are one table and cannot disagree. Use it instead; this function still works and will be removed in a future release.

Usage

```
mean_sd_tab(data, vars, wt)
```

Arguments

`data`	A data.frame.
`vars`	<tidy-select> The names of the numeric variables to compute means and sds with.
`wt`	The name of the weight variable, if needed.

Value

A `tabxplor` table — see [ggfacto_summary] for how it prints.

See Also

[ggfacto_summary], [interpret()].

Examples

```
mean_sd_tab(mtcars, 1:7)
```

multiple_correspondence_analysis
Multiple Correspondence Analysis

Description

A user-friendly wrapper around MCA, made to work with **ggfacto** functions like ggmca, interpret and hierarchical_clust. Variables are selected the way of the ‘tidyverse‘, as in tabxplor::tab(). Supplementary variables are not given here: they are added afterwards, in ggmca.

‘MCA2()‘ keeps the fit of **ggfacto** 0.3.2, on the individuals: ‘$ind‘ has one row per analysed row, so that ‘FactoMineR::HCPC()‘ of it classifies the individuals, in their order. It gives the same graphs, tables and clusters as ‘multiple_correspondence_analysis()‘, more slowly on large data, and will be deprecated.

Usage

```
multiple_correspondence_analysis(
  data,
  active_vars,
  wt,
  excl = NA,
  ncp = Inf,
  graph = FALSE,
  filter,
  ...
)

MCA2(data, active_vars, wt, excl = NA, ncp = Inf, graph = FALSE, filter, ...)
```

Arguments

data	The data frame. To analyse a subset of the population, give the whole data frame and ‘filter‘, or filter it inside the call with the native pipe, ‘data |> dplyr::filter(...) |> multiple_correspondence_analysis(...)‘: the analysis then remembers which rows it used, so that ggmca, hierarchical_clust or is_in_analysis can be given the whole data frame afterwards.
active_vars	<tidy-select> The active variables.
wt	<tidy-select> The weight variable, if any.
excl	The levels to exclude from the calculation of the axes (specific multiple correspondence analysis), matched exactly by name. The missing values of each active variable become a level named ‘<VAR>.NA‘, and ‘NA‘, the default, excludes all of them: ‘excl = NA‘ for missing values only, ‘excl = c(NA, "Other")‘ to exclude a level too, ‘excl = "DIPLOMA.NA"‘ for the missing values of one variable only, ‘excl = NULL‘ to keep every level.

ncp	The number of axes to keep. All of them by default: the eigenvalue table is how one chooses how many axes to interpret, and a truncated one cannot show the drop — it also renormalises Benzecri's modified rate over the axes it kept, so the same axis gets a different rate. To cluster on the first axes, give `hierarchical_clust` its own 'ncp'.
graph	By default no graph is made, since the result can be plotted with `ggfacto`.
filter	A condition on the rows of 'data', as in `dplyr::filter()`: only the rows where it is 'TRUE' are analysed ('filter = AGE >= 18').
...	Additional arguments to pass to `MCA`, except those that index its rows or columns ('ind.sup', 'quali.sup', 'quanti.sup', 'tab.disj').

Value

A 'MCA' object from **FactoMineR**, fitted on the distinct answer profiles (the combinations of active answers), each weighted by its individuals: the eigenvalues and every result on the levels are the individuals', and '$ind' has one row per profile (per individual with 'MCA2()'). Use `axis_coord` and `hierarchical_clust` to write coordinates and clusters into the data frame ('FactoMineR::HCPC()' would cluster the profiles). One more element, 'source', records for each row of 'data' its row of '$ind' ('NA' if it was not analysed) and its weight.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
interpret(res.mca)                          # the eigenvalues, then the axes

ggfacto(res.mca, tea, sup_vars = c(sex, SPC))  # the graph, with supplementary variables
ggfacto(res.mca, tea, sup_vars = c(sex, SPC), interactive = TRUE)  # hover: the crosstables


# A subset of the population: the analysis remembers which rows it used
res.mca_young <- tea |>
  dplyr::filter(age < 30) |>
  multiple_correspondence_analysis(1:18)
# the same analysis
res.mca_young <- multiple_correspondence_analysis(tea, 1:18, filter = age < 30)
```

`name_axes` *Name the Axes of an Analysis*

Description

Gives the axes of an analysis the names its interpretation arrived at: every graph (`ggfacto`) prints them in its axis titles, and every interpretation table (`interpret`) in its axis headings. The names are given in the order of the axes; an empty name, '""', leaves an axis unnamed, so that 'name_axes(res, "", "")' can wait in a script to be filled.

Usage

```
name_axes(res, ...)
```

Arguments

res	An analysis made with multiple_correspondence_analysis, correspondence_analysis or principal_component_analysis (or with 'FactoMineR' or 'GDAtools').
...	The names, as character strings: the first names axis 1, the second axis 2, and so on. A name given as ‘"3" = "..."‘ names axis 3 alone, and leaves the others as they are.

Value

The analysis, its axes named.

Examples

```
data(tea, package = "FactoMineR")
res.mca <- multiple_correspondence_analysis(tea, 1:18)
interpret(res.mca, axes = 1:2)

res.mca <- name_axes(res.mca, "tea as a habit / tea as an outing", "tea time / tea shop")

ggfacto(res.mca)
```

---

principal_component_analysis *Principal Component Analysis*

Description

A user-friendly wrapper around PCA, made to work with **ggfacto** functions like interpret, ggfacto and hierarchical_clust. Variables are selected the way of the ‘tidyverse‘, as in tabxplor::tab(). ‘PCA2()‘ is its name in **ggfacto** 0.3.2, kept for former code.

Usage

```
principal_component_analysis(
  data,
  active_vars,
  wt,
  col.w = NULL,
  ind_name,
  scale.unit = TRUE,
  ind.sup = NULL,
  ncp = Inf,
```

```
  graph = FALSE,
  na = "mean",
  filter,
  ...
)

PCA2(
  data,
  active_vars,
  wt,
  col.w = NULL,
  ind_name,
  scale.unit = TRUE,
  ind.sup = NULL,
  ncp = Inf,
  graph = FALSE,
  na = "mean",
  filter,
  ...
)
```

Arguments

data	The data frame. To analyse a subset of the population, give the whole data frame and ‘filter‘, or filter it inside the call with the native pipe, ‘data |> dplyr::filter(...) |> principal_component_analysis(...)‘: the analysis then remembers which rows it used, so that ggpca, hierarchical_clust or is_in_analysis can be given the whole data frame afterwards.
active_vars	<tidy-select> The names of the active variables.
wt	<tidy-select> The weight variable, if any.
col.w	The weights of the columns, as a numeric vector of the same length than ‘active_vars.‘
ind_name	<tidy-select> Possibly, the variable holding the names of the individuals.
scale.unit	A boolean, if ‘TRUE‘ (value set by default) then data are scaled to unit variance.
ind.sup	A vector indicating the indexes of the supplementary individuals, rows of ‘data‘.
ncp	Number of dimensions kept in the results. All of them by default: the eigenvalue table is how one chooses how many axes to interpret, and a truncated one cannot show the drop. To cluster on the first axes, give hierarchical_clust its own ‘ncp‘.
graph	A boolean, set to ‘TRUE‘ to display the base graph.
na	How missing values of the active variables are treated. ‘"mean"‘, the default, places each one at its variable’s weighted mean, where it adds nothing to the axes (as an excluded level does in a specific multiple correspondence analysis); the tooltips and interpret count them. ‘"drop"‘ leaves out the rows with a missing value.

filter	A condition on the rows of ‘data‘, as in `dplyr::filter()`: only the rows where it is ‘TRUE‘ are analysed (‘filter = AGE >= 18‘). Supplementary individuals are kept.
...	Additional arguments to pass to `PCA`.

Value

A ‘PCA‘ object from **FactoMineR**, with one more element, ‘source‘, which records for each row of ‘data‘ its row in the analysis (‘NA‘ if it was not analysed).

Examples

```
cars <- dplyr::mutate(mtcars, cyl = factor(cyl))
res.pca <- principal_component_analysis(cars, c(mpg, disp, hp, drat, wt, qsec))
interpret(res.pca)                          # the eigenvalues, then the axes

ggfacto(res.pca, cars, sup_vars = cyl)      # the individuals and the variables (biplot)
ggfacto(res.pca, profiles = FALSE)          # the circle of correlations alone
ggfacto(res.pca, cars, sup_vars = cyl, interactive = TRUE)  # hover: the means
```

`theme_facto` *A ggplot2 Theme for Geometrical Data Analysis*

Description

A ggplot2 Theme for Geometrical Data Analysis

Usage

```
theme_facto(
  res,
  axes = c(1, 2),
  legend.position = c("none", "left", "right", "bottom", "top"),
  no_color_scale = FALSE,
  size_scale_max = 8,
  xlim,
  ylim
)
```

Arguments

res	An analysis, made with `multiple_correspondence_analysis`, `correspondence_analysis`, `principal_component_analysis`, FactoMineR or GDAtools.
axes	The axes to print, as a numeric vector of length 2.
legend.position	One of `c("none", "left", "right", "bottom", "top")`.

`no_color_scale`	When TRUE, you can provide color_scale next without warning.
`size_scale_max`	Maximum size of the points.
`xlim`	Horizontal axe limits.
`ylim`	Vertical axe limits.

Value

A list of ggplot2 objects.

Index

axis_coord, 3, *42*

benzecri_mrv, 4

CA, *6, 7, 9*
clust_tab, 4, *14*, *26*, *34*
correspondence_analysis, *3*, 6, *8, 9*, *12*, *34*, *37*, *43*, *45*

dsvg, *17*

eigenvalues, 7

FactoMineR::PCA, *30*, *32*
fct_recode, *21*, *29*

gdtools::font_set(), *17*
geom_text_repel, *21*, *28*
ggca, 8, *11*, *13*, *16*
ggfacto, *8*, 11, *13*, *16, 17*, *26*, *32*, *34*, *42, 43*
ggfacto_plot, 13
ggfacto_summary, 14
ggfacto_widget, 15, *17*
ggi, *8*, *10*, *12, 13*, *15*, 16, *17*, *22*, *25, 26*, *29*, *32*
ggmca, *4*, *11–13*, *16*, 17, *24*, *41*
ggmca_3d, *15*, 23
ggmca_data, *17*, *22*
ggmca_data *(ggmca)*, 17
ggmca_initial_dims, 24, *25*
ggmca_plot, *17*, *22*
ggmca_plot *(ggmca)*, 17
ggmca_with_base_ref, *24*, 25
ggpca, *11*, *13*, *16*, 26, *44*
ggpca_3d, *15*, 29
ggpca_cor_circle, *11*, *13*, *16*, *26*, 32
ggplot, *10*, *13*, *22*, *25, 26*, *29*, *32*
ggplot2, *17*
ggsave2, 33
girafe, *17*, *22*, *25, 26*

HCPC, *34, 35*
HCPC_tab, *5*
hierarchical_clust, *3*, *5*, *8*, *10*, *12*, *17*, *20*, *23*, *26*, *28*, 34, *41–44*

interpret, *7*, *11*, *14*, *34*, 36, *40–44*
is_in_analysis, 38, *41*, *44*

material_colors_dark, 39
material_colors_light, 39
MCA, *19*, *23*, *25*, *41, 42*
MCA2 *(multiple_correspondence_analysis)*, 41
mca_interpret *(interpret)*, 36
mean_sd_tab, *14*, 40
multiple_correspondence_analysis, *3–5*, *8*, *12*, *17*, *19*, *23*, *25*, *34*, *37, 38*, 41, *43*, *45*

name_axes, 42

PCA, *43*, *45*
PCA2 *(principal_component_analysis)*, 43
pca_interpret *(interpret)*, 36
plotly, *24*, *31*
principal_component_analysis, *3*, *5*, *8*, *12*, *27*, *32*, *34*, *37, 38*, *43*, 43, *45*
print.ggfacto_summary *(ggfacto_summary)*, 14

set_footer_tabs, *14*
set_legend_words, *15*

tab, *5*
tab_html, *14, 15*, *37*
tab_md, *14*, *37*
theme, *10*, *21*, *29*
theme_facto, 45
tidy-select, *5*, *12*, *19, 20*, *27*, *40, 41*, *44*