Package ‘sjPlot’

April 2, 2023

Type Package
Encoding UTF-8
Title Data Visualization for Statistics in Social Science
Version 2.8.14
Maintainer Daniel Lüdecke <d.luedecke@uke.de>
Description Collection of plotting and table output functions for data visualization. Results of various statistical analyses (that are commonly used in social sciences) can be visualized using this package, including simple and cross tabulated frequencies, histograms, box plots, (generalized) linear models, mixed effects models, principal component analysis and correlation matrices, cluster analyses, scatter plots, stacked scales, effects plots of regression models (including interaction terms) and much more. This package supports labelled data.
License GPL-3
Depends R (>= 3.6)
Imports graphics, grDevices, stats, utils, bayestestR, datawizard, dplyr, ggeffects, ggplot2 (>= 3.2.0), knitr, insight, MASS, parameters, performance, purr, rlang, scales, sjlabelled (>= 1.1.2), sjmisc (>= 2.8.2), sjstats (>= 0.17.8), tidyr (>= 1.0.0)
Suggests brms, car, clubSandwich, cluster, cowplot, haven, GPArotation, ggrepel, glmmTMB, gridExtra, ggridges, httr, lme4, nFactors, pscl, psych, rmarkdown, rstanarm, sandwich, splines, survey, TMB, testthat
URL https://strengejacke.github.io/sjPlot/
BugReports https://github.com/strengejacke/sjPlot/issues
RoxygenNote 7.2.3
VignetteBuilder knitr
NeedsCompilation no
Author  Daniel Lüdecke [aut, cre] (<https://orcid.org/0000-0002-8895-3206>),
        Alexander Bartel [ctb] (<https://orcid.org/0000-0002-1280-6138>),
        Carsten Schwemmer [ctb],
        Chuck Powell [ctb] (<https://orcid.org/0000-0002-3606-2188>),
        Amir Djalovski [ctb],
        Johannes Titz [ctb] (<https://orcid.org/0000-0002-1102-5719>)

Repository  CRAN

Date/Publication  2023-04-02 15:30:02 UTC

R topics documented:

sjPlot-package .................................................. 3
dist_chisq ..................................................... 4
dist_f ............................................................ 5
dist_norm .......................................................... 6
dist_t .............................................................. 7
efc ................................................................. 8
plot_frq ............................................................ 9
plot_gpt ........................................................... 13
plot_grid .......................................................... 16
plot_grpfreq ....................................................... 17
plot_kfold_cv ...................................................... 22
plot_likert ........................................................ 23
plot_model ........................................................ 28
plot_models ......................................................... 38
plot_residuals ...................................................... 43
plot_scatter ....................................................... 45
plot_stackfreq ...................................................... 48
plot_xtab .......................................................... 50
save_plot .......................................................... 54
set_theme ........................................................... 56
sjp.aov1 ........................................................... 61
sjp.chi2 ............................................................ 63
sjp.corr ............................................................. 65
sjp.poly ............................................................. 66
sjplot ............................................................... 69
sjPlot-themes ....................................................... 70
tab_corr ............................................................ 73
tab_df ............................................................... 76
tab_fa ............................................................... 79
tab_itemscale ........................................................ 81
tab_model ........................................................... 85
tab_pca ............................................................. 93
tab_stackfreq ...................................................... 96
tab_xtab ............................................................. 99
view_df ............................................................. 103
Description

Collection of plotting and table output functions for data visualization. Results of various statistical analyses (that are commonly used in social sciences) can be visualized using this package, including simple and cross tabulated frequencies, histograms, box plots, (generalized) linear models, mixed effects models, PCA and correlation matrices, cluster analyses, scatter plots, Likert scales, effects plots of interaction terms in regression models, constructing index or score variables and much more.

The package supports labelled data, i.e. value and variable labels from labelled data (like vectors or data frames) are automatically used to label the output. Own labels can be specified as well.

What does this package do?

In short, the functions in this package mostly do two things:

1. compute basic or advanced statistical analyses
2. either plot the results as ggplot-figure or print them as html-table

How does this package help me?

One of the more challenging tasks when working with R is to get nicely formatted output of statistical analyses, either in graphical or table format. The sjPlot-package takes over these tasks and makes it easy to create beautiful figures or tables.

There are many examples for each function in the related help files and a comprehensive online documentation at https://strengejacke.github.io/sjPlot/.

A note on the package functions

The main functions follow specific naming conventions, hence starting with a specific prefix, which indicates what kind of task these functions perform.

- sjc - cluster analysis functions
- sjp - plotting functions
- sjt - (HTML) table output functions

Author(s)

Daniel Lüdecke <d.luedecke@uke.de>
**dist_chisq**  
*Plot chi-squared distributions*

**Description**

This function plots a simple chi-squared distribution or a chi-squared distribution with shaded areas that indicate at which chi-squared value a significant p-level is reached.

**Usage**

```r
dist_chisq(
  chi2 = NULL,
  deg.f = NULL,
  p = NULL,
  xmax = NULL,
  geom.colors = NULL,
  geom.alpha = 0.7
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>chi2</code></td>
<td>Numeric, optional. If specified, a chi-squared distribution with <code>deg.f</code> degrees of freedom is plotted and a shaded area at <code>chi2</code> value position is plotted that indicates whether or not the specified value is significant or not. If both <code>chi2</code> and <code>p</code> are not specified, a distribution without shaded area is plotted.</td>
</tr>
<tr>
<td><code>deg.f</code></td>
<td>Numeric. The degrees of freedom for the chi-squared distribution. Needs to be specified.</td>
</tr>
<tr>
<td><code>p</code></td>
<td>Numeric, optional. If specified, a chi-squared distribution with <code>deg.f</code> degrees of freedom is plotted and a shaded area at the position where the specified p-level starts is plotted. If both <code>chi2</code> and <code>p</code> are not specified, a distribution without shaded area is plotted.</td>
</tr>
<tr>
<td><code>xmax</code></td>
<td>Numeric, optional. Specifies the maximum x-axis-value. If not specified, the x-axis ranges to a value where a p-level of 0.00001 is reached.</td>
</tr>
<tr>
<td><code>geom.colors</code></td>
<td>user defined color for geoms. See 'Details' in plot_grpfrq.</td>
</tr>
<tr>
<td><code>geom.alpha</code></td>
<td>Specifies the alpha-level of the shaded area. Default is 0.7, range between 0 to 1.</td>
</tr>
</tbody>
</table>

**Examples**

```r
# a simple chi-squared distribution
# for 6 degrees of freedom
dist_chisq(deg.f = 6)

# a chi-squared distribution for 6 degrees of freedom,
# and a shaded area starting at chi-squared value of ten.
# With a df of 6, a chi-squared value of 12.59 would be "significant",
```
Thus, the shaded area from 10 to 12.58 is filled as "non-significant", while the area starting from chi-squared value 12.59 is filled as "significant". 

```
dist_chisq(chi2 = 10, deg.f = 6)
```

# a chi-squared distribution for 6 degrees of freedom, 
# and a shaded area starting at that chi-squared value, which has  
# a p-level of about 0.125 (which equals a chi-squared value of about 10).  
# With a df of 6, a chi-squared value of 12.59 would be "significant",  
# thus the shaded area from 10 to 12.58 (p-level 0.125 to p-level 0.05)  
# is filled as "non-significant", while the area starting from chi-squared  
# value 12.59 (p-level < 0.05) is filled as "significant". 

dist_chisq(p = 0.125, deg.f = 6)

---

**dist_f**  

**Plot F distributions**

**Description**

This function plots a simple F distribution or an F distribution with shaded areas that indicate at which F value a significant p-level is reached.

**Usage**

```
dist_f( 
  f = NULL, 
  deg.f1 = NULL, 
  deg.f2 = NULL, 
  p = NULL, 
  xmax = NULL, 
  geom.colors = NULL, 
  geom.alpha = 0.7
)
```

**Arguments**

- **f** Numeric, optional. If specified, an F distribution with **deg.f1** and **deg.f2** degrees of freedom is plotted and a shaded area at **f** value position is plotted that indicates whether or not the specified value is significant or not. If both **f** and **p** are not specified, a distribution without shaded area is plotted.

- **deg.f1** Numeric. The first degrees of freedom for the F distribution. Needs to be specified.

- **deg.f2** Numeric. The second degrees of freedom for the F distribution. Needs to be specified.

- **p** Numeric, optional. If specified, a F distribution with **deg.f1** and **deg.f2** degrees of freedom is plotted and a shaded area at the position where the specified p-level starts is plotted. If both **f** and **p** are not specified, a distribution without shaded area is plotted.
dist_norm

Plot normal distributions

Description

This function plots a simple normal distribution or a normal distribution with shaded areas that indicate at which value a significant p-level is reached.

Usage

```r
dist_norm(
  norm = NULL,
  mean = 0,
  sd = 1,
  p = NULL,
  xmax = NULL,
  geom.colors = NULL,
  geom.alpha = 0.7
)
```

Arguments

- **norm**: Numeric, optional. If specified, a normal distribution with `mean` and `sd` is plotted and a shaded area at `norm` value position is plotted that indicates whether or not the specified value is significant or not. If both `norm` and `p` are not specified, a distribution without shaded area is plotted.
dist_t

**Description**

This function plots a simple t-distribution or a t-distribution with shaded areas that indicate at which t-value a significant p-level is reached.

**Usage**

```r
dist_t(
  t = NULL,
  deg.f = NULL,
  p = NULL,
  xmax = NULL,
  geom.colors = NULL,
  geom.alpha = 0.7
)
```
Arguments

**t**
Numeric, optional. If specified, a t-distribution with *deg.f* degrees of freedom is plotted and a shaded area at *t* value position is plotted that indicates whether or not the specified value is significant or not. If both *t* and *p* are not specified, a distribution without shaded area is plotted.

**deg.f**
Numeric. The degrees of freedom for the t-distribution. Needs to be specified.

**p**
Numeric, optional. If specified, a t-distribution with *deg.f* degrees of freedom is plotted and a shaded area at the position where the specified p-level starts is plotted. If both *t* and *p* are not specified, a distribution without shaded area is plotted.

**xmax**
Numeric, optional. Specifies the maximum x-axis-value. If not specified, the x-axis ranges to a value where a p-level of 0.00001 is reached.

**geom.colors**
user defined color for geoms. See 'Details' in `plot_grpfrq`.

**geom.alpha**
Specifies the alpha-level of the shaded area. Default is 0.7, range between 0 to 1.

Examples

```r
# a simple t-distribution
# for 6 degrees of freedom
dist_t(deg.f = 6)

# a t-distribution for 6 degrees of freedom,
# and a shaded area starting at t-value of one.
# With a df of 6, a t-value of 1.94 would be "significant".
dist_t(t = 1, deg.f = 6)

# a t-distribution for 6 degrees of freedom,
# and a shaded area starting at p-level of 0.4
# (t-value of about 0.26).
dist_t(p = 0.4, deg.f = 6)
```

---

**efc**

*Sample dataset from the EUROFAMCARE project*

Description

A SPSS sample data set, imported with the `read_spss` function.
**plot_freq**

*Plot frequencies of variables*

**Description**

Plot frequencies of a variable as bar graph, histogram, box plot etc.

**Usage**

```r
plot_freq(
  data,
  ...,  
  title = "",
  weight.by = NULL,
  title.wtd.suffix = NULL,
  sort.frq = c("none", "asc", "desc"),
  type = c("bar", "dot", "histogram", "line", "density", "boxplot", "violin"),
  geom.size = NULL,
  geom.colors = "#336699",
  errorbar.color = "darkred",
  axis.title = NULL,
  axis.labels = NULL,
  xlim = NULL,
  ylim = NULL,
  wrap.title = 50,
  wrap.labels = 20,
  grid.breaks = NULL,
  expand.grid = FALSE,
  show.values = TRUE,
  show.n = TRUE,
  show.prc = TRUE,
  show.axis.values = TRUE,
  show.ci = FALSE,
  show.na = FALSE,
  show.mean = FALSE,
  show.mean.val = TRUE,
  show.sd = TRUE,
  drop.empty = TRUE,
  mean.line.type = 2,
  mean.line.size = 0.5,
  inner.box.width = 0.15,
  inner.box.dotsize = 3,
  normal.curve = FALSE,
  normal.curve.color = "red",
  normal.curve.size = 0.8,
  normal.curve.alpha = 0.4,
  auto.group = NULL,
)```

coord.flip = FALSE,
vjust = "bottom",
hjust = "center",
y.offset = NULL
)

Arguments

data
A data frame, or a grouped data frame.

...  Optional, unquoted names of variables that should be selected for further processing. Required, if data is a data frame (and no vector) and only selected variables from data should be processed. You may also use functions like : or tidyselect's select_helpers.

title
Character vector, used as plot title. By default, response_labels is called to retrieve the label of the dependent variable, which will be used as title. Use title = "" to remove title.

weight.by
Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.

title.wtd.suffix
Suffix (as string) for the title, if weight.by is specified, e.g. title.wtd.suffix=" (weighted)". Default is NULL, so title will not have a suffix when cases are weighted.

sort.frq  Determines whether categories should be sorted according to their frequencies or not. Default is "none", so categories are not sorted by frequency. Use "asc" or "desc" for sorting categories ascending or descending order.

type
Specifies the plot type. May be abbreviated.
"bar" for simple bars (default)
"dot" for a dot plot
"histogram" for a histogram (does not apply to grouped frequencies)
"line" for a line-styled histogram with filled area
"density" for a density plot (does not apply to grouped frequencies)
"boxplot" for box plot
"violin" for violin plots

geom.size
size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.

geom.colors
User defined color for geoms, e.g. geom.colors = "#0080ff".

errorbar.color
Color of confidence interval bars (error bars). Only applies to type = "bar". In case of dot plots, error bars will have same colors as dots (see geom.colors).

axis.title
Character vector of length one or two (depending on the plot function and type), used as title(s) for the x and y axis. If not specified, a default labelling is chosen.

Note: Some plot types do not support this argument. In such cases, use the return value and add axis titles manually with labs, e.g.: $plot.list[[1]] + labs(x = ...)
axis.labels character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.

xlim Numeric vector of length two, defining lower and upper axis limits of the x scale. By default, this argument is set to NULL, i.e. the x-axis fits to the required range of the data.

ylim numeric vector of length two, defining lower and upper axis limits of the y scale. By default, this argument is set to NULL, i.e. the y-axis fits to the required range of the data.

wrap.title Numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

grid.breaks numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks'th position a major grid is being printed.

expand.grid logical, if TRUE, the plot grid is expanded, i.e. there is a small margin between axes and plotting region. Default is FALSE.

show.values Logical, whether values should be plotted or not.

show.n logical, if TRUE, adds total number of cases for each group or category to the labels.

show.prc logical, if TRUE (default), percentage values are plotted to each bar. If FALSE, percentage values are removed.

show.axis.values logical, whether category, count or percentage values for the axis should be printed or not.

show.ci Logical, if TRUE, adds notches to the box plot, which are used to compare groups; if the notches of two boxes do not overlap, medians are considered to be significantly different.

show.na logical, if TRUE, NA's (missing values) are added to the output.

show.mean Logical, if TRUE, a vertical line in histograms is drawn to indicate the mean value of the variables. Only applies to histogram-charts.

show.mean.val Logical, if TRUE (default), the mean value is printed to the vertical line that indicates the variable’s mean. Only applies to histogram-charts.

show.sd Logical, if TRUE, the standard deviation is annotated as shaded rectangle around the mean intercept line. Only applies to histogram-charts.

drop.empty Logical, if TRUE and the variable's values are labeled, values / factor levels with no occurrence in the data are omitted from the output. If FALSE, labeled values that have no observations are still printed in the table (with frequency 0).

mean.line.type Numeric value, indicating the linetype of the mean intercept line. Only applies to histogram-charts and when show.mean = TRUE.

mean.line.size Numeric, size of the mean intercept line. Only applies to histogram-charts and when show.mean = TRUE.

inner.box.width width of the inner box plot that is plotted inside of violin plots. Only applies if type = "violin". Default value is 0.15
inner.box.dotsize

size of mean dot inside a violin or box plot. Applies only when type = "violin" or "boxplot".

normal.curve

Logical, if TRUE, a normal curve, which is adjusted to the data, is plotted over the histogram or density plot. Default is FALSE. Only applies when histograms or density plots are plotted (see type).

normal.curve.color

Color of the normal curve line. Only applies if normal.curve = TRUE.

normal.curve.size

Numeric, size of the normal curve line. Only applies if normal.curve = TRUE.

normal.curve.alpha

Transparency level (alpha value) of the normal curve. Only applies if normal.curve = TRUE.

auto.group

numeric value, indicating the minimum amount of unique values in the count variable, at which automatic grouping into smaller units is done (see group_var). Default value for auto.group is NULL, i.e. auto-grouping is off. See group_var for examples on grouping.

coord.flip

logical, if TRUE, the x and y axis are swapped.

vjust

character vector, indicating the vertical position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

hjust

character vector, indicating the horizontal position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

y.offset

numeric, offset for text labels when their alignment is adjusted to the top/bottom of the geom (see hjust and vjust).

Value

A ggplot-object.

Note

This function only works with variables with integer values (or numeric factor levels), i.e. scales / centered variables with fractional part may result in unexpected behaviour.

Examples

library(sjlabelled)
data(efc)
data(iris)

# simple plots, two different notations
plot_frq(iris, Species)
plot_frq(efc$tot_sc_e)
# boxplot
plot_frq(efc$e17age, type = "box")

if (require("dplyr")) {
    # histogram, pipe-workflow
efc %>%
        dplyr::select(e17age, c160age) %>%
        plot_frq(type = "hist", show.mean = TRUE)

    # bar plot(s)
    plot_frq(efc, e42dep, c172code)
}

if (require("dplyr") && require("gridExtra")) {
    # grouped data frame, all panels in one plot
    efec %>%
        group_by(e42dep) %>%
        plot_frq(c161sex) %>%
        plot_grid()
}

library(sjmisc)
# grouped variable
ageGrp <- group_var(efc$e17age)
ageGrpLab <- group_labels(efc$e17age)
plot_frq(ageGrp, title = get_label(efc$e17age), axis.labels = ageGrpLab)

# plotting confidence intervals. expand grid and v/hjust for text labels
plot_frq(
    efec$e15relat, type = "dot", show.ci = TRUE, sort.frq = "desc",
    coord.flip = TRUE, expand.grid = TRUE, vjust = "bottom", hjust = "left"
)

# histogram with overlayed normal curve
plot_frq(efc$c160age, type = "h", show.mean = TRUE, show.mean.val = TRUE,
    normal.curve = TRUE, show.sd = TRUE, normal.curve.color = "blue",
    normal.curve.size = 3, ylim = c(0,50))

---

**plot_gpt**

*Plot grouped proportional tables*

**Description**

Plot grouped proportional crosstables, where the proportion of each level of x for the highest category in y is plotted, for each subgroup of grp.
Usage

plot_gpt(
  data,
  x,
  y,
  grp,
  colors = "metro",
  geom.size = 2.5,
  shape.fill.color = "#f0f0f0",
  shapes = c(15, 16, 17, 18, 21, 22, 23, 24, 25, 7, 8, 9, 10, 12),
  title = NULL,
  axis.labels = NULL,
  axis.titles = NULL,
  legend.title = NULL,
  legend.labels = NULL,
  wrap.title = 50,
  wrap.labels = 15,
  wrap.legend.title = 20,
  wrap.legend.labels = 20,
  axis.lim = NULL,
  grid.breaks = NULL,
  show.total = TRUE,
  annotate.total = TRUE,
  show.p = TRUE,
  show.n = TRUE
)

Arguments

data A data frame, or a grouped data frame.
x Categorical variable, where the proportion of each category in x for the highest category of y will be printed along the x-axis.
y Categorical or numeric variable. If not a binary variable, y will be recoded into a binary variable, dichotomized at the highest category and all remaining categories.
grp Grouping variable, which will define the y-axis.
colors May be a character vector of color values in hex-format, valid color value names (see demo("colors")). A name of a pre-defined color palette. Following options are valid for the colors argument:
  • If not specified, a default color brewer palette will be used, which is suitable for the plot style.
  • If "gs", a greyscale will be used.
  • If "bw", and plot-type is a line-plot, the plot is black/white and uses different line types to distinguish groups (see this package-vignette).
  • If colors is any valid color brewer palette name, the related palette will be used. Use RColorBrewer::display.brewer.all() to view all available palette names.
There are some pre-defined color palettes in this package, see sjPlot-themes for details.
Else specify own color values or names as vector (e.g. colors = "#00ff00" or colors = c("firebrick", "blue").

geom.size  size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.

shape.fill.color  Optional color vector, fill-color for non-filled shapes

shapes  Numeric vector with shape styles, used to map the different categories of x.

title  Character vector, used as plot title. By default, response_labels is called to retrieve the label of the dependent variable, which will be used as title. Use title = "" to remove title.

axis.labels  character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.

axis.titles  character vector of length one or two, defining the title(s) for the x-axis and y-axis.

legend.title  Character vector, used as legend title for plots that have a legend.

legend.labels  character vector with labels for the guide/legend.

wrap.title  numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels  numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

wrap.legend.title  numeric, determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

wrap.legend.labels  numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

axis.lim  Numeric vector of length 2, defining the range of the plot axis. Depending on plot type, may effect either x- or y-axis, or both. For multiple plot outputs (e.g., from type = "eff" or type = "slope" in plot_model), axis.lim may also be a list of vectors of length 2, defining axis limits for each plot (only if non-faceted).

grid.breaks  numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks’th position a major grid is being printed.

show.total  Logical, if TRUE, a total summary line for all aggregated grp is added.

annotate.total  Logical, if TRUE and show.total = TRUE, the total-row in the figure will be highlighted with a slightly shaded background.

show.p  Logical, adds significance levels to values, or value and variable labels.

show.n  logical, if TRUE, adds total number of cases for each group or category to the labels.

Details

The p-values are based on chisq.test of x and y for each grp.
Value

A ggplot-object.

Examples

```r
if (requireNamespace("haven")) {
  data(efc)

  # the proportion of dependency levels in female elderly, for each family carer's relationship to elderly
  plot_gpt(efc, e42dep, e16sex, e15relat)

  # proportion of educational levels in highest dependency category of elderly, for different care levels
  plot_gpt(efc, c172code, e42dep, n4pstu)
}
```

---

**plot_grid**

**Arrange list of plots as grid**

Description

Plot multiple ggplot-objects as a grid-arranged single plot.

Usage

```r
plot_grid(x, margin = c(1, 1, 1, 1), tags = NULL)
```

Arguments

- **x**
  
  A list of ggplot-objects. See 'Details'.

- **margin**
  
  A numeric vector of length 4, indicating the top, right, bottom and left margin for each plot, in centimetres.

- **tags**
  
  Add tags to your subfigures. Can be `TRUE` (letter tags) or character vector containing tags labels.

Details

This function takes a list of ggplot-objects as argument. Plotting functions of this package that produce multiple plot objects (e.g., when there is an argument `facet.grid`) usually return multiple plots as list (the return value is named `plot.list`). To arrange these plots as grid as a single plot, use `plot_grid`.

Value

An object of class `gtable`. 
Examples

```r
if (require("dplyr") && require("gridExtra")) {
  library(ggeffects)
data(efc)

  # fit model
  fit <- glm(
    tot_sc_e ~ c12hour + e17age + e42dep + neg_c_7,
    data = efc,
    family = poisson
  )

  # plot marginal effects for each predictor, each as single plot
  p1 <- ggpredict(fit, "c12hour") %>%
       plot(show.y.title = FALSE, show.title = FALSE)
p2 <- ggpredict(fit, "e17age") %>%
       plot(show.y.title = FALSE, show.title = FALSE)
p3 <- ggpredict(fit, "e42dep") %>%
       plot(show.y.title = FALSE, show.title = FALSE)
p4 <- ggpredict(fit, "neg_c_7") %>%
       plot(show.y.title = FALSE, show.title = FALSE)

  # plot grid
  plot_grid(list(p1, p2, p3, p4))

  # plot grid
  plot_grid(list(p1, p2, p3, p4), tags = TRUE)
}
```

---

**plot_grpfreq**

*Plot grouped or stacked frequencies*

**Description**

Plot grouped or stacked frequencies of variables as bar/dot, box or violin plots, or line plot.

**Usage**

```r
plot_grpfreq(
  var.cnt,  
  var.grp,  
  type = c("bar", "dot", "line", "boxplot", "violin"),
  bar.pos = c("dodge", "stack"),
  weight.by = NULL,
  intr.var = NULL,
  title = "",
  title.wtd.suffix = NULL,
  legend.title = NULL,
  axis.titles = NULL,
  ```
`plot_grpfrq`

```r
axis.labels = NULL,
legend.labels = NULL,
intr.var.labels = NULL,
wrap.title = 50,
wrap.labels = 15,
wrap.legend.title = 20,
wrap.legend.labels = 20,
geom.size = NULL,
geom.spacing = 0.15,
geom.colors = "Paired",
show.values = TRUE,
show.n = TRUE,
show.prc = TRUE,
show.axis.values = TRUE,
show.ci = FALSE,
show.grpcnt = FALSE,
show.legend = TRUE,
show.na = FALSE,
show.summary = FALSE,
drop.empty = TRUE,
auto.group = NULL,
ylim = NULL,
grid.breaks = NULL,
expand.grid = FALSE,
inner.box.width = 0.15,
inner.box.dotsize = 3,
smooth.lines = FALSE,
emph.dots = TRUE,
summary.pos = "r",
facet.grid = FALSE,
coord.flip = FALSE,
y.offset = NULL,
vjust = "bottom",
hjust = "center"
```

**Arguments**

- **var.cnt**: Vector of counts, for which frequencies or means will be plotted or printed.
- **var.grp**: Factor with the cross-classifying variable, where var.cnt is grouped into the categories represented by var.grp.
- **type**: Specifies the plot type. May be abbreviated.
  - "bar" for simple bars (default)
  - "dot" for a dot plot
  - "histogram" for a histogram (does not apply to grouped frequencies)
  - "line" for a line-styled histogram with filled area
  - "density" for a density plot (does not apply to grouped frequencies)
"boxplot" for box plot
"violin" for violin plots

bar.pos  Indicates whether bars should be positioned side-by-side (default), or stacked (bar.pos = "stack"). May be abbreviated.

weight.by  Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.

intr.var  An interaction variable which can be used for box plots. Divides each category indicated by var.grp into the factors of intr.var, so that each category of var.grp is subgrouped into intr.var's categories. Only applies when type = "boxplot" or type = "violin".

title  character vector, used as plot title. Depending on plot type and function, will be set automatically. If title = "", no title is printed. For effect-plots, may also be a character vector of length > 1, to define titles for each sub-plot or facet.

title.wtd.suffix  Suffix (as string) for the title, if weight.by is specified, e.g. title.wtd.suffix="(weighted)". Default is NULL, so title will not have a suffix when cases are weighted.

legend.title  character vector, used as title for the plot legend.
axis.titles  character vector of length one or two, defining the title(s) for the x-axis and y-axis.
axis.labels  character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.
legend.labels  character vector with labels for the guide/legend.
intr.var.labels  a character vector with labels for the x-axis breaks when having interaction variables included. These labels replace the axis.labels. Only applies, when using box or violin plots (i.e. type = "boxplot" or "violin") and intr.var is not NULL.

wrap.title  numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels  numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

wrap.legend.title  numeric, determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

wrap.legend.labels  numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

geom.size  size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.

geom.spacing  the spacing between geoms (i.e. bar spacing)

geom.colors  user defined color for geoms. See 'Details' in plot_grpfrq.

show.values  Logical, whether values should be plotted or not.
show.n logical, if TRUE, adds total number of cases for each group or category to the labels.

show.prc logical, if TRUE (default), percentage values are plotted to each bar If FALSE, percentage values are removed.

show.axis.values logical, whether category, count or percentage values for the axis should be printed or not.

show.ci Logical, if TRUE, adds notches to the box plot, which are used to compare groups; if the notches of two boxes do not overlap, medians are considered to be significantly different.

show.grpcnt logical, if TRUE, the count within each group is added to the category labels (e.g. "Cat 1 (n=87)"). Default value is FALSE.

show.legend logical, if TRUE, and depending on plot type and function, a legend is added to the plot.

show.na logical, if TRUE, NA's (missing values) are added to the output.

show.summary logical, if TRUE (default), a summary with chi-squared statistics (see chisq.test), Cramer's V or Phi-value etc. is shown. If a cell contains expected values lower than five (or lower than 10 if df is 1), the Fisher's exact test (see fisher.test) is computed instead of chi-squared test. If the table's matrix is larger than 2x2, Fisher's exact test with Monte Carlo simulation is computed.

drop.empty Logical, if TRUE and the variable's values are labeled, values / factor levels with no occurrence in the data are omitted from the output. If FALSE, labeled values that have no observations are still printed in the table (with frequency 0).

auto.group numeric value, indicating the minimum amount of unique values in the count variable, at which automatic grouping into smaller units is done (see group_var). Default value for auto.group is NULL, i.e. auto-grouping is off. See group_var for examples on grouping.

ylim numeric vector of length two, defining lower and upper axis limits of the y scale. By default, this argument is set to NULL, i.e. the y-axis fits to the required range of the data.

grid.breaks numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks' th position a major grid is being printed.

expand.grid logical, if TRUE, the plot grid is expanded, i.e. there is a small margin between axes and plotting region. Default is FALSE.

inner.box.width width of the inner box plot that is plotted inside of violin plots. Only applies if type = "violin". Default value is 0.15

inner.box.dotsize size of mean dot inside a violin or box plot. Applies only when type = "violin" or "boxplot".

smooth.lines prints a smooth line curve. Only applies, when argument type = "line".

emph.dots logical, if TRUE, the groups of dots in a dot-plot are highlighted with a shaded rectangle.
summary.pos position of the model summary which is printed when show.summary is TRUE. Default is "r", i.e. it's printed to the upper right corner. Use "l" for upper left corner.

facet.grid TRUE to arrange the lay out of of multiple plots in a grid of an integrated single plot. This argument calls facet_wrap or facet_grid to arrange plots. Use plot_grid to plot multiple plot-objects as an arranged grid with grid.arrange.

coord.flip logical, if TRUE, the x and y axis are swapped.

y.offset numeric, offset for text labels when their alignment is adjusted to the top/bottom of the geom (see hjust and vjust).

vjust character vector, indicating the vertical position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

hjust character vector, indicating the horizontal position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

details

geom.colors may be a character vector of color values in hex-format, valid color value names (see demo("colors")) or a name of a color brewer palette. Following options are valid for the geom.colors argument:

- If not specified, a default color brewer palette will be used, which is suitable for the plot style (i.e. diverging for likert scales, qualitative for grouped bars etc.).
- If "gs", a greyscale will be used.
- If "bw", and plot-type is a line-plot, the plot is black/white and uses different line types to distinguish groups (see this package-vignette).
- If geom.colors is any valid color brewer palette name, the related palette will be used. Use RColorBrewer::display.brewer.all() to view all available palette names.
- Else specify own color values or names as vector (e.g. geom.colors = c("#f00000", "#00ff00")).

details

value A ggplot-object.

details

examples
data(efc)
plot.grpfrq(efc$e17age, efc$e16sex, show.values = FALSE)

# boxplot
plot.grpfrq(efc$e17age, efc$e42dep, type = "box")

# grouped bars
plot.grpfrq(efc$e42dep, efc$e16sex, title = NULL)
# box plots with interaction variable
plot_grpfreq(efc$e17age, efc$e42dep, intr.var = efc$e16sex, type = "box")

# Grouped bar plot
plot_grpfreq(efc$neg_c_7, efc$e42dep, show.values = FALSE)

# same data as line plot
plot_grpfreq(efc$neg_c_7, efc$e42dep, type = "line")

# show only categories where we have data (i.e. drop zero-counts)
library(dplyr)
efc <- dplyr::filter(efc, e42dep %in% c(3,4))
plot_grpfreq(efc$c161sex, efc$e42dep, drop.empty = TRUE)

# show all categories, even if not in data
plot_grpfreq(efc$c161sex, efc$e42dep, drop.empty = FALSE)

---

plot_kfold_cv  
Plot model fit from k-fold cross-validation

Description

This function plots the aggregated residuals of k-fold cross-validated models against the outcome. This allows to evaluate how the model performs according over- or underestimation of the outcome.

Usage

plot_kfold_cv(data, formula, k = 5, fit)

Arguments

data  
A data frame, used to split the data into k training-test-pairs.

formula  
A model formula, used to fit linear models (lm) over all k training data sets. Use fit to specify a fitted model (also other models than linear models), which will be used to compute cross validation. If fit is not missing, formula will be ignored.

k  
Number of folds.

fit  
Model object, which will be used to compute cross validation. If fit is not missing, formula will be ignored. Currently, only linear, poisson and negative binomial regression models are supported.
Details

This function, first, generates k cross-validated test-training pairs and fits the same model, specified in the formula- or fit- argument, over all training data sets.

Then, the test data is used to predict the outcome from all models that have been fit on the training data, and the residuals from all test data is plotted against the observed values (outcome) from the test data (note: for poisson or negative binomial models, the deviance residuals are calculated). This plot can be used to validate the model and see, whether it over- (residuals > 0) or underestimates (residuals < 0) the model’s outcome.

Note

Currently, only linear, poisson and negative binomial regression models are supported.

Examples

data(efc)
plot_kfold_cv(efc, neg_c_7 ~ e42dep + c172code + c12hour)
plot_kfold_cv(mtcars, mpg ~ .)

# for poisson models. need to fit a model and use 'fit'-argument
fit <- glm(tot_sc_e ~ neg_c_7 + c172code, data = efc, family = poisson)
plot_kfold_cv(efc, fit = fit)

# and for negative binomial models
fit <- MASS::glm.nb(tot_sc_e ~ neg_c_7 + c172code, data = efc)
plot_kfold_cv(efc, fit = fit)

Description

Plot likert scales as centered stacked bars.

Usage

plot_likert(
  items,
  groups = NULL,
  groups.titles = "auto",
  title = NULL,
  legend.title = NULL,
  legend.labels = NULL,
  axis.titles = NULL,
  axis.labels = NULL,
)
Arguments

items      Data frame, or a grouped data frame, with each column representing one item.
groups     (optional) Must be a vector of same length as ncol(items), where each item in
            this vector represents the group number of the related columns of items. See
            'Examples'.
groups.titles (optional, only used if groups are supplied) Titles for each factor group that will
            be used as table caption for each component-table. Must be a character vector
            of same length as length(unique(groups)). Default is "auto", which means
            that each table has a standard caption *Component x*. Use NULL to use names as
            supplied to groups and use FALSE to suppress table captions.
title      character vector, used as plot title. Depending on plot type and function, will be
            set automatically. If title = "", no title is printed. For effect-plots, may also be
            a character vector of length > 1, to define titles for each sub-plot or facet.
legend.title character vector, used as title for the plot legend.
legend.labels character vector with labels for the guide/legend.
axis.titles  character vector of length one or two, defining the title(s) for the x-axis and y-axis.

axis.labels character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.

catcount optional, amount of categories of items (e.g. "strongly disagree", "disagree", "agree" and "strongly agree" would be catcount = 4). Note that this argument only applies to "valid" answers, i.e. if you have an additional neutral category (see cat.neutral) like "don't know", this won't count for catcount (e.g. "strongly disagree", "disagree", "agree", "strongly agree" and neutral category "don't know" would still mean that catcount = 4). See 'Note'.

cat.neutral If there's a neutral category (like "don't know" etc.), specify the index number (value) for this category. Else, set cat.neutral = NULL (default). The proportions of neutral category answers are plotted as grey bars on the left side of the figure.

sort.frq Indicates whether the items of items should be ordered by total sum of positive or negative answers.

"pos.asc" to order ascending by sum of positive answers
"pos.desc" to order descending by sum of positive answers
"neg.asc" for sorting ascending negative answers
"neg.desc" for sorting descending negative answers
NULL (default) for no sorting

weight.by Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.

title.wtd.suffix Suffix (as string) for the title, if weight.by is specified, e.g. title.wtd.suffix=" (weighted)". Default is NULL, so title will not have a suffix when cases are weighted.

wrap.title numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

wrap.legend.title numeric, determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

wrap.legend.labels numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

gem.size size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.

gem.colors user defined color for geoms. See 'Details' in plot.grpfrq.

cat.neutral.color Color of the neutral category, if plotted (see cat.neutral).

intercept.line.color Color of the vertical intercept line that divides positive and negative values.
reverse.colors logical, if TRUE, the color scale from geom.colors will be reversed, so positive and negative values switch colors.

values Determines style and position of percentage value labels on the bars:

"show" (default) shows percentage value labels in the middle of each category bar
"hide" hides the value labels, so no percentage values on the bars are printed
"sum.inside" shows the sums of percentage values for both negative and positive values and prints them inside the end of each bar
"sum.outside" shows the sums of percentage values for both negative and positive values and prints them outside the end of each bar

show.n logical, if TRUE, adds total number of cases for each group or category to the labels.

show.legend logical, if TRUE, and depending on plot type and function, a legend is added to the plot.

show.prc.sign logical, if TRUE, %-signs for value labels are shown.

grid.range Numeric, limits of the x-axis-range, as proportion of 100. Default is 1, so the x-scale ranges from zero to 100% on both sides from the center. Can alternatively be supplied as a vector of 2 positive numbers (e.g. grid.range = c(1, .8)) to set the left and right limit separately. You can use values beyond 1 (100%) in case bar labels are not printed because they exceed the axis range. E.g. grid.range = 1.4 will set the axis from -140 to +140%, however, only (valid) axis labels from -100 to +100% are printed. Neutral categories are adjusted to the most left limit.

grid.breaks numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks’th position a major grid is being printed.

expand.grid logical, if TRUE, the plot grid is expanded, i.e. there is a small margin between axes and plotting region. Default is FALSE.

digits Numeric, amount of digits after decimal point when rounding estimates or values.

reverse.scale logical, if TRUE, the ordering of the categories is reversed, so positive and negative values switch position.

coord.flip logical, if TRUE, the x and y axis are swapped.

sort.groups (optional, only used if groups are supplied) logical, if groups should be sorted according to the values supplied to groups. Defaults to TRUE.

legend.pos (optional, only used if groups are supplied) Defines the legend position. Possible values are c("bottom", "top", "both", "all", "none"). If the is only one group or this option is set to "all" legends will be printed as defined with set_theme.

rel_heights (optional, only used if groups are supplied) This option can be used to adjust the height of the subplots. The bars in subplots can have different heights due to a differing number of items or due to legend placement. This can be adjusted here. Takes a vector of numbers, one for each plot. Values are evaluated relative to each other.
plot_likert

- **group.legend.options**
  
  (optional, only used if groups are supplied) List of options to be passed to `guide_legend`. The most notable options are `byrow = T` (default), this will order the categories row wise. And with `group.legend.options = list(nrow = 1)` all categories can be forced to be on a single row.

- **cowplot.options**
  
  (optional, only used if groups are supplied) List of label options to be passed to `plot_grid`.

**Value**

A ggplot-object.

**Note**

Note that only even numbers of categories are possible to plot, so the "positive" and "negative" values can be splitted into two halves. A neutral category (like "don't know") can be used, but must be indicated by `cat.neutral`.

The `catcount`-argument indicates how many item categories are in the Likert scale. Normally, this argument can be ignored because the amount of valid categories is retrieved automatically. However, sometimes (for instance, if a certain category is missing in all items), auto-detection of the amount of categories fails. In such cases, specify the amount of categories with the `catcount`-argument.

**Examples**

```r
if (requireNamespace("ggrepel") && requireNamespace("sjmisc")) {
  library(sjmisc)
data(efc)
  # find all variables from COPE-Index, which all have a "cop" in their
  # variable name, and then plot that subset as likert-plot
  mydf <- find_var(efc, pattern = "cop", out = "df")
  plot_likert(mydf)

  plot_likert(
    mydf,
    grid.range = c(1.2, 1.4),
    expand.grid = FALSE,
    values = "sum.outside",
    show.prc.sign = TRUE
  )

  # Plot in groups
  plot_likert(mydf, c(2,1,1,1,2,2,2,1))
}
```

if (require("parameters") && require("nFactors")) {
  groups <- parameters::principal_components(mydf)
  plot_likert(mydf, groups = parameters::closest_component(groups))
}
plot_model()

plot_model() creates plots from regression models, either estimates (as so-called forest or dot whisker plots) or marginal effects.
plot_model

Usage

plot_model(
  model,
  type = c("est", "re", "eff", "emm", "pred", "int", "std", "std2", "slope", "resid", 
            "diag"),
  transform,
  terms = NULL,
  sort.est = NULL,
  rm.terms = NULL,
  group.terms = NULL,
  order.terms = NULL,
  pred.type = c("fe", "re"),
  mdr.t.values = c("minmax", "meansd", "zeromax", "quart", "all"),
  ri.nr = NULL,
  title = NULL,
  axis.title = NULL,
  axis.labels = NULL,
  legend.title = NULL,
  wrap.title = 50,
  wrap.labels = 25,
  axis.lim = NULL,
  grid.breaks = NULL,
  ci.lvl = NULL,
  se = NULL,
  robust = FALSE,
  vcov.fun = NULL,
  vcov.type = NULL,
  vcov.args = NULL,
  colors = "Set1",
  show.intercept = FALSE,
  show.values = FALSE,
  show.p = TRUE,
  show.data = FALSE,
  show.legend = TRUE,
  show.zeroinf = TRUE,
  value.offset = NULL,
  value.size,
  jitter = NULL,
  digits = 2,
  dot.size = NULL,
  line.size = NULL,
  vline.color = NULL,
  p.threshold = c(0.05, 0.01, 0.001),
  p.val = NULL,
  p.adjust = NULL,
  grid,
  case,
  auto.label = TRUE,
prefix.labels = c("none", "varname", "label"),
bpe = "median",
bpe.style = "line",
bpe.color = "white",
ci.style = c("whisker", "bar"),
std.response = TRUE,
...
)

get_model_data(
  model,
  type = c("est", "re", "eff", "pred", "int", "std", "std2", "slope", "resid", "diag"),
  transform,
  terms = NULL,
  sort.est = NULL,
  rm.terms = NULL,
  group.terms = NULL,
  order.terms = NULL,
  pred.type = c("fe", "re"),
  ri.nr = NULL,
  ci.lvl = NULL,
  colors = "Set1",
  grid,
  case = "parsed",
  digits = 2,
  ...
)

Arguments

- **model**: A regression model object. Depending on the *type*, many kinds of models are supported, e.g. from packages like *stats*, *lme4*, *nlme*, *rstanarm*, *survey*, *glmmTMB*, *MASS*, *brms* etc.

- **type**: Type of plot. There are three groups of plot-types:

  - **Coefficients** (related vignette)
    type = "est" Forest-plot of estimates. If the fitted model only contains one predictor, slope-line is plotted.
    type = "re" For mixed effects models, plots the random effects.
    type = "std" Forest-plot of standardized coefficients.
    type = "std2" Forest-plot of standardized coefficients, however, standardization is done by dividing by two SD (see 'Details').

  - **Marginal Effects** (related vignette)
    type = "pred" Predicted values (marginal effects) for specific model terms. See *ggpredict* for details.
    type = "eff" Similar to type = "pred", however, discrete predictors are held constant at their proportions (not reference level). See *ggeffect* for details.
plot_model

`type = "emm"` Similar to `type = "eff"`, see `ggeffects` for details.

`type = "int"` Marginal effects of interaction terms in model.

**Model diagnostics**

`type = "slope"` Slope of coefficients for each single predictor, against the response (linear relationship between each model term and response). See 'Details'.

`type = "resid"` Slope of coefficients for each single predictor, against the residuals (linear relationship between each model term and residuals). See 'Details'.

`type = "diag"` Check model assumptions. See 'Details'.

**Note:** For mixed models, the diagnostic plots like linear relationship or check for Homoscedasticity, do not take the uncertainty of random effects into account, but is only based on the fixed effects part of the model.

**transform**

A character vector, naming a function that will be applied on estimates and confidence intervals. By default, `transform` will automatically use "exp" as transformation for applicable classes of model (e.g. logistic or poisson regression). Estimates of linear models remain untransformed. Use `NULL` if you want the raw, non-transformed estimates.

**terms**

Character vector with the names of those terms from `model` that should be plotted. This argument depends on the plot-type:

**Coefficients** Select terms that should be plotted. All other term are removed from the output. Note that the term names must match the names of the model’s coefficients. For factors, this means that the variable name is suffixed with the related factor level, and each category counts as one term. E.g. `rm.terms = "t_name [2,3]"` would remove the terms "t_name2" and "t_name3" (assuming that the variable `t_name` is categorical and has at least the factor levels 2 and 3). Another example for the `iris`-dataset: `terms = "Species"` would not work, instead you would write `terms = "Species [versicolor,virginica]"` to remove these two levels, or `terms = "Speciesversicolor"` if you just want to remove the level versicolor from the plot.

**Marginal Effects** Here `terms` indicates for which terms marginal effects should be displayed. At least one term is required to calculate effects, maximum length is three terms, where the second and third term indicate the groups, i.e. predictions of first term are grouped by the levels of the second (and third) term. `terms` may also indicate higher order terms (e.g. interaction terms). Indicating levels in square brackets allows for selecting only specific groups. Term name and levels in brackets must be separated by a whitespace character, e.g. `terms = c("age", "education [1,3]")`. It is also possible to specify a range of numeric values for the predictions with a colon, for instance `terms = c("education [1,3]", "age [30:50]")`. Furthermore, it is possible to specify a function name. Values for predictions will then be transformed, e.g. `terms = "income [exp]"`. This is useful when model predictors were transformed for fitting the model and should be back-transformed to the original scale for predictions. Finally, numeric vectors for which no specific values are given, a "pretty range" is calculated, to avoid memory allocation problems for vectors with many unique values.
If a numeric vector is specified as second or third term (i.e. if this vector represents a grouping structure), representative values (see `values_at`) are chosen. If all values for a numeric vector should be used to compute predictions, you may use e.g. `terms = "age [all]"`. For more details, see `ggpredict`.

`sort.est` Determines in which way estimates are sorted in the plot:
- If `NULL` (default), no sorting is done and estimates are sorted in the same order as they appear in the model formula.
- If `TRUE`, estimates are sorted in descending order, with highest estimate at the top.
- If `sort.est = "sort.all"`, estimates are re-sorted for each coefficient (only applies if `type = "re"` and `grid = FALSE`), i.e. the estimates of the random effects for each predictor are sorted and plotted to an own plot.
- If `type = "re"`, specify a predictor’s / coefficient’s name to sort estimates according to this random effect.

`rm.terms` Character vector with names that indicate which terms should be removed from the plot. Counterpart to `terms`. `rm.terms = "t_name"` would remove the term `t_name`. Default is `NULL`, i.e. all terms are used. For factors, levels that should be removed from the plot need to be explicitly indicated in square brackets, and match the model’s coefficient names, e.g. `rm.terms = "t_name [2,3]"` would remove the terms “t_name2” and “t_name3” (assuming that the variable `t_name` was categorical and has at least the factor levels 2 and 3). Another example for the `iris` dataset would be `rm.terms = "Species [versicolor,virginica]"`. Note that the `rm.terms`-argument does not apply to Marginal Effects plots.

`group.terms` Numeric vector with group indices, to group coefficients. Each group of coefficients gets its own color (see 'Examples').

`order.terms` Numeric vector, indicating in which order the coefficients should be plotted. See examples in this package-vignette.

`pred.type` Character, only applies for Marginal Effects plots with mixed effects models. Indicates whether predicted values should be conditioned on random effects (`pred.type = "re"`) or fixed effects only (`pred.type = "fe"`, the default). For details, see documentation of the `type`-argument in `ggpredict`.

`mdrt.values` Indicates which values of the moderator variable should be used when plotting interaction terms (i.e. `type = "int"`).
- "minmax" (default) minimum and maximum values (lower and upper bounds) of the moderator are used to plot the interaction between independent variable and moderator(s).
- "meansd" uses the mean value of the moderator as well as one standard deviation below and above mean value to plot the effect of the moderator on the independent variable (following the convention suggested by Cohen and Cohen and popularized by Aiken and West (1991), i.e. using the mean, the value one standard deviation above, and the value one standard deviation below the mean as values of the moderator, see Grace-Martin K: 3 Tips to Make Interpreting Moderation Effects Easier).
- "zeromax" is similar to the "minmax" option, however, 0 is always used as minimum value for the moderator. This may be useful for predictors that
don’t have an empirical zero-value, but absence of moderation should be simulated by using 0 as minimum.

"quart" calculates and uses the quartiles (lower, median and upper) of the moderator value.

"all" uses all values of the moderator variable.

ri.nr Numeric vector. If type = "re" and fitted model has more than one random intercept, ri.nr indicates which random effects of which random intercept (or: which list elements of ranef) will be plotted. Default is NULL, so all random effects will be plotted.

title Character vector, used as plot title. By default, response_labels is called to retrieve the label of the dependent variable, which will be used as title. Use title = "" to remove title.

axis.title Character vector of length one or two (depending on the plot function and type), used as title(s) for the x and y axis. If not specified, a default labelling is chosen. Note: Some plot types may not support this argument sufficiently. In such cases, use the returned ggplot-object and add axis titles manually with labs. Use axis.title = "" to remove axis titles.

axis.labels Character vector with labels for the model terms, used as axis labels. By default, term_labels is called to retrieve the labels of the coefficients, which will be used as axis labels. Use axis.labels = "" or auto.label = FALSE to use the variable names as labels instead. If axis.labels is a named vector, axis labels (by default, the names of the model’s coefficients) will be matched with the names of axis.label. This ensures that labels always match the related axis value, no matter in which way axis labels are sorted.

legend.title Character vector, used as legend title for plots that have a legend.

wrap.title Numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

axis.lim Numeric vector of length 2, defining the range of the plot axis. Depending on plot-type, may effect either x- or y-axis. For Marginal Effects plots, axis.lim may also be a list of two vectors of length 2, defining axis limits for both the x and y axis.

grid.breaks Numeric value or vector; if grid.breaks is a single value, sets the distance between breaks for the axis at every grid.breaks’th position, where a major grid line is plotted. If grid.breaks is a vector, values will be used to define the axis positions of the major grid lines.

ci.lvl Numeric, the level of the confidence intervals (error bars). Use ci.lvl = NA to remove error bars. For stanreg-models, ci.lvl defines the (outer) probability for the credible interval that is plotted (see ci). By default, stanreg-models are printed with two intervals: the “inner” interval, which defaults to the 50%-CI, and the “outer” interval, which defaults to the 89%-CI. ci.lvl affects only the outer interval in such cases. See prob.inner and prob.outer under the ...-argument for more details.
Logical, if TRUE, the standard errors are also printed. If robust standard errors are required, use arguments vcov.fun, vcov.type and vcov.args (see standard_error for details), or use argument robust as shortcut. se overrides ci.lvl: if not NULL, arguments ci.lvl and transform will be ignored. Currently, se only applies to Coefficients plots.

Deprecated. Please use vcov.fun directly to specify the estimation of the variance-covariance matrix.

Variance-covariance matrix used to compute uncertainty estimates (e.g., for robust standard errors). This argument accepts a covariance matrix, a function which returns a covariance matrix, or a string which identifies the function to be used to compute the covariance matrix. See model_parameters() for details.

Deprecated. The type-argument is now included in vcov.args.

List of arguments to be passed to the function identified by the vcov.fun argument. This function is typically supplied by the sandwich or clubSandwich packages. Please refer to their documentation (e.g., ?sandwich::vcovHAC) to see the list of available arguments.

May be a character vector of color values in hex-format, valid color value names (see demo("colors")) or a name of a pre-defined color palette. Following options are valid for the colors argument:

• If not specified, a default color brewer palette will be used, which is suitable for the plot style.
• If "gs", a greyscale will be used.
• If "bw", and plot-type is a line-plot, the plot is black/white and uses different line types to distinguish groups (see this package-vignette).
• If colors is any valid color brewer palette name, the related palette will be used. Use RColorBrewer::display.brewer.all() to view all available palette names.
• There are some pre-defined color palettes in this package, see sjPlot-themes for details.
• Else specify own color values or names as vector (e.g. colors = "#00ff00" or colors = c("firebrick", "blue").

Logical, if TRUE, the intercept of the fitted model is also plotted. Default is FALSE. If transform = "exp", please note that due to exponential transformation of estimates, the intercept in some cases is non-finite and the plot can not be created.

Logical, whether values should be plotted or not.

Logical, adds asterisks that indicate the significance level of estimates to the value labels.

Logical, for Marginal Effects plots, also plots the raw data points.

For Marginal Effects plots, shows or hides the legend.

Logical, if TRUE, shows the zero-inflation part of hurdle- or zero-inflated models.

Numeric, offset for text labels to adjust their position relative to the dots or lines.

Numeric, indicates the size of value labels. Can be used for all plot types where the argument show.values is applicable, e.g. value.size = 4.
jitter Numeric, between 0 and 1. If show.data = TRUE, you can add a small amount of random variation to the location of each data point. jitter then indicates the width, i.e. how much of a bin’s width will be occupied by the jittered values.

digits Numeric, amount of digits after decimal point when rounding estimates or values.
dot.size Numeric, size of the dots that indicate the point estimates.
line.size Numeric, size of the lines that indicate the error bars.
vline.color Color of the vertical "zero effect" line. Default color is inherited from the current theme.
p.threshold Numeric vector of length 3, indicating the threshold for annotating p-values with asterisks. Only applies if p.style = "asterisk".
p.val Character specifying method to be used to calculate p-values. Defaults to "profile" for glm/polr models, otherwise "wald".
p.adjust Character vector, if not NULL, indicates the method to adjust p-values. See p.adjust for details.
grid Logical, if TRUE, multiple plots are plotted as grid layout.
case Desired target case. Labels will automatically converted into the specified character case. See snakecase::to_any_case() for more details on this argument. By default, if case is not specified, it will be set to "parsed", unless prefix.labels is not "none". If prefix.labels is either "label" (or "l") or "varname" (or "v") and case is not specified, it will be set to NULL - this is a more convenient default when prefixing labels.
auto.label Logical, if TRUE (the default), and data is labelled, term_labels is called to retrieve the labels of the coefficients, which will be used as predictor labels. If data is not labelled, format_parameters() is used to create pretty labels. If auto.label = FALSE, original variable names and value labels (factor levels) are used.
prefix.labels Indicates whether the value labels of categorical variables should be prefixed, e.g. with the variable name or variable label. See argument prefix in term_labels for details.
bpe For Stan-models (fitted with the rstanarm- or brms-package), the Bayesian point estimate is, by default, the median of the posterior distribution. Use bpe to define other functions to calculate the Bayesian point estimate. bpe needs to be a character naming the specific function, which is passed to the fun-argument in typical_value. So, bpe = "mean" would calculate the mean value of the posterior distribution.
bpe.style Character vector, indicating the color of the Bayesian point estimate. Setting bpe.style = NULL will inherit the color from the mapped aesthetic to match it with the geom’s color.

bpe.color Character vector, defining whether inner and outer intervals for Bayesian models are shown in boxplot-style ("whisker") or in bars with different alpha-levels ("bar").
std.response  Logical, whether the response variable will also be standardized if standardized coefficients are requested. Setting both std.response = TRUE and show.std = TRUE will behave as if the complete data was standardized before fitting the model.

Other arguments, passed down to various functions. Here is a list of supported arguments and their description in detail.

prob.inner and prob.outer For Stan-models (fitted with the rstanarm- or brms-package) and coefficients plot-types, you can specify numeric values between 0 and 1 for prob.inner and prob.outer, which will then be used as inner and outer probabilities for the uncertainty intervals (HDI). By default, the inner probability is 0.5 and the outer probability is 0.89 (unless ci.lvl is specified - in this case, ci.lvl is used as outer probability).

size.inner For Stan-models and Coefficients plot-types, you can specify the width of the bar for the inner probabilities. Default is 0.1. Setting size.inner = 0 removes the inner probability regions.

width, alpha, and scale Passed down to geom_errorbar() or geom_density_ridges(), for forest or diagnostic plots.

width, alpha, dot.alpha, dodge and log.y Passed down to plot.ggeffects for Marginal Effects plots.

show.loess Logical, for diagnostic plot-types "slope" and "resid", adds (or hides) a loess-smoothed line to the plot.

Marginal Effects plot-types When plotting marginal effects, arguments are also passed down to ggpredict, ggeffect or plot.ggeffects.

Case conversion of labels For case conversion of labels (see argument case), arguments sep.in and sep.out will be passed down to snakecase::to_any_case(). This only applies to automatically retrieved term labels, not if term labels are provided by the axis.labels-argument.

Details

Different Plot Types:

- type = "std" Plots standardized estimates. See details below.
- type = "std2" Plots standardized estimates, however, standardization follows Gelman’s (2008) suggestion, rescaling the estimates by dividing them by two standard deviations instead of just one. Resulting coefficients are then directly comparable for untransformed binary predictors.
- type = "pred" Plots estimated marginal means (or marginal effects). Simply wraps ggpredict. See also this package-vignette.
- type = "eff" Plots estimated marginal means (or marginal effects). Simply wraps ggeffect. See also this package-vignette.
- type = "int" A shortcut for marginal effects plots, where interaction terms are automatically detected and used as terms-argument. Furthermore, if the moderator variable (the second - and third - term in an interaction) is continuous, type = "int" automatically chooses useful values based on the mdrt.values-argument, which are passed to terms. Then, ggpredict is called. type = "int" plots the interaction term that appears first in the formula along the x-axis, while the second (and possibly third) variable in an interaction is used as grouping
factor(s) (moderating variable). Use type = "pred" or type = "eff" and specify a certain order in the terms-argument to indicate which variable(s) should be used as moderator. See also this package-vignette.

**type = "slope" and type = "resid"** Simple diagnostic-plots, where a linear model for each single predictor is plotted against the response variable, or the model’s residuals. Additionally, a loess-smoothed line is added to the plot. The main purpose of these plots is to check whether the relationship between outcome (or residuals) and a predictor is roughly linear or not. Since the plots are based on a simple linear regression with only one model predictor at the moment, the slopes (i.e. coefficients) may differ from the coefficients of the complete model.

**type = "diag"** For Stan-models, plots the prior versus posterior samples. For **linear (mixed) models**, plots for multicollinearity-check (Variance Inflation Factors), QQ-plots, checks for normal distribution of residuals and homoscedasticity (constant variance of residuals) are shown. For **generalized linear mixed models**, returns the QQ-plot for random effects.

**Standardized Estimates:** Default standardization is done by completely refitting the model on the standardized data. Hence, this approach is equal to standardizing the variables before fitting the model, which is particularly recommended for complex models that include interactions or transformations (e.g., polynomial or spline terms). When type = "std2", standardization of estimates follows Gelman’s (2008) suggestion, rescaling the estimates by dividing them by two standard deviations instead of just one. Resulting coefficients are then directly comparable for untransformed binary predictors.

**Value**

Depending on the plot-type, plot_model() returns a ggplot-object or a list of such objects. get_model_data returns the associated data with the plot-object as tidy data frame, or (depending on the plot-type) a list of such data frames.

**References**


**Examples**

```{r}
# prepare data
if (requireNamespace("haven")) {
  library(sjmisc)
  data(efc)
  efc <- to_factor(efc, c161sex, e42dep, c172code)
  m <- lm(neg_c_7 ~ pos_v_4 + c12hour + e42dep + c172code, data = efc)

  # simple forest plot
  plot_model(m)

  # grouped coefficients
  plot_model(m, group.terms = c(1, 2, 3, 3, 4))
```
# keep only selected terms in the model: pos_v_4, the
# levels 3 and 4 of factor e42dep and levels 2 and 3 for c172code
plot_model(m, terms = c("pos_v_4", "e42dep [3,4]", "c172code [2,3]"))
}

# multiple plots, as returned from "diagnostic"-plot type,
# can be arranged with 'plot_grid()'
## Not run:
p <- plot_model(m, type = "diag")
plot_grid(p)
## End(Not run)

# plot random effects
if (require("lme4") & require("glmmTMB")) {
  m <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
  plot_model(m, type = "re")
}

# plot marginal effects
plot_model(m, type = "pred", terms = "Days")

# plot interactions
## Not run:
m <- glm( 
tot_sc_e ~ c161sex + c172code * neg_c_7,
data = efc,
family = poisson()
)
# type = "int" automatically selects groups for continuous moderator
# variables - see argument 'mdrt.values'. The following function call is
# identical to:
# plot_model(m, type = "pred", terms = c("c172code", "neg_c_7 [7,28]"))
plot_model(m, type = "int")

# switch moderator
plot_model(m, type = "pred", terms = c("neg_c_7", "c172code"))
# same as
# ggeffects::ggpredict(m, terms = c("neg_c_7", "c172code"))
## End(Not run)

# plot Stan-model
## Not run:
if (require("rstanarm")) {
data(mtcars)
m <- stan_glm(mpg ~ wt + am + cyl + gear, data = mtcars, chains = 1)
plot_model(m, bpe.style = "dot")
}  
## End(Not run)
Description

Plot and compare regression coefficients with confidence intervals of multiple regression models in one plot.

Usage

plot_models(
    ..., 
    transform = NULL, 
    std.est = NULL, 
    std.response = TRUE, 
    rm.terms = NULL, 
    title = NULL, 
    m.labels = NULL, 
    legend.title = "Dependent Variables", 
    legend.pval.title = "p-level", 
    axis.labels = NULL, 
    axis.title = NULL, 
    axis.lim = NULL, 
    wrap.title = 50, 
    wrap.labels = 25, 
    wrap.legend.title = 20, 
    grid.breaks = NULL, 
    dot.size = 3, 
    line.size = NULL, 
    value.size = NULL, 
    spacing = 0.4, 
    colors = "Set1", 
    show.values = FALSE, 
    show.legend = TRUE, 
    show.intercept = FALSE, 
    show.p = TRUE, 
    p.shape = FALSE, 
    p.threshold = c(0.05, 0.01, 0.001), 
    p.adjust = NULL, 
    ci.lvl = 0.95, 
    robust = FALSE, 
    vcov.fun = NULL, 
    vcov.type = c("HC3", "const", "HC", "HC0", "HC1", "HC2", "HC4", "HC4m", "HC5"), 
    vcov.args = NULL, 
    vline.color = NULL, 
    digits = 2, 
    grid = FALSE, 
    auto.label = TRUE, 
    prefix.labels = c("none", "varname", "label")
)
plot_models

Arguments

... One or more regression models, including glm’s or mixed models. May also be a list with fitted models. See 'Examples'.

transform A character vector, naming a function that will be applied on estimates and confidence intervals. By default, transform will automatically use "exp" as transformation for applicable classes of model (e.g. logistic or poisson regression). Estimates of linear models remain untransformed. Use NULL if you want the raw, non-transformed estimates.

std.est Choose whether standardized coefficients should be used for plotting. Default is no standardization (std.est = NULL). May be "std" for standardized beta values or "std2", where standardization is done by rescaling estimates by dividing them by two sd.

std.response Logical, whether the response variable will also be standardized if standardized coefficients are requested. Setting both std.response = TRUE and show.std = TRUE will behave as if the complete data was standardized before fitting the model.

rm.terms Character vector with names that indicate which terms should be removed from the plot. Counterpart to terms. rm.terms = "t_name" would remove the term t_name. Default is NULL, i.e. all terms are used. For factors, levels that should be removed from the plot need to be explicitly indicated in square brackets, and match the model’s coefficient names, e.g. rm.terms = "t_name [2,3]" would remove the terms "t_name2" and "t_name3" (assuming that the variable t_name was categorical and has at least the factor levels 2 and 3). Another example for the iris dataset would be rm.terms = "Species [versicolor,virginica]". Note that the rm.terms-argument does not apply to Marginal Effects plots.

title Character vector, used as plot title. By default, response_labels is called to retrieve the label of the dependent variable, which will be used as title. Use title = "" to remove title.

m.labels Character vector, used to indicate the different models in the plot’s legend. If not specified, the labels of the dependent variables for each model are used.

legend.title Character vector, used as legend title for plots that have a legend.

legend.pval.title Character vector, used as title of the plot legend that indicates the p-values. Default is "p-level". Only applies if p.shape = TRUE.

axis.labels Character vector with labels for the model terms, used as axis labels. By default, term_labels is called to retrieve the labels of the coefficients, which will be used as axis labels. Use axis.labels = "" or auto.label = FALSE to use the variable names as labels instead. If axis.labels is a named vector, axis labels (by default, the names of the model’s coefficients) will be matched with the names of axis.label. This ensures that labels always match the related axis value, no matter in which way axis labels are sorted.

axis.title Character vector of length one or two (depending on the plot function and type), used as title(s) for the x and y axis. If not specified, a default labelling is chosen. Note: Some plot types may not support this argument sufficiently. In such cases, use the returned ggplot-object and add axis titles manually with labs. Use axis.title = "" to remove axis titles.
axis.lim Numeric vector of length 2, defining the range of the plot axis. Depending on plot-type, may effect either x- or y-axis. For Marginal Effects plots, axis.lim may also be a list of two vectors of length 2, defining axis limits for both the x and y axis.

wrap.title Numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

wrap.legend.title Numeric, determines how many chars of the legend’s title are displayed in one line and when a line break is inserted.

grid.breaks Numeric value or vector; if grid.breaks is a single value, sets the distance between breaks for the axis at every grid.breaks’th position, where a major grid line is plotted. If grid.breaks is a vector, values will be used to define the axis positions of the major grid lines.

dot.size Numeric, size of the dots that indicate the point estimates.

line.size Numeric, size of the lines that indicate the error bars.

value.size Numeric, indicates the size of value labels. Can be used for all plot types where the argument show.values is applicable, e.g. value.size = 4.

spacing Numeric, spacing between the dots and error bars of the plotted fitted models. Default is 0.3.

colors May be a character vector of color values in hex-format, valid color value names (see demo("colors")) or a name of a pre-defined color palette. Following options are valid for the colors argument:

- If not specified, a default color brewer palette will be used, which is suitable for the plot style.
- If "gs", a greyscale will be used.
- If "bw", and plot-type is a line-plot, the plot is black/white and uses different line types to distinguish groups (see this package-vignette).
- If colors is any valid color brewer palette name, the related palette will be used. Use RColorBrewer::display.brewer.all() to view all available palette names.
- There are some pre-defined color palettes in this package, see sjPlot-themes for details.
- Else specify own color values or names as vector (e.g. colors = "#00ff00" or colors = c("firebrick", "blue")).

show.values Logical, whether values should be plotted or not.

show.legend For Marginal Effects plots, shows or hides the legend.

show.intercept Logical, if TRUE, the intercept of the fitted model is also plotted. Default is FALSE. If transform = "exp", please note that due to exponential transformation of estimates, the intercept in some cases is non-finite and the plot can not be created.

show.p Logical, adds asterisks that indicate the significance level of estimates to the value labels.
p.shape Logical, if TRUE, significant levels are distinguished by different point shapes and a related legend is plotted. Default is FALSE.

p.threshold Numeric vector of length 3, indicating the threshold for annotating p-values with asterisks. Only applies if p.style = "asterisk".

p.adjust Character vector, if not NULL, indicates the method to adjust p-values. See p.adjust for details.

ci.lvl Numeric, the level of the confidence intervals (error bars). Use ci.lvl = NA to remove error bars. For stanreg-models, ci.lvl defines the (outer) probability for the credible interval that is plotted (see ci). By default, stanreg-models are printed with two intervals: the "inner" interval, which defaults to the 50%-CI; and the "outer" interval, which defaults to the 89%-CI. ci.lvl affects only the outer interval in such cases. See prob.inner and prob.outer under the ...-argument for more details.

robust Deprecated. Please use vcov.fun directly to specify the estimation of the variance-covariance matrix.

vcov.fun Variance-covariance matrix used to compute uncertainty estimates (e.g., for robust standard errors). This argument accepts a covariance matrix, a function which returns a covariance matrix, or a string which identifies the function to be used to compute the covariance matrix. See model_parameters().

vcov.type Deprecated. The type-argument is now included in vcov.args.

vcov.args List of arguments to be passed to the function identified by the vcov.fun argument. This function is typically supplied by the sandwich or clubSandwich packages. Please refer to their documentation (e.g., ?sandwich::vcovHAC) to see the list of available arguments.

vline.color Color of the vertical "zero effect" line. Default color is inherited from the current theme.

digits Numeric, amount of digits after decimal point when rounding estimates or values.

grid Logical, if TRUE, multiple plots are plotted as grid layout.

auto.label Logical, if TRUE (the default), and data is labelled, term_labels is called to retrieve the labels of the coefficients, which will be used as predictor labels. If data is not labelled, format_parameters() is used to create pretty labels. If auto.label = FALSE, original variable names and value labels (factor levels) are used.

prefix.labels Indicates whether the value labels of categorical variables should be prefixed, e.g. with the variable name or variable label. See argument prefix in term_labels for details.

Value

A ggplot-object.

Examples

data(efc)
```r
# fit three models
fit1 <- lm(barthtot ~ c160age + c12hour + c161sex + c172code, data = efc)
fit2 <- lm(neg_c_7 ~ c160age + c12hour + c161sex + c172code, data = efc)
fit3 <- lm(tot_sc_e ~ c160age + c12hour + c161sex + c172code, data = efc)

# plot multiple models
plot_models(fit1, fit2, fit3, grid = TRUE)

# plot multiple models with legend labels and
# point shapes instead of value labels
plot_models(
  fit1, fit2, fit3,
  axis.labels = c(
    "Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"
  ),
  m.labels = c("Barthel Index", "Negative Impact", "Services used"),
  show.values = FALSE, show.p = FALSE, p.shape = TRUE
)

## Not run:
# plot multiple models from nested lists argument
all.models <- list()
all.models[[1]] <- fit1
all.models[[2]] <- fit2
all.models[[3]] <- fit3

plot_models(all.models)

# plot multiple models with different predictors (stepwise inclusion),
# standardized estimates
fit1 <- lm(mpg ~ wt + cyl + disp + gear, data = mtcars)
fit2 <- update(fit1, . ~ . + hp)
fit3 <- update(fit2, . ~ . + am)

plot_models(fit1, fit2, fit3, std.est = "std2")

## End(Not run)
```

---

**plot_residuals**  
*Plot predicted values and their residuals*

**Description**

This function plots observed and predicted values of the response of linear (mixed) models for each coefficient and highlights the observed values according to their distance (residuals) to the predicted values. This allows to investigate how well actual and predicted values of the outcome fit across the predictor variables.
plot_residuals

Usage

plot_residuals(
  fit,
  geom.size = 2,
  remove.estimates = NULL,
  show.lines = TRUE,
  show.resid = TRUE,
  show.pred = TRUE,
  show.ci = FALSE
)

Arguments

fit  Fitted linear (mixed) regression model (including objects of class gls or plm).
geom.size  size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.
remove.estimates  Numeric vector with indices (order equals to row index of coef(fit)) or character vector with coefficient names that indicate which estimates should be removed from the table output. The first estimate is the intercept, followed by the model predictors. \textit{The intercept cannot be removed from the table output!} remove.estimates = c(2:4) would remove the 2nd to the 4th estimate (1st to 3rd predictor after intercept) from the output. remove.estimates = "est_name" would remove the estimate est_name. Default is NULL, i.e. all estimates are printed.
show.lines  Logical, if TRUE, a line connecting predicted and residual values is plotted. Set this argument to FALSE, if plot-building is too time consuming.
show.resid  Logical, if TRUE, residual values are plotted.
show.pred  Logical, if TRUE, predicted values are plotted.
show.ci  Logical, if TRUE), adds notches to the box plot, which are used to compare groups; if the notches of two boxes do not overlap, medians are considered to be significantly different.

Value

A ggplot-object.

Note

The actual (observed) values have a coloured fill, while the predicted values have a solid outline without filling.

Examples

data(efc)
# fit model
plot_scatter

fit <- lm(neg_c_7 ~ c12hour + e17age + e42dep, data = efc)

# plot residuals for all independent variables
plot_residuals(fit)

# remove some independent variables from output
plot_residuals(fit, remove.estimates = c("e17age", "e42dep"))

---

plot_scatter  
*Plot (grouped) scatter plots*

**Description**

Display scatter plot of two variables. Adding a grouping variable to the scatter plot is possible. Furthermore, fitted lines can be added for each group as well as for the overall plot.

**Usage**

```r
plot_scatter(
  data,
  x,
  y,
  grp,
  title = "",
  legend.title = NULL,
  legend.labels = NULL,
  dot.labels = NULL,
  axis.titles = NULL,
  dot.size = 1.5,
  label.size = 3,
  colors = "metro",
  fit.line = NULL,
  fit.grps = NULL,
  show.rug = FALSE,
  show.legend = TRUE,
  show.ci = FALSE,
  wrap.title = 50,
  wrap.legend.title = 20,
  wrap.legend.labels = 20,
  jitter = 0.05,
  emph.dots = FALSE,
  grid = FALSE
)
```
Arguments

**data**
A data frame, or a grouped data frame.

**x**
Name of the variable for the x-axis.

**y**
Name of the variable for the y-axis.

**grp**
Optional, name of the grouping-variable. If not missing, the scatter plot will be grouped. See 'Examples'.

**title**
Character vector, used as plot title. By default, `response_labels` is called to retrieve the label of the dependent variable, which will be used as title. Use `title = ""` to remove title.

**legend.title**
Character vector, used as legend title for plots that have a legend.

**legend.labels**
Character vector with labels for the guide/legend.

**dot.labels**
Character vector with names for each coordinate pair given by x and y, so text labels are added to the plot. Must be of same length as x and y. If `dot.labels` has a different length, data points will be trimmed to match `dot.labels`. If `dot.labels = NULL` (default), no labels are printed.

**axis.titles**
character vector of length one or two, defining the title(s) for the x-axis and y-axis.

**dot.size**
Numeric, size of the dots that indicate the point estimates.

**label.size**
Size of text labels if argument `dot.labels` is used.

**colors**
May be a character vector of color values in hex-format, valid color value names (see `demo("colors")`) or a name of a pre-defined color palette. Following options are valid for the `colors` argument:

- If not specified, a default color brewer palette will be used, which is suitable for the plot style.
- If "gs", a greyscale will be used.
- If "bw", and plot-type is a line-plot, the plot is black/white and uses different line types to distinguish groups (see this package-vignette).
- If `colors` is any valid color brewer palette name, the related palette will be used. Use `RColorBrewer::display.brewer.all()` to view all available palette names.
- There are some pre-defined color palettes in this package, see `sjPlot-themes` for details.
- Else specify own color values or names as vector (e.g. `colors = "#00ff00"` or `colors = c("firebrick", "blue")`).

**fit.line, fit.grps**
Specifies the method to add a fitted line across the data points. Possible values are for instance "lm", "glm", "loess" or "auto". If NULL, no line is plotted. `fit.line` adds a fitted line for the complete data, while `fit.grps` adds a fitted line for each subgroup of `grp`.

**show.rug**
Logical, if TRUE, a marginal rug plot is displayed in the graph.

**show.legend**
For Marginal Effects plots, shows or hides the legend.
show.ci Logical, if TRUE, adds notches to the box plot, which are used to compare groups; if the notches of two boxes do not overlap, medians are considered to be significantly different.

wrap.title Numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.legend.title numeric, determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

wrap.legend.labels numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

jitter Numeric, between 0 and 1. If show.data = TRUE, you can add a small amount of random variation to the location of each data point. jitter then indicates the width, i.e. how much of a bin's width will be occupied by the jittered values.

emph.dots Logical, if TRUE, overlapping points at same coordinates will be become larger, so point size indicates amount of overlapping.

grid Logical, if TRUE, multiple plots are plotted as grid layout.

Value A ggplot-object. For grouped data frames, a list of ggplot-objects for each group in the data.

Examples

# load sample data
library(sjmisc)
library(sjlabelled)
data(efc)

# simple scatter plot
plot_scatter(efc, e16sex, neg_c_7)

# simple scatter plot, increased jittering
plot_scatter(efc, e16sex, neg_c_7, jitter = .4)

# grouped scatter plot
plot_scatter(efc, c160age, e17age, e42dep)

# grouped scatter plot with marginal rug plot
# and add fitted line for complete data
plot_scatter(
  efc, c12hour, c160age, c172code,
  show.rug = TRUE, fit.line = "lm"
)

# grouped scatter plot with marginal rug plot
# and add fitted line for each group
plot_scatter(
  efc, c12hour, c160age, c172code,
  show.rug = TRUE, fit.grps = "loess",
)
plot_stackfrq

Data frame, or a grouped data frame, with each column representing one item.

**title**

character vector, used as plot title. Depending on plot type and function, will be set automatically. If title = "", no title is printed. For effect-plots, may also be a character vector of length > 1, to define titles for each sub-plot or facet.
legend.title character vector, used as title for the plot legend.
legend.labels character vector with labels for the guide/legend.
axis.titles character vector of length one or two, defining the title(s) for the x-axis and y-axis.
axis.labels character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.
weight.by Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.
sort.frq Indicates whether the items should be ordered by by highest count of first or last category of items.
  "first.asc" to order ascending by lowest count of first category,
  "first.desc" to order descending by lowest count of first category,
  "last.asc" to order ascending by lowest count of last category,
  "last.desc" to order descending by lowest count of last category,
  NULL (default) for no sorting.
wrap.title numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.
wrap.labels numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.
wrap.legend.title numeric, determines how many chars of the legend’s title are displayed in one line and when a line break is inserted.
wrap.legend.labels numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.
geom.size size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.
geom.colors user defined color for geoms. See ‘Details’ in plot_grpfrq.
show.prc Logical, whether percentage values should be plotted or not.
show.n Logical, whether count values hould be plotted or not.
show.total logical, if TRUE, adds total number of cases for each group or category to the labels.
show.axis.prc Logical, if TRUE (default), the percentage values at the x-axis are shown.
show.legend logical, if TRUE, and depending on plot type and function, a legend is added to the plot.
grid.breaks numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks’th position a major grid is being printed.
expand.grid logical, if TRUE, the plot grid is expanded, i.e. there is a small margin between axes and plotting region. Default is FALSE.
digits Numeric, amount of digits after decimal point when rounding estimates or values.
vjust  character vector, indicating the vertical position of value labels. Allowed are same values as for `vjust` aesthetics from `ggplot2`: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

coord.flip  logical, if TRUE, the x and y axis are swapped.

Value

A ggplot-object.

Examples

```r
# Data from the EUROFAMCARE sample dataset
library(sjmisc)
data(efc)
# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive first item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")
# auto-detection of labels
plot_stackfrq(efc[, start:end])

# works on grouped data frames as well
library(dplyr)
efc %>%
  group_by(c161sex) %>%
  select(start:end) %>%
  plot_stackfrq()
```

## plot_xtab

### Plot contingency tables

<table>
<thead>
<tr>
<th>Description</th>
<th>Plot proportional crosstables (contingency tables) of two variables as <code>ggplot</code> diagram.</th>
</tr>
</thead>
</table>
| Usage       | `plot_xtab(x, grp, type = c("bar", "line"), margin = c("col", "cell", "row"),
|             | bar.pos = c("dodge", "stack"), title = "", title.wtd.suffix = NULL,
|             | axis.titles = NULL, ...)`                                                                  |
Arguments

x  A vector of values (variable) describing the bars which make up the plot.
grp Grouping variable of same length as x, where x is grouped into the categories represented by grp.
type Plot type. may be either "bar" (default) for bar charts, or "line" for line diagram.
margin Indicates which data of the proportional table should be plotted. Use "row" for calculating row percentages, "col" for column percentages and "cell" for cell percentages. If margin = "col", an additional bar with the total sum of each column can be added to the plot (see show.total).
bar.pos Indicates whether bars should be positioned side-by-side (default), or stacked (bar.pos = "stack"). May be abbreviated.
title character vector, used as plot title. Depending on plot type and function, will be set automatically. If title = "", no title is printed. For effect-plots, may also be a character vector of length > 1, to define titles for each sub-plot or facet.
title.wtd.suffix
Suffix (as string) for the title, if weight.by is specified, e.g. title.wtd.suffix="(weighted)". Default is NULL, so title will not have a suffix when cases are weighted.

axis.titles
character vector of length one or two, defining the title(s) for the x-axis and y-axis.

axis.labels
character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.

legend.title
character vector, used as title for the plot legend.

legend.labels
character vector with labels for the guide/legend.

weight.by
Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.

rev.order
Logical, if TRUE, order of categories (groups) is reversed.

show.values
Logical, whether values should be plotted or not.

show.n
logical, if TRUE, adds total number of cases for each group or category to the labels.

show.prc
logical, if TRUE (default), percentage values are plotted to each bar. If FALSE, percentage values are removed.

show.total
When margin = "col", an additional bar with the sum within each category and it's percentages will be added to each category.

show.legend
logical, if TRUE, and depending on plot type and function, a legend is added to the plot.

show.summary
logical, if TRUE (default), a summary with chi-squared statistics (see chisq.test), Cramer's V or Phi-value etc. is shown. If a cell contains expected values lower than five (or lower than 10 if df is 1), the Fisher's exact test (see fisher.test) is computed instead of chi-squared test. If the table's matrix is larger than 2x2, Fisher's exact test with Monte Carlo simulation is computed.

summary.pos
position of the model summary which is printed when show.summary is TRUE. Default is "r", i.e. it's printed to the upper right corner. Use "l" for upper left corner.

drop.empty
Logical, if TRUE and the variable's values are labeled, values / factor levels with no occurrence in the data are omitted from the output. If FALSE, labeled values that have no observations are still printed in the table (with frequency 0).

string.total
String for the legend label when a total-column is added. Only applies if show.total = TRUE. Default is "Total".

wrap.title
numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels
numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

wrap.legend.title
numeric, determines how many chars of the legend's title are displayed in one line and when a line break is inserted.
wrap.legend.labels
numeric, determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

geom.size
size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.

geom.spacing
the spacing between geoms (i.e. bar spacing)

dot.size
Dot size, only applies, when argument type = "line".

smooth.lines
prints a smooth line curve. Only applies, when argument type = "line".

grid.breaks
numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks’th position a major grid is being printed.

expand.grid
logical, if TRUE, the plot grid is expanded, i.e. there is a small margin between axes and plotting region. Default is FALSE.

ylim
numeric vector of length two, defining lower and upper axis limits of the y scale. By default, this argument is set to NULL, i.e. the y-axis fits to the required range of the data.

vjust
character vector, indicating the vertical position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

hjust
character vector, indicating the horizontal position of value labels. Allowed are same values as for vjust aesthetics from ggplot2: "left", "center", "right", "bottom", "middle", "top" and new options like "inward" and "outward", which align text towards and away from the center of the plot respectively.

y.offset
numeric, offset for text labels when their alignment is adjusted to the top/bottom of the geom (see hjust and vjust).

coord.flip
logical, if TRUE, the x and y axis are swapped.

Value
A ggplot-object.

Examples
# create 4-category-items
grp <- sample(1:4, 100, replace = TRUE)
# create 3-category-items
x <- sample(1:3, 100, replace = TRUE)

# plot "cross tablulation" of x and grp
plot_xtab(x, grp)

# plot "cross tablulation" of x and y, including labels
plot_xtab(x, grp, axis.labels = c("low", "mid", "high"),
legend.labels = c("Grp 1", "Grp 2", "Grp 3", "Grp 4"))
# plot "cross tabulation" of x and grp
# as stacked proportional bars
plot_xtab(x, grp, margin = "row", bar.pos = "stack",
          show.summary = TRUE, coord.flip = TRUE)

# example with vertical labels
library(sjmisc)
library(sjlabelled)
data(efc)
set_theme(geom.label.angle = 90)
plot_xtab(efc$e42dep, efc$e16sex, vjust = "center", hjust = "bottom")

# grouped bars with EUROFAMCARE sample dataset
# dataset was importet from an SPSS-file,
# see ?sjmisc::read_spss
data(efc)
efc.val <- get_labels(efc)
efc.var <- get_label(efc)
plot_xtab(efc$e42dep, efc$e16sex, title = efc.var[['Var e42dep']],
          axis.labels = efc.val[['Var e42dep']],
          legend.title = efc.var[['Var e16sex']],
          legend.labels = efc.val[['Var e16sex']])
plot_xtab(efc$e16sex, efc$e42dep, title = efc.var[['Var e16sex']],
          axis.labels = efc.val[['Var e16sex']],
          legend.title = efc.var[['Var e42dep']],
          legend.labels = efc.val[['Var e42dep']])

# auto-detection of labels works here
# so no need to specify labels. For
# title-auto-detection, use NULL
# -------------------------------------
plot_xtab(efc$e16sex, efc$e42dep, title = NULL)
plot_xtab(efc$e16sex, efc$e42dep, margin = "row",
          bar.pos = "stack", coord.flip = TRUE)

---------------------------
save_plot
---------------------------

Description

Convenient function to save the last ggplot-figure in high quality for publication.

Usage

save_plot(
  filename,
\begin{verbatim}
fig = last_plot(),
width = 12,
height = 9,
dpi = 300,
theme = theme_get(),
label.color = "black",
label.size = 2.4,
axis.textsize = 0.8,
axis.titlesize = 0.75,
legend.textsize = 0.6,
legend.titlesize = 0.65,
legend.itemsize = 0.5
\)
\end{verbatim}

**Arguments**

- **filename**
  - Name of the output file; filename must end with one of the following accepted file types: ".png", ".jpg", ".svg" or ".tif".
- **fig**
  - The plot that should be saved. By default, the last plot is saved.
- **width**
  - Width of the figure, in centimetres.
- **height**
  - Height of the figure, in centimetres.
- **dpi**
  - Resolution in dpi (dots per inch). Ignored for vector formats, such as ".svg".
- **theme**
  - The default theme to use when saving the plot.
- **label.color**
  - Color value for labels (axis, plot, etc.).
- **label.size**
  - Fontsize of value labels inside plot area.
- **axis.textsize**
  - Fontsize of axis labels.
- **axis.titlesize**
  - Fontsize of axis titles.
- **legend.textsize**
  - Fontsize of legend labels.
- **legend.titlesize**
  - Fontsize of legend title.
- **legend.itemsize**
  - Size of legend’s item (legend key), in centimetres.

**Note**

This is a convenient function with some default settings that should come close to most of the needs for fontsize and scaling in figures when saving them for printing or publishing. It uses cairographics anti-aliasing (see \texttt{png}).

For adjusting plot appearance, see also \texttt{sjPlot-themes}.
set_theme

Set global theme options for sjp-functions.

Description

Set global theme options for sjp-functions.

Usage

set_theme(
  base = theme_grey(),
  theme.font = NULL,
  title.color = "black",
  title.size = 1.2,
  title.align = "left",
  title.vjust = NULL,
  geom.outline.color = NULL,
  geom.outline.size = 0,
  geom.boxoutline.size = 0.5,
  geom.boxoutline.color = "black",
  geom.alpha = 1,
  geom.linetype = 1,
  geom.errorbar.size = 0.7,
  geom.errorbar.linetype = 1,
  geom.label.color = NULL,
  geom.label.size = 4,
  geom.label.alpha = 1,
  geom.label.angle = 0,
  axis.title.color = "grey30",
  axis.title.size = 1.1,
  axis.title.x.vjust = NULL,
  axis.title.y.vjust = NULL,
  axis.angle.x = 0,
  axis.angle.y = 0,
  axis.textcolor.x = "grey30",
  axis.textcolor.y = "grey30",
  axis.textcolor = NULL,
  axis.linecolor.x = NULL,
  axis.linecolor.y = NULL,
  axis.linecolor = NULL,
  axis.line.size = 0.5,
  axis.textsize.x = 1,
  axis.textsize.y = 1,
  axis.textsize = NULL,
  axis.tickslen = NULL,
  axis.tickscol = NULL,
set_theme

axis.ticksmar = NULL,
axis.ticksize.x = NULL,
axis.ticksize.y = NULL,
panel.backcol = NULL,
panel.bordercol = NULL,
panel.col = NULL,
panel.major.gridcol = NULL,
panel.minor.gridcol = NULL,
panel.gridcol = NULL,
panel.gridcol.x = NULL,
panel.gridcol.y = NULL,
panel.major.linetype = 1,
panel.minor.linetype = 1,
plot.backcol = NULL,
plot.bordercol = NULL,
plot.col = NULL,
plot.margins = NULL,
legend.pos = "right",
legend.just = NULL,
legend.inside = FALSE,
legend.size = 1,
legend.color = "black",
legend.title.size = 1,
legend.title.color = "black",
legend.title.face = "bold",
legend.backgroundcol = "white",
legend.bordercol = "white",
legend.item.size = NULL,
legend.item.backcol = "grey90",
legend.item.bordercol = "white"
)

Arguments

base

base theme where theme is built on. By default, all metrics from theme_gray() are used. See 'Details'.

theme.font

base font family for the plot.

title.color

Color of plot title. Default is "black".

title.size

size of plot title. Default is 1.3.

title.align

alignment of plot title. Must be one of "left" (default), "center" or "right". You may use initial letter only.

title.vjust

numeric, vertical adjustment for plot title.

geom.outline.color

Color of geom outline. Only applies, if geom.outline.size is larger than 0.

geom.outline.size

size of bar outlines. Default is 0.1. Use size of 0 to remove geom outline.
geom.boxoutline.size
  size of outlines and median bar especially for boxplots. Default is 0.5. Use size of 0 to remove boxplot outline.

geom.boxoutline.color
  Color of outlines and median bar especially for boxplots. Only applies, if geom.boxoutline.size is larger than 0.

geom.alpha
  specifies the transparency (alpha value) of geoms

geom.linetype
  linetype of line geoms. Default is 1 (solid line).

geom.errorbar.size
  size (thickness) of error bars. Default is 0.8

geom.errorbar.linetype
  linetype of error bars. Default is 1 (solid line).

geom.label.color
  Color of geom’s value and annotation labels

geom.label.size
  size of geom’s value and annotation labels

geom.label.alpha
  alpha level of geom’s value and annotation labels

geom.label.angle
  angle of geom’s value and annotation labels

axis.title.color
  Color of x- and y-axis title labels

axis.title.size
  size of x- and y-axis title labels

axis.title.x.vjust
  numeric, vertical adjustment of x-axis-title.

axis.title.y.vjust
  numeric, vertical adjustment of y-axis-title.

axis.angle.x
  angle for x-axis labels

axis.angle.y
  angle for y-axis labels

axis.angle
  angle for x- and y-axis labels. If set, overrides both axis.angle.x and axis.angle.y

axis.textcolor.x
  Color for x-axis labels. If not specified, a default dark gray color palette will be used for the labels.

axis.textcolor.y
  Color for y-axis labels. If not specified, a default dark gray color palette will be used for the labels.

axis.textcolor
  Color for both x- and y-axis labels. If set, overrides both axis.textcolor.x and axis.textcolor.y

axis.linecolor.x
  Color of x-axis border

axis.linecolor.y
  Color of y-axis border
axis.linecolor  Color for both x- and y-axis borders. If set, overrides both axis.linecolor.x and axis.linecolor.y.
axis.line.size  size (thickness) of axis lines. Only affected, if axis.linecolor is set.
axis.textsize.x  size of x-axis labels
axis.textsize.y  size of y-axis labels
axis.textsize  size for both x- and y-axis labels. If set, overrides both axis.textsize.x and axis.textsize.y.
axis.tickslen  length of axis tick marks
axis.tickscolor  Color of axis tick marks
axis.ticksámargin  margin between axis labels and tick marks
axis.ticksize.x  size of tick marks at x-axis.
axis.ticksize.y  size of tick marks at y-axis.
panel.backcolor  Color of the diagram’s background
panel.bordercolor  Color of whole diagram border (panel border)
panel.color  Color of both diagram’s border and background. If set, overrides both panel.bordercolor and panel.backcolor.
panel.major.gridcolor  Color of the major grid lines of the diagram background
panel.minor.gridcolor  Color of the minor grid lines of the diagram background
panel.gridcolor  Color for both minor and major grid lines of the diagram background. If set, overrides both panel.major.gridcolor and panel.minor.gridcolor.
panel.gridcolor.x  See panel.gridcolor.
panel.gridcolor.y  See panel.gridcolor.
panel.major.linetype  line type for major grid lines
panel.minor.linetype  line type for minor grid lines
plot.backcolor  Color of the plot’s background
plot.bordercolor  Color of whole plot’s border (panel border)
plot.color  Color of both plot’s region border and background. If set, overrides both plot.backcolor and plot.bordercolor.
plot.margins  numeric vector of length 4, indicating the top, right, bottom and left margin of the plot region.
legend.pos  position of the legend, if a legend is drawn.
**legend outside plot**  Use "bottom", "top", "left" or "right" to position the legend above, below, on the left or right side of the diagram. Right positioning is default.

**legend inside plot**  If `legend.inside = TRUE`, legend can be placed inside plot. Use "top left", "top right", "bottom left" and "bottom right" to position legend in any of these corners, or a two-element numeric vector with values from 0-1. See also `legend.inside`.

- **legend.just**  justification of legend, relative to its position ("center" or two-element numeric vector with values from 0-1. By default (outside legend), justification is centered. If legend is inside and justification not specified, legend justification is set according to legend position.

- **legend.inside**  logical, use TRUE to put legend inside the plotting area. See `legend.pos`.

- **legend.size**  text size of the legend. Default is 1. Relative size, so recommended values are from 0.3 to 2.5.

- **legend.color**  Color of the legend labels

- **legend.title.size**  text size of the legend title

- **legend.title.color**  Color of the legend title

- **legend.title.face**  font face of the legend title. By default, "bold" face is used.

- **legend.backgroundcol**  fill color of the legend’s background. Default is "white", so no visible background is drawn.

- **legend.bordercol**  Color of the legend’s border. Default is "white", so no visible border is drawn.

- **legend.item.size**  size of legend’s item (legend key), in centimetres.

- **legend.item.backcol**  fill color of the legend’s item-background. Default is "grey90".

- **legend.item.bordercol**  Color of the legend’s item-border. Default is "white".

**Value**

The customized theme object, or NULL, if a ggplot-theme was used.

**See Also**

- `sjPlot-themes`

**Examples**

```r
## Not run:
library(sjmisc)
data(efc)
```
# set sjPlot-defaults, a slightly modification
# of the ggplot base theme
set_theme()

# legends of all plots inside
set_theme(legend.pos = "top left", legend.inside = TRUE)
plot_xtab(efc$e42dep, efc$e16sex)

# Use classic-theme. you may need to
# load the ggplot2-library.
library(ggplot2)
set_theme(base = theme_classic())
plot_frq(efc$e42dep)

# adjust value labels
set_theme(
    geom.label.size = 3.5,
    geom.label.color = "#3366cc",
    geom.label.angle = 90
)

# hjust-aes needs adjustment for this
update_geom_defaults('text', list(hjust = -0.1))
plot_xtab(efc$e42dep, efc$e16sex, vjust = "center", hjust = "center")

# Create own theme based on classic-theme
set_theme(
    base = theme_classic(), axis.linecolor = "grey50",
    axis.textcolor = "#6699cc"
)
plot_frq(efc$e42dep)
## End(Not run)

---

**sjp.aov1**

*Plot One-Way-Anova tables*

**Description**

Plot One-Way-Anova table sum of squares (SS) of each factor level (group) against the dependent variable. The SS of the factor variable against the dependent variable (variance within and between groups) is printed to the model summary.

**Usage**

sjp.aov1(
    var.dep,
    var.grp,
    meansums = FALSE,
    title = NULL,
)
axis.labels = NULL,
rev.order = FALSE,
string.interc = "(Intercept)",
axis.title = "",
axis.lim = NULL,
geom.colors = c("#3366a0", "#aa3333"),
geom.size = 3,
wrap.title = 50,
wrap.labels = 25,
grid.breaks = NULL,
show.values = TRUE,
digits = 2,
y.offset = 0.15,
show.p = TRUE,
show.summary = FALSE
)

Arguments

var.dep: Dependent variable. Will be used with following formula: aov(var.dep ~ var.grp)

var.grp: Factor with the cross-classifying variable, where var.dep is grouped into the categories represented by var.grp.

meansums: Logical, if TRUE, the values reported are the true group mean values. If FALSE (default), the values are reported in the standard way, i.e. the values indicate the difference of the group mean in relation to the intercept (reference group).

title: character vector, used as plot title. Depending on plot type and function, will be set automatically. If title = "", no title is printed. For effect-plots, may also be a character vector of length > 1, to define titles for each sub-plot or facet.

axis.labels: character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.

rev.order: Logical, if TRUE, order of categories (groups) is reversed.

string.interc: Character vector that indicates the reference group (intercept), that is appended to the value label of the grouping variable. Default is "(Intercept)".

axis.title: Character vector of length one or two (depending on the plot function and type), used as title(s) for the x and y axis. If not specified, a default labelling is chosen. Note: Some plot types may not support this argument sufficiently. In such cases, use the returned ggplot-object and add axis titles manually with labs. Use axis.title = "" to remove axis titles.

axis.lim: Numeric vector of length 2, defining the range of the plot axis. Depending on plot type, may effect either x- or y-axis, or both. For multiple plot outputs (e.g., from type = "eff" or type = "slope" in plot_model), axis.lim may also be a list of vectors of length 2, defining axis limits for each plot (only if non-faceted).

geom.colors: user defined color for geoms. See 'Details' in plot_grpfrq.

geom.size: size resp. width of the geoms (bar width, line thickness or point size, depending on plot type and function). Note that bar and bin widths mostly need smaller values than dot sizes.
wrap.title numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

grid.breaks numeric; sets the distance between breaks for the axis, i.e. at every grid.breaks’th position a major grid is being printed.

show.values Logical, whether values should be plotted or not.

digits Numeric, amount of digits after decimal point when rounding estimates or values.

y.offset numeric, offset for text labels when their alignment is adjusted to the top/bottom of the geom (see hjust and vjust).

show.p Logical, adds significance levels to values, or value and variable labels.

show.summary logical, if TRUE (default), a summary with chi-squared statistics (see chisq.test), Cramer’s V or Phi-value etc. is shown. If a cell contains expected values lower than five (or lower than 10 if df is 1), the Fisher’s exact test (see fisher.test) is computed instead of chi-squared test. If the table’s matrix is larger than 2x2, Fisher’s exact test with Monte Carlo simulation is computed.

Value

A ggplot-object.

Examples

data(efc)
# note: "var.grp" does not need to be a factor.
# coercion to factor is done by the function
sjp.aov1(efc$c12hour, efc$e42dep)

---

sjp.chi2

Plot Pearson’s Chi2-Test of multiple contingency tables

Description

Plot p-values of Pearson’s Chi2-tests for multiple contingency tables as ellipses or tiles. Requires a data frame with dichotomous (dummy) variables. Calculation of Chi2-matrix taken from Tales of R.
Usage

sjp.chi2(
  df,
  title = "Pearson's Chi2-Test of Independence",
  axis.labels = NULL,
  wrap.title = 50,
  wrap.labels = 20,
  show.legend = FALSE,
  legend.title = NULL
)

Arguments

df
A data frame with (dichotomous) factor variables.
title
character vector, used as plot title. Depending on plot type and function, will be
set automatically. If title = "", no title is printed. For effect-plots, may also be
a character vector of length > 1, to define titles for each sub-plot or facet.
axis.labels
character vector with labels used as axis labels. Optional argument, since in
most cases, axis labels are set automatically.
wrap.title
numeric, determines how many chars of the plot title are displayed in one line
and when a line break is inserted.
wrap.labels
numeric, determines how many chars of the value, variable or axis labels are
displayed in one line and when a line break is inserted.
show.legend
logical, if TRUE, and depending on plot type and function, a legend is added to
the plot.
legend.title
character vector, used as title for the plot legend.

Value

A ggplot-object.

Examples

# create data frame with 5 dichotomous (dummy) variables
mydf <- data.frame(as.factor(sample(1:2, 100, replace=TRUE)),
  as.factor(sample(1:2, 100, replace=TRUE)),
  as.factor(sample(1:2, 100, replace=TRUE)),
  as.factor(sample(1:2, 100, replace=TRUE)),
  as.factor(sample(1:2, 100, replace=TRUE)))

# create variable labels
items <- list(c("Item 1", "Item 2", "Item 3", "Item 4", "Item 5"))

# plot Chi2-contingency-table
sjp.chi2(mydf, axis.labels = items)
Description

Plot correlation matrix as ellipses or tiles.

Usage

sjp.corr(
  data,
  title = NULL,
  axis.labels = NULL,
  sort.corr = TRUE,
  decimals = 3,
  na.deletion = c("listwise", "pairwise"),
  corr.method = c("pearson", "spearman", "kendall"),
  geom.colors = "RdBu",
  wrap.title = 50,
  wrap.labels = 20,
  show.legend = FALSE,
  legend.title = NULL,
  show.values = TRUE,
  show.p = TRUE,
  p.numeric = FALSE
)

Arguments

data Matrix with correlation coefficients as returned by the cor-function, or a data.frame of variables where correlations between columns should be computed.
title character vector, used as plot title. Depending on plot type and function, will be set automatically. If title = "", no title is printed. For effect-plots, may also be a character vector of length > 1, to define titles for each sub-plot or facet.
axis.labels character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.
sort.corr Logical, if TRUE (default), the axis labels are sorted according to the correlation strength. If FALSE, axis labels appear in order of how variables were included in the cor-computation or data frame.
decimals Indicates how many decimal values after comma are printed when the values labels are shown. Default is 3. Only applies when show.values = TRUE.
na.deletion Indicates how missing values are treated. May be either "listwise" (default) or "pairwise". May be abbreviated.
corr.method Indicates the correlation computation method. May be one of "pearson" (default), "spearman" or "kendall". May be abbreviated.
geom.colors: user defined color for geoms. See 'Details' in plot_grpfrq.

wrap.title: numeric, determines how many chars of the plot title are displayed in one line and when a line break is inserted.

wrap.labels: numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

show.legend: logical, if TRUE, and depending on plot type and function, a legend is added to the plot.

legend.title: character vector, used as title for the plot legend.

show.values: Logical, whether values should be plotted or not.

show.p: Logical, adds significance levels to values, or value and variable labels.

p.numeric: Logical, if TRUE, the p-values are printed as numbers. If FALSE (default), asterisks are used.

Details

Required argument is either a data.frame or a matrix with correlation coefficients as returned by the cor-function. In case of ellipses, the ellipses size indicates the strength of the correlation. Furthermore, blue and red colors indicate positive or negative correlations, where stronger correlations are darker.

Value

(Invisibly) returns the ggplot-object with the complete plot (plot) as well as the data frame that was used for setting up the ggplot-object (df) and the original correlation matrix (corr.matrix).

Note

If data is a matrix with correlation coefficients as returned by the cor-function, p-values can’t be computed. Thus, show.p and p.numeric only have an effect if data is a data.frame.

sjp.poly

Plot polynomials for (generalized) linear regression

Description

This function plots a scatter plot of a term poly.term against a response variable x and adds - depending on the amount of numeric values in poly.degree - multiple polynomial curves. A loess-smoothed line can be added to see which of the polynomial curves fits best to the data.
Usage

sjp.poly(
  x,
  poly.term,  # A vector, representing the response variable of a linear (mixed) model; or a
  poly.degree,  # linear (mixed) model as returned by \texttt{lm} or \texttt{lmer}.
  poly.scale = FALSE,  # If \texttt{x} is a vector, \texttt{poly.term} should also be a vector, representing the polynomial
  fun = NULL,  # term (independent variable) in the model; if \texttt{x} is a fitted model, \texttt{poly.term}
  axis.title = NULL,  # should be the polynomial term's name as character string. See 'Examples'.
  geom.colors = NULL,  # Numeric, or numeric vector, indicating the degree of the polynomial. If \texttt{poly.degree}
  geom.size = 0.8,  # is a numeric vector, multiple polynomial curves for each degree are plotted. See
  show.loess = TRUE,  # 'Examples'.
  show.loess.ci = TRUE,
  show.p = TRUE,
  show.scatter = TRUE,
  point.alpha = 0.2,
  point.color = "#404040",
  loess.color = "#808080"
)

Arguments

\begin{itemize}
  \item \texttt{x}  
    \begin{itemize}
    \item A vector, representing the response variable of a linear (mixed) model; or a
      \begin{itemize}
      \item linear (mixed) model as returned by \texttt{lm} or \texttt{lmer}.
      \end{itemize}
    \end{itemize}
  \item \texttt{poly.term}  
    \begin{itemize}
    \item If \texttt{x} is a vector, \texttt{poly.term} should also be a vector, representing the polynomial
      \begin{itemize}
      \item term (independent variable) in the model; if \texttt{x} is a fitted model, \texttt{poly.term}
      \end{itemize}
    \end{itemize}
  \item \texttt{poly.degree}  
    \begin{itemize}
    \item Numeric, or numeric vector, indicating the degree of the polynomial. If \texttt{poly.degree}
      \begin{itemize}
      \item is a numeric vector, multiple polynomial curves for each degree are plotted. See
      \end{itemize}
    \end{itemize}
  \item \texttt{poly.scale}  
    \begin{itemize}
    \item Logical, if \texttt{TRUE}, \texttt{poly.term} will be scaled before linear regression is computed.
      \begin{itemize}
      \item Default is \texttt{FALSE}. Scaling the polynomial term may have an impact on the resulting p-values.
      \end{itemize}
    \end{itemize}
  \item \texttt{fun}  
    \begin{itemize}
    \item Linear function when modelling polynomial terms. Use \texttt{fun = "lm"} for linear
      \begin{itemize}
      \item models, or \texttt{fun = "glm"} for generalized linear models. When \texttt{x} is not a vector,
      \end{itemize}
    \end{itemize}
  \item \texttt{axis.title}  
    \begin{itemize}
    \item Character vector of length one or two (depending on the plot function and type),
      \begin{itemize}
      \item used as \texttt{title(s)} for the x and y axis. If not specified, a default labelling is chosen.
      \item \textbf{Note:} Some plot types may not support this argument sufficiently. In such
      \end{itemize}
    \end{itemize}
  \item \texttt{geom.colors}  
    \begin{itemize}
    \item user defined color for geoms. See 'Details' in \texttt{plot_grpfrq}.
    \end{itemize}
  \item \texttt{geom.size}  
    \begin{itemize}
    \item size resp. width of the geoms (bar width, line thickness or point size, depending
      \begin{itemize}
      \item on plot type and function). Note that bar and bin widths mostly need smaller
      \end{itemize}
    \end{itemize}
  \item \texttt{show.loess}  
    \begin{itemize}
    \item Logical, if \texttt{TRUE}, an additional loess-smoothed line is plotted.
    \end{itemize}
\end{itemize}
show.loess.ci Logical, if TRUE, a confidence region for the loess-smoothed line will be plotted.
show.p Logical, if TRUE (default), p-values for polynomial terms are printed to the console.
show.scatter Logical, if TRUE (default), adds a scatter plot of data points to the plot.
point.alpha Alpha value of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
point.color Color of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
loess.color Color of the loess-smoothed line. Only applies, if show.loess = TRUE.

Details

For each polynomial degree, a simple linear regression on x (resp. the extracted response, if x is a fitted model) is performed, where only the polynomial term poly.term is included as independent variable. Thus, \( \text{lm}(y \sim x + I(x^2) + \ldots + I(x^i)) \) is repeatedly computed for all values in poly.degree, and the predicted values of the response are plotted against the raw values of poly.term. If x is a fitted model, other covariates are ignored when finding the best fitting polynomial.

This function evaluates raw polynomials, not orthogonal polynomials. Polynomials are computed using the \texttt{poly} function, with argument \texttt{raw = TRUE}.

To find out which polynomial degree fits best to the data, a loess-smoothed line (in dark grey) can be added (with show.loess = TRUE). The polynomial curves that comes closest to the loess-smoothed line should be the best fit to the data.

Value

A ggplot-object.

Examples

library(sjmisc)
data(efc)
# linear fit. loess-smoothed line indicates a more
# or less cubic curve
sjp.poly(efc$c160age, efc$quol_5, 1)

# quadratic fit
sjp.poly(efc$c160age, efc$quol_5, 2)

# linear to cubic fit
sjp.poly(efc$c160age, efc$quol_5, 1:4, show.scatter = FALSE)

# fit sample model
fit <- \texttt{lm}(tot_sc_e \sim c12hour + e17age + e42dep, data = efc)
# inspect relationship between predictors and response
plot_model(fit, type = "slope")
# "e17age" does not seem to be linear correlated to response
# try to find appropriate polynomial. Grey line (loess smoothed)
# indicates best fit. Looks like x^4 has the best fit,
# however, only x^3 has significant p-values.
sjp.poly(fit, "e17age", 2:4, show.scatter = FALSE)

## Not run:
# fit new model
fit <- lm(tot_sc_e ~ c12hour + e42dep + e17age + I(e17age^2) + I(e17age^3),
         data = efc)
# plot marginal effects of polynomial term
plot_model(fit, type = "pred", terms = "e17age")
## End(Not run)

sjplot
Wrapper to create plots and tables within a pipe-workflow

Description

This function has a pipe-friendly argument-structure, with the first argument always being the data, followed by variables that should be plotted or printed as table. The function then transforms the input and calls the requested sjp.- resp. sjt.-function to create a plot or table.

Both sjplot() and sjtab() support grouped data frames.

Usage

sjplot(data, ..., fun = c("grpfrq", "xtab", "aov1", "likert"))

sjtab(data, ..., fun = c("xtab", "stackfrq"))

Arguments

data A data frame. May also be a grouped data frame (see 'Note' and 'Examples').

... Names of variables that should be plotted, and also further arguments passed down to the sjPlot-functions. See 'Examples'.

fun Plotting function. Refers to the function name of sjPlot-functions. See 'Details' and 'Examples'.

Details

Following fun-values are currently supported:

"aov1" calls sjp.aov1. The first two variables in data are used (and required) to create the plot.
"grpfrq" calls plot grpfrq. The first two variables in data are used (and required) to create the plot.
"likert" calls plot likert. data must be a data frame with items to plot.
"stackfrq" calls `tab_stackfrq`. Data must be a data frame with items to create the table. "xtab" calls `plot_xtab` or `tab_xtab`. The first two variables in data are used (and required) to create the plot or table.

**Value**

See related sjp. and sjt.-functions.

**Note**

The `...`-argument is used, first, to specify the variables from data that should be plotted, and, second, to name further arguments that are used in the subsequent plotting functions. Refer to the online-help of supported plotting-functions to see valid arguments.

data may also be a grouped data frame (see `group_by`) with up to two grouping variables. Plots are created for each subgroup then.

**Examples**

```r
library(dplyr)
data(efc)

# Grouped frequencies
efc %>% sjplot(e42dep, c172code, fun = "grpfrq")

# Grouped frequencies, as box plots
efc %>% sjplot(e17age, c172code, fun = "grpfrq",
              type = "box", geom.colors = "Set1")

## Not run:
# table output of grouped data frame
efc %>%
  group_by(e16sex, c172code) %>%
  select(e42dep, n4pstu, e16sex, c172code) %>%
  sjtab(fun = "xtab", use.viewer = FALSE) # open all tables in browser
## End(Not run)
```

---

**sjPlot-themes**

Modify plot appearance

**Description**

Set default plot themes, use pre-defined color scales or modify plot or table appearance.
Usage

theme_sjplot(base_size = 12, base_family = "")

theme_sjplot2(base_size = 12, base_family = "")

theme_blank(base_size = 12, base_family = "")

theme_538(base_size = 12, base_family = "")

font_size(
  title,
  axis_title.x,
  axis_title.y,
  labels.x,
  labels.y,
  offset.x,
  offset.y,
  base.theme
)

label_angle(angle.x, angle.y, base.theme)

legend_style(inside, pos, justify, base.theme)

scale_color_sjplot(palette = "metro", discrete = TRUE, reverse = FALSE, ...)

scale_fill_sjplot(palette = "metro", discrete = TRUE, reverse = FALSE, ...)

sjplot_pal(palette = "metro", n = NULL)

show_sjplot_pals()

css_theme(css.theme = "regression")

Arguments

  base_size  Base font size.
  base_family Base font family.
  title      Font size for plot titles.
  axis_title.x Font size for x-axis titles.
  axis_title.y Font size for y-axis titles.
  labels.x   Font size for x-axis labels.
  labels.y   Font size for y-axis labels.
  offset.x   Offset for x-axis titles.
  offset.y   Offset for y-axis titles.
Optional ggplot-theme-object, which is needed in case multiple functions should be combined, e.g. `theme_sjplot() + label_angle()`. In such cases, use `label_angle(base.theme = theme_sjplot())`.

Angle for x-axis labels.

Angle for y-axis labels.

Logical, use `TRUE` to put legend inside the plotting area. See also `pos`.

Position of the legend, if a legend is drawn.

Legend outside plot Use "bottom", "top", "left" or "right" to position the legend above, below, on the left or right side of the diagram.

Legend inside plot If `inside = TRUE`, legend can be placed inside plot. Use "top left", "top right", "bottom left" and "bottom right" to position legend in any of these corners, or a two-element numeric vector with values from 0-1. See also `inside`.

Justification of legend, relative to its position ("center" or two-element numeric vector with values from 0-1).

Character name of color palette.

Logical, if `TRUE`, a discrete colour palette is returned. Else, a gradient palette is returned, where colours of the requested palette are interpolated using `colorRampPalette`.

Logical, if `TRUE`, order of returned colours is reversed.

Further arguments passed down to ggplot’s `scale()`-functions.

Numeric, number of colors to be returned. By default, the complete colour palette is returned.

Name of the CSS pre-set theme-style. Can be used for table-functions.

When using the `colors` argument in function calls (e.g. `plot_model()`) or when calling one of the predefined scale-functions (e.g. `scale_color_sjplot()`), there are pre-defined colour palettes in this package. Use `show_sjplot_pals()` to show all available colour palettes.

# prepare data
if (requireNamespace("haven")) {
  library(sjmisc)
data(efc)
efc <- to_factor(efc, c161sex, e42dep, c172code)
m <- lm(neg_c_7 ~ pos_v_4 + c12hour + e42dep + c172code, data = efc)

# create plot-object
p <- plot_model(m)

# change theme
p + theme_sjplot()

# change font-size
tab_corr

p + font_size(axis_title.x = 30)

# apply color theme
p + scale_color_sjplot()

# show all available colour palettes
show_sjplot_pals()

# get colour values from specific palette
sjplot_pal(pal = "breakfast club")

---

**tab_corr**

Summary of correlations as HTML table

---

**Description**

Shows the results of a computed correlation as HTML table. Requires either a `data.frame` or a matrix with correlation coefficients as returned by the `cor`-function.

**Usage**

```r
tab_corr(
  data,
  na.deletion = c("listwise", "pairwise"),
  corr.method = c("pearson", "spearman", "kendall"),
  title = NULL,
  var.labels = NULL,
  wrap.labels = 40,
  show.p = TRUE,
  p.numeric = FALSE,
  fade.ns = TRUE,
  val.rm = NULL,
  digits = 3,
  triangle = "both",
  string.diag = NULL,
  CSS = NULL,
  encoding = NULL,
  file = NULL,
  use.viewer = TRUE,
  remove.spaces = TRUE
)
```

**Arguments**

- **data** Matrix with correlation coefficients as returned by the `cor`-function, or a `data.frame` of variables where correlations between columns should be computed.
**na.deletion** Indicates how missing values are treated. May be either "listwise" (default) or "pairwise". May be abbreviated.

**corr.method** Indicates the correlation computation method. May be one of "pearson" (default), "spearman" or "kendall". May be abbreviated.

**title** String, will be used as table caption.

**var.labels** Character vector with variable names, which will be used to label variables in the output.

**wrap.labels** Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

**show.p** Logical, if TRUE, p-values are also printed.

**p.numeric** Logical, if TRUE, the p-values are printed as numbers. If FALSE (default), asterisks are used.

**fade.ns** Logical, if TRUE (default), non-significant correlation-values appear faded (by using a lighter grey text color). See 'Note'.

**val.rm** Specify a number between 0 and 1 to suppress the output of correlation values that are smaller than val.rm. The absolute correlation values are used, so a correlation value of -.5 would be greater than val.rm = .4 and thus not be omitted. By default, this argument is NULL, hence all values are displayed in the table. If a correlation value is below the specified value of val.rm, it is still printed to the HTML table, but made "invisible" with white foreground color. You can use the CSS argument ("css.valueremove") to change color and appearance of those correlation value that are smaller than the limit specified by val.rm.

**digits** Amount of decimals for estimates

**triangle** Indicates whether only the upper right (use "upper"), lower left (use "lower") or both (use "both") triangles of the correlation table is filled with values. Default is "both". You can specify the initial letter only.

**string.diag** A vector with string values of the same length as ncol(data) (number of correlated items) that can be used to display content in the diagonal cells where row and column item are identical (i.e. the "self-correlation"). By default, this argument is NULL and the diagonal cells are empty.

**CSS** A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

**encoding** Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

**file** Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE's viewer pane or the default web browser.

**use.viewer** Logical, if TRUE, the HTML table is shown in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

**remove.spaces** Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.
Value

Invisibly returns

- the web page style sheet (`page.style`),
- the web page content (`page.content`),
- the complete html-output (`page.complete`) and
- the html-table with inline-css for use with knitr (`knitr`)

for further use.

Note

If `data` is a matrix with correlation coefficients as returned by the `cor`-function, p-values can’t be computed. Thus, `show.p`, `p.numeric` and `fade.ns` only have an effect if `data` is a `data.frame`.

Examples

```r
## Not run:
if (interactive()) {
  # Data from the EUROFAMCARE sample dataset
  library(sjmisc)
  data(efc)

  # retrieve variable and value labels
  varlabs <- get_label(efc)

  # receive first item of COPE-index scale
  start <- which(colnames(efc) == "c83cop2")
  # receive last item of COPE-index scale
  end <- which(colnames(efc) == "c88cop7")

  # create data frame with COPE-index scale
  mydf <- data.frame(efc[, c(start:end)])
  colnames(mydf) <- varlabs[c(start:end)]

  # we have high correlations here, because all items
  # belong to one factor.
  tab_corr(mydf, p.numeric = TRUE)

  # auto-detection of labels, only lower triangle
  tab_corr(efc[, c(start:end)], triangle = "lower")

  # auto-detection of labels, only lower triangle, all correlation
  # values smaller than 0.3 are not shown in the table
  tab_corr(efc[, c(start:end)], triangle = "lower", val.rm = 0.3)

  # auto-detection of labels, only lower triangle, all correlation
  # values smaller than 0.3 are printed in blue
  tab_corr(efc[, c(start:end)], triangle = "lower", val.rm = 0.3, CSS = list(css.valueremove = 'color:blue;'))
}
## End(Not run)
```
Print data frames as HTML table.

Description

These functions print data frames as HTML-table, showing the results in RStudio's viewer pane or in a web browser.

Usage

```r
tab_df(
  x,
  title = NULL,
  footnote = NULL,
  col.header = NULL,
  show.type = FALSE,
  show.rownames = FALSE,
  show.footnote = FALSE,
  alternate.rows = FALSE,
  sort.column = NULL,
  digits = 2,
  encoding = "UTF-8",
  CSS = NULL,
  file = NULL,
  use.viewer = TRUE,
  ...
)

tab_dfs(
  x,
  titles = NULL,
  footnotes = NULL,
  col.header = NULL,
  show.type = FALSE,
  show.rownames = FALSE,
  show.footnote = FALSE,
  alternate.rows = FALSE,
  sort.column = NULL,
  digits = 2,
  encoding = "UTF-8",
  CSS = NULL,
  file = NULL,
  use.viewer = TRUE,
  rnames = NULL,
  ...
)
```
Arguments

x  For `tab_df()` a data frame; and for `tab_dfs()`, a list of data frames.
title, titles, footnote, footnotes  Character vector with table caption(s) resp. footnote(s). For `tab_df()`, must be a character of length 1; for `tab_dfs()`, a character vector of same length as x (i.e. one title or footnote per data frame).
col.header  Character vector with elements used as column header for the table. If NULL, column names from x are used as column header.
show.type  Logical, if TRUE, adds information about the variable type to the variable column.
show.rownames  Logical, if TRUE, adds a column with the data frame’s rowname to the table output.
show.footnote  Logical, if TRUE, adds a summary footnote below the table. For `tab_df()` specify the string in footnote, for `tab_dfs()` provide a character vector in footnotes.
alternate.rows  Logical, if TRUE, rows are printed in alternatig colors (white and light grey by default).
sort.column  Numeric vector, indicating the index of the column that should sorted. By default, the column is sorted in ascending order. Use negative index for descending order, for instance, `sort.column = -3` would sort the third column in descending order. Note that the first column with rownames is not counted.
digits  Numeric, amount of digits after decimal point when rounding values.
encoding  Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, `encoding = "Windows-1252"` might be necessary for proper display of special characters.
CSS  A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.
file  Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.
use.viewer  Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.
...  Currently not used.
rnames  Character vector, can be used to set row names when `show.rownames=TRUE`.

Details

How do I use CSS-argument?

With the CSS-argument, the visual appearance of the tables can be modified. To get an overview of all style-sheet-classnames that are used in this function, see return value `page.style` for details. Arguments for this list have following syntax:

1. the class-name as argument name and
2. each style-definition must end with a semicolon

You can add style information to the default styles by using a + (plus-sign) as initial character for the argument attributes. Examples:

- `table = 'border:2px solid red;'` for a solid 2-pixel table border in red.
- `summary = 'font-weight:bold;'` for a bold fontweight in the summary row.
- `lasttablerow = 'border-bottom: 1px dotted blue;'` for a blue dotted border of the last table row.
- `colnames = '+color:green'` to add green color formatting to column names.
- `arc = 'color:blue;'` for a blue text color each 2nd row.
- `caption = '+color:red;'` to add red font-color to the default table caption style.

See further examples in this package-vignette.

Value

A list with following items:

- the web page style sheet (`page.style`),
- the HTML content of the data frame (`page.content`),
- the complete HTML page, including header, style sheet and body (`page.complete`)
- the HTML table with inline-css for use with knitr (`knitr`)
- the file path, if the HTML page should be saved to disk (`file`)

Note

The HTML tables can either be saved as file and manually opened (use argument `file`) or they can be saved as temporary files and will be displayed in the RStudio Viewer pane (if working with RStudio) or opened with the default web browser. Displaying resp. opening a temporary file is the default behaviour.

Examples

```r
## Not run:
data(iris)
data(mtcars)
tab_df(iris[1:5, ])
tab_dfs(list(iris[1:5, ], mtcars[1:5, 1:5]))

# sort 2nd column ascending
tab_df(iris[1:5, ], sort.column = 2)

# sort 2nd column descending
tab_df(iris[1:5, ], sort.column = -2)
## End(Not run)
```
Summary of factor analysis as HTML table

Description

Performs a factor analysis on a data frame or matrix and displays the factors as HTML table, or saves them as file.

In case a data frame is used as parameter, the Cronbach’s Alpha value for each factor scale will be calculated, i.e. all variables with the highest loading for a factor are taken for the reliability test. The result is an alpha value for each factor dimension.

Usage

tab_fa(
    data,
    rotation = "promax",
    method = c("ml", "minres", "wls", "gls", "pa", "minchi", "minrank"),
    nmbr.fctr = NULL,
    fctr.load.tlrn = 0.1,
    sort = FALSE,
    title = "Factor Analysis",
    var.labels = NULL,
    wrap.labels = 40,
    show.cronb = TRUE,
    show.comm = FALSE,
    alternate.rows = FALSE,
    digits = 2,
    CSS = NULL,
    encoding = NULL,
    file = NULL,
    use.viewer = TRUE,
    remove.spaces = TRUE
)

Arguments

data A data frame that should be used to compute a PCA, or a prcomp object.
rotation Rotation of the factor loadings. May be one of "varimax", "quartimax", "promax", "oblimin", "simpimax", "cluster" or "none".
method the factoring method to be used. "ml" will do a maximum likelihood factor analysis (default). "minres" will do a minimum residual (OLS), "wls" will do a weighted least squares (WLS) solution, "gls" does a generalized weighted least squares (GLS), "pa" will do the principal factor solution, "minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair. "minrank" will do a minimum rank factor analysis.
Number of factors used for calculating the rotation. By default, this value is NULL and the amount of factors is calculated according to the Kaiser-criteria.

Specifies the minimum difference a variable needs to have between factor loadings (components) in order to indicate a clear loading on just one factor and not diffusing over all factors. For instance, a variable with 0.8, 0.82 and 0.84 factor loading on 3 possible factors can not be clearly assigned to just one factor and thus would be removed from the principal component analysis. By default, the minimum difference of loading values between the highest and 2nd highest factor should be 0.1

logical, if TRUE, sort the loadings for each factors (items will be sorted in terms of their greatest loading, in descending order)

String, will be used as table caption.

Character vector with variable names, which will be used to label variables in the output.

Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

Logical, if TRUE (default), the cronbach’s alpha value for each factor scale will be calculated, i.e. all variables with the highest loading for a factor are taken for the reliability test. The result is an alpha value for each factor dimension. Only applies when data is a data frame.

Logical, if TRUE, show the communality column in the table.

Logical, if TRUE, rows are printed in alternatig colors (white and light grey by default).

Amount of decimals for estimates

A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.

Invisibly returns

- the web page style sheet (page.style),
- the web page content (page.content),

Value
• the complete html-output (page.complete),
• the html-table with inline-css for use with knitr (knitr),
• the factor.index, i.e. the column index of each variable with the highest factor loading for each factor and
• the removed.items, i.e. which variables have been removed because they were outside of the fc.tr.load.tlrn's range.

for further use.

Note

This method for factor analysis relies on the functions fa and fa.parallel from the psych package.

Examples

```r
## Not run:
# Data from the EUROFAMCARE sample dataset
library(sjmisc)
library(GPArotation)
data(efc)

# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")
# auto-detection of labels
if (interactive()) {
  tab_fa(efc[, start:end])
}
## End(Not run)
```

---

**Description**

This function performs an item analysis with certain statistics that are useful for scale or index development. The resulting tables are shown in the viewer pane resp. webbrowser or can be saved as file. Following statistics are computed for each item of a data frame:

- percentage of missing values
- mean value
- standard deviation
- skew
- item difficulty
- item discrimination
- Cronbach’s Alpha if item was removed from scale
- mean (or average) inter-item-correlation

Optional, following statistics can be computed as well:

- kurtosis
- Shapiro-Wilk Normality Test

If `factor.groups` is not `NULL`, the data frame `df` will be split into groups, assuming that `factor.groups` indicate those columns of the data frame that belong to a certain factor (see return value of function `tab_pca` as example for retrieving factor groups for a scale and see examples for more details).

Usage

```r
tab_itemscale(
  df,
  factor.groups = NULL,
  factor.groups.titles = "auto",
  scale = FALSE,
  min.valid.rowmean = 2,
  alternate.rows = TRUE,
  sort.column = NULL,
  show.shapiro = FALSE,
  show.kurtosis = FALSE,
  show.corr.matrix = TRUE,
  CSS = NULL,
  encoding = NULL,
  file = NULL,
  use.viewer = TRUE,
  remove.spaces = TRUE
)
```

```r
sjt.itemanalysis(
  df,
  factor.groups = NULL,
  factor.groups.titles = "auto",
  scale = FALSE,
  min.valid.rowmean = 2,
  alternate.rows = TRUE,
  sort.column = NULL,
  show.shapiro = FALSE,
  show.kurtosis = FALSE,
  show.corr.matrix = TRUE,
  CSS = NULL,
  encoding = NULL,
  file = NULL,
  use.viewer = TRUE,
  remove.spaces = TRUE
)
```
Arguments

df
A data frame with items.

factor.groups
If not NULL, df will be split into sub-groups, where the item analysis is carried out for each of these groups. Must be a vector of same length as ncol(df), where each item in this vector represents the group number of the related columns of df. If factor.groups = "auto", a principal component analysis with Varimax rotation is performed, and the resulting groups for the components are used as group index. See 'Examples'.

factor.groups.titles
Titles for each factor group that will be used as table caption for each component-table. Must be a character vector of same length as length(unique(factor.groups)). Default is "auto", which means that each table has a standard caption Component x. Use NULL to suppress table captions.

scale
Logical, if TRUE, the data frame's vectors will be scaled when calculating the Cronbach's Alpha value (see item.reliability). Recommended, when the variables have different measures / scales.

min.valid.rowmean
Minimum amount of valid values to compute row means for index scores. Default is 2, i.e. the return values index.scores and df.index.scores are computed for those items that have at least min.valid.rowmean per case (observation, or technically, row). See mean_n for details.

alternate.rows
Logical, if TRUE, rows are printed in alternating colors (white and light grey by default).

sort.column
Numeric vector, indicating the index of the column that should be sorted. By default, the column is sorted in ascending order. Use negative index for descending order, for instance, sort.column = -3 would sort the third column in descending order. Note that the first column with rownames is not counted.

show.shapiro
Logical, if TRUE, a Shapiro-Wilk normality test is computed for each item. See shapiro.test for details.

show.kurtosis
Logical, if TRUE, the kurtosis for each item will also be shown (see kurtosi and describe in the psych-package for more details).

show.corr.matrix
Logical, if TRUE (default), a correlation matrix of each component's index score is shown. Only applies if factor.groups is not NULL and df has more than one group. First, for each case (df's row), the sum of all variables (df's columns) is scaled (using the scale-function) and represents a "total score" for each component (a component is represented by each group of factor.groups). After that, each case (df's row) has a scales sum score for each component. Finally, a correlation of these "scale sum scores" is computed.

CSS
A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

encoding
Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.
file

Destination file, if the output should be saved as file. If \texttt{NULL} (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

use.viewer

Logical, if \texttt{TRUE}, the HTML table is shown in the IDE’s viewer pane. If \texttt{FALSE} or no viewer available, the HTML table is opened in a web browser.

remove.spaces

Logical, if \texttt{TRUE}, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.

\textbf{Value}

Invisibly returns

- \texttt{df.list}: List of data frames with the item analysis for each sub-group (or complete, if \texttt{factor.groups} was \texttt{NULL})
- \texttt{index.scores}: A data frame with of standardized scale / index scores for each case (mean value of all scale items for each case) for each sub-group.
- \texttt{ideal.item.diff}: List of vectors that indicate the ideal item difficulty for each item in each sub-group. Item difficulty only differs when items have different levels.
- \texttt{cronbach.values}: List of Cronbach’s Alpha values for the overall item scale for each sub-group.
- \texttt{knitr.list}: List of html-tables with inline-css for use with knitr for each table (sub-group)
- \texttt{knitr}: html-table of all complete output with inline-css for use with knitr
- \texttt{complete.page}: Complete html-output.

If \texttt{factor.groups = NULL}, each list contains only one element, since just one table is printed for the complete scale indicated by \texttt{df}. If \texttt{factor.groups} is a vector of group-index-values, the lists contain elements for each sub-group.

\textbf{Note}

- The \textit{Shapiro-Wilk Normality Test} (see column \texttt{W(p)}) tests if an item has a distribution that is significantly different from normal.
- \textit{Item difficulty} should range between 0.2 and 0.8. Ideal value is \( p^+(1-p)/2 \) (which mostly is between 0.5 and 0.8).
- For \textit{item discrimination}, acceptable values are 0.20 or higher; the closer to 1.00 the better. See \texttt{item_reliability} for more details.
- In case the total \textit{Cronbach’s Alpha} value is below the acceptable cut-off of 0.7 (mostly if an index has few items), the \textit{mean inter-item-correlation} is an alternative measure to indicate acceptability. Satisfactory range lies between 0.2 and 0.4. See also \texttt{item_intercor}.

\textbf{References}

- Trochim WMK (2008) Types of Reliability. (web)

Examples

```r
# Data from the EUROFAMCARE sample dataset
library(sjmisc)
library(sjlabelled)
data(efc)

# retrieve variable and value labels
varlabs <- get_label(efc)

# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

# create data frame with COPE-index scale
mydf <- data.frame(efc[, start:end])
colnames(mydf) <- varlabs[start:end]

## Not run:
if (interactive()) {
  tab_itemscale(mydf)
  # auto-detection of labels
  tab_itemscale(efc[, start:end])
  # Compute PCA on Cope-Index, and perform a
  # item analysis for each extracted factor.
  indices <- tab_pca(mydf)$factor.index
  tab_itemscale(mydf, factor.groups = indices)

  # or, equivalent
  tab_itemscale(mydf, factor.groups = "auto")
}
## End(Not run)
```

---

**tab_model**  
*Print regression models as HTML table*

**Description**

`tab_model()` creates HTML tables from regression models.
Usage

tab_model(
    ..., 
    transform, 
    show.intercept = TRUE, 
    show.est = TRUE, 
    show.ci = 0.95, 
    show.ci50 = FALSE, 
    show.se = NULL, 
    show.std = NULL, 
    std.response = TRUE, 
    show.p = TRUE, 
    show.stat = FALSE, 
    show.df = FALSE, 
    show.zeroinf = TRUE, 
    show.r2 = TRUE, 
    show.icc = TRUE, 
    show.re.var = TRUE, 
    show.ngroups = TRUE, 
    show.fstat = FALSE, 
    show.aic = FALSE, 
    show.aicc = FALSE, 
    show.dev = FALSE, 
    show.loglik = FALSE, 
    show.obs = TRUE, 
    show.reflvl = FALSE, 
    terms = NULL, 
    rm.terms = NULL, 
    order.terms = NULL, 
    keep = NULL, 
    drop = NULL, 
    title = NULL, 
    pred.labels = NULL, 
    dv.labels = NULL, 
    wrap.labels = 25, 
    bootstrap = FALSE, 
    iterations = 1000, 
    seed = NULL, 
    robust = FALSE, 
    vcov.fun = NULL, 
    vcov.type = NULL, 
    vcov.args = NULL, 
    string.pred = "Predictors", 
    string.est = "Estimate", 
    string.std = "std. Beta", 
    string.ci = "CI", 
    string.se = "std. Error", 
    string.std_se = "standardized std. Error"
string.std_ci = "standardized CI",
string.p = "p",
string.std.p = "std. p",
string.df = "df",
string.stat = "Statistic",
string.std.stat = "std. Statistic",
string.resp = "Response",
string.intercept = "(Intercept)",
strings = NULL,
ci.hyphen = "&nbsp;&ndash;&nbsp;",
minus.sign = "&#45;",
collapse.ci = FALSE,
collapse.se = FALSE,
linebreak = TRUE,
col.order = c("est", "se", "std.est", "std.se", "ci", "std.ci", "ci.inner", "ci.outer",
            "stat", "std.stat", "p", "std.p", "df.error", "response.level"),
digits = 2,
digits.p = 3,
digits.rsq = 3,
digits.re = 2,
emph.p = TRUE,
p.val = NULL,
df.method = NULL,
p.style = c("numeric", "stars", "numeric_stars", "scientific", "scientific_stars"),
p.threshold = c(0.05, 0.01, 0.001),
p.adjust = NULL,
case = "parsed",
auto.label = TRUE,
prefix.labels = c("none", "varname", "label"),
bpe = "median",
CSS = css_theme("regression"),
file = NULL,
use.viewer = TRUE,
encoding = "UTF-8"
)

Arguments

... One or more regression models, including glm's or mixed models. May also be
a list with fitted models. See 'Examples'.

transform A character vector, naming a function that will be applied on estimates and
confidence intervals. By default, transform will automatically use "exp" as
transformation for applicable classes of model (e.g. logistic or poisson regres-
sion). Estimates of linear models remain untransformed. Use NULL if you want
the raw, non-transformed estimates.

show.intercept Logical, if TRUE, the intercepts are printed.

show.est Logical, if TRUE, the estimates are printed.
show.ci
Either logical, and if TRUE, the confidence intervals is printed to the table; if FALSE, confidence intervals are omitted. Or numeric, between 0 and 1, indicating the range of the confidence intervals.

show.ci50
Logical, if TRUE, for Bayesian models, a second credible interval is added to the table output.

show.se
Logical, if TRUE, the standard errors are also printed. If robust standard errors are required, use arguments vcov.fun, vcov.type and vcov.args (see standard_error for details).

show.std
Indicates whether standardized beta-coefficients should also printed, and if yes, which type of standardization is done. See 'Details'.

std.response
Logical, whether the response variable will also be standardized if standardized coefficients are requested. Setting both std.response = TRUE and show.std = TRUE will behave as if the complete data was standardized before fitting the model.

show.p
Logical, if TRUE, p-values are also printed.

show.stat
Logical, if TRUE, the coefficients' test statistic is also printed.

show.df
Logical, if TRUE and p.val = "kr", the p-values for linear mixed models are based on df with Kenward-Rogers approximation. These df-values are printed. See p.value for details.

show.zeroinf
Logical, if TRUE and model has a zero-inflated model part, this is also printed to the table.

show.r2
Logical, if TRUE, the r-squared value is also printed. Depending on the model, these might be pseudo-r-squared values, or Bayesian r-squared etc. See r2 for details.

show.icc
Logical, if TRUE, prints the intraclass correlation coefficient for mixed models. See icc for details.

show.re.var
Logical, if TRUE, prints the random effect variances for mixed models. See get_variance for details.

show.ngroups
Logical, if TRUE, shows number of random effects groups for mixed models.

show.fstat
Logical, if TRUE, the F-statistics for each model is printed in the table summary. This option is not supported by all model types.

show.aic
Logical, if TRUE, the AIC value for each model is printed in the table summary.

show.aicc
Logical, if TRUE, the second-order AIC value for each model is printed in the table summary.

show.dev
Logical, if TRUE, shows the deviance of the model.

show.loglik
Logical, if TRUE, shows the log-Likelihood of the model.

show.obs
Logical, if TRUE, the number of observations per model is printed in the table summary.

show.reflvl
Logical, if TRUE, an additional row is inserted to the table before each predictor of type factor, which will indicate the reference level of the related factor.
**tab_model**

**terms** Character vector with names of those terms (variables) that should be printed in the table. All other terms are removed from the output. If NULL, all terms are printed. Note that the term names must match the names of the model’s coefficients. For factors, this means that the variable name is suffixed with the related factor level, and each category counts as one term. E.g. `rm.terms = "t_name [2,3]"` would remove the terms "t_name2" and "t_name3" (assuming that the variable `t_name` is categorical and has at least the factor levels 2 and 3). Another example for the *iris*-dataset: `terms = "Species"` would not work, instead use `terms = "Species [versicolor,virginica]"`.

**rm.terms** Character vector with names that indicate which terms should be removed from the output. Counterpart to `terms`. `rm.terms = "t_name"` would remove the term `t_name`. Default is NULL, i.e. all terms are used. For factors, levels that should be removed from the plot need to be explicitly indicated in square brackets, and match the model’s coefficient names, e.g. `rm.terms = "t_name [2,3]"` would remove the terms "t_name2" and "t_name3" (assuming that the variable `t_name` was categorical and has at least the factor levels 2 and 3).

**order.terms** Numeric vector, indicating in which order the coefficients should be plotted. See examples in this package-vignette.

**keep, drop** Character containing a regular expression pattern that describes the parameters that should be included (for `keep`) or excluded (for `drop`) in the returned data frame. `keep` may also be a named list of regular expressions. All non-matching parameters will be removed from the output. If `keep` has more than one element, these will be merged with an `OR` operator into a regular expression pattern like this: "(one|two|three)". See further details in ?parameters::model_parameters.

**title** String, will be used as table caption.

**pred.labels** Character vector with labels of predictor variables. If not NULL, `pred.labels` will be used in the first table column with the predictors’ names. By default, if `auto.label = TRUE` and data is labelled, `term_labels` is called to retrieve the labels of the coefficients, which will be used as predictor labels. If data is not labelled, `format_parameters()` is used to create pretty labels. If `pred.labels = ""` or `auto.label = FALSE`, the raw variable names as used in the model formula are used as predictor labels. If `pred.labels` is a named vector, predictor labels (by default, the names of the model’s coefficients) will be matched with the names of `pred.labels`. This ensures that labels always match the related predictor in the table, no matter in which way the predictors are sorted. See 'Examples'.

**dv.labels** Character vector with labels of dependent variables of all fitted models. If `dv.labels = ""`, the row with names of dependent variables is omitted from the table.

**wrap.labels** Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

**bootstrap** Logical, if TRUE, returns bootstrapped estimates.

**iterations** Numeric, number of bootstrap iterations (default is 1000).

**seed** Numeric, the number of the seed to replicate bootstrapped estimates. If NULL, uses random seed.
tab_model

robust    Deprecated. Please use vcov.fun directly to specify the estimation of the variance-covariance matrix.

vcov.fun  Variance-covariance matrix used to compute uncertainty estimates (e.g., for robust standard errors). This argument accepts a covariance matrix, a function which returns a covariance matrix, or a string which identifies the function to be used to compute the covariance matrix. See model_parameters().

vcov.type  Deprecated. The type-argument is now included in vcov.args.

vcov.args List of arguments to be passed to the function identified by the vcov.fun argument. This function is typically supplied by the sandwich or clubSandwich packages. Please refer to their documentation (e.g., ?sandwich::vcovHAC) to see the list of available arguments.

string.pred Character vector, used as headline for the predictor column. Default is "Predictors".

string.est  Character vector, used for the column heading of coefficients. Default is based on the response scale, e.g. for logistic regression models, "Odds Ratios" will be chosen, while for Poisson models it is "Incidence Rate Ratios" etc. Default if not specified is "Estimate".

string.std Character vector, used for the column heading of standardized beta coefficients. Default is "std. Beta".

string.ci  Character vector, used for the column heading of confidence interval values. Default is "CI".

string.se  Character vector, used for the column heading of standard error values. Default is "std. Error".

string.std_se Character vector, used for the column heading of standard error of standardized coefficients. Default is "standardized std. Error".

string.std_ci Character vector, used for the column heading of confidence intervals of standardized coefficients. Default is "standardized std. Error".

string.p  Character vector, used for the column heading of p values. Default is "p".

string.std.p Character vector, used for the column heading of p values. Default is "std. p".

string.df  Character vector, used for the column heading of degrees of freedom. Default is "df".

string.stat Character vector, used for the test statistic. Default is "Statistic".

string.std.stat Character vector, used for the test statistic. Default is "std. Statistic".

string.resp Character vector, used for the column heading of the response level for multinominal or categorical models. Default is "Response".

string.intercept Character vector, used as name for the intercept parameter. Default is "(Intercept)".

strings  Named character vector, as alternative to arguments like string.ci or string.p etc. The name (lhs) must be one of the string-indicator from the aforementioned arguments, while the value (rhs) is the string that is used as column heading. E.g., strings = c(ci = "Conf.Int.", se = "std. Err") would be equivalent to setting string.ci = "Conf.Int.", string.se = "std. Err". 
ci.hyphen | Character vector, indicating the hyphen for confidence interval range. May be an HTML entity. See 'Examples'.

minus.sign | string, indicating the minus sign for negative numbers. May be an HTML entity. See 'Examples'.

collapse.ci | Logical, if FALSE, the CI values are shown in a separate table column.

collapse.se | Logical, if FALSE, the SE values are shown in a separate table column.

linebreak | Logical, if TRUE and collapse.ci = FALSE or collapse.se = FALSE, inserts a line break between estimate and CI resp. SE values. If FALSE, values are printed in the same line as estimate values.

col.order | Character vector, indicating which columns should be printed and in which order. Column names that are excluded from col.order are not shown in the table output. However, column names that are included, are only shown in the table when the related argument (like show.est for "estimate") is set to TRUE or another valid value. Table columns are printed in the order as they appear in col.order.

digits | Amount of decimals for estimates

digits.p | Amount of decimals for p-values

digits.rsq | Amount of decimals for r-squared values

digits.re | Amount of decimals for random effects part of the summary table.

emph.p | Logical, if TRUE, significant p-values are shown bold faced.

df.method, p.val | Method for computing degrees of freedom for p-values, standard errors and confidence intervals (CI). Only applies to mixed models. Use df.method = "wald" for a faster, but less precise computation. This will use the residual degrees of freedom (as returned by df.residual()) for linear mixed models, and Inf degrees if freedom for all other model families. df.method = "kenward" (or df.method = "kr") uses Kenward-Roger approximation for the degrees of freedom. df.method = "satterthwaite" uses Satterthwaite's approximation and "ml1" uses a "m-l-1" heuristic see degrees_of_freedom for details). Use show.df = TRUE to show the approximated degrees of freedom for each coefficient.

p.style | Character, indicating if p-values should be printed as numeric value ("numeric"), as 'stars' (asterisks) only ("stars"), or scientific ("scientific"). Scientific and numeric style can be combined with "stars", e.g. "numeric_stars"

p.threshold | Numeric vector of length 3, indicating the treshold for annotating p-values with asterisks. Only applies if p.style = "asterisk".

p.adjust | Character vector, if not NULL, indicates the method to adjust p-values. See p.adjust for details.

case | Desired target case. Labels will automatically converted into the specified character case. See snakecase::to_any_case() for more details on this argument. By default, if case is not specified, it will be set to "parsed", unless prefix.labels is not "none". If prefix.labels is either "label" (or "l") or "varname" (or "v") and case is not specified, it will be set to NULL - this is a more convenient default when prefixing labels.
auto.label Logical, if TRUE (the default), and data is labelled, term_labels is called to retrieve the labels of the coefficients, which will be used as predictor labels. If data is not labelled, format_parameters() is used to create pretty labels. If auto.label = FALSE, original variable names and value labels (factor levels) are used.

prefix.labels Indicates whether the value labels of categorical variables should be prefixed, e.g. with the variable name or variable label. See argument prefix in term_labels for details.

bpe For Stan-models (fitted with the rstanarm- or brms-package), the Bayesian point estimate is, by default, the median of the posterior distribution. Use bpe to define other functions to calculate the Bayesian point estimate. bpe needs to be a character naming the specific function, which is passed to the fun-argument in typical_value. So, bpe = "mean" would calculate the mean value of the posterior distribution.

CSS A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

file Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

use.viewer Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

encoding Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

Details

**Standardized Estimates:** Default standardization is done by completely refitting the model on the standardized data. Hence, this approach is equal to standardizing the variables before fitting the model, which is particularly recommended for complex models that include interactions or transformations (e.g., polynomial or spline terms). When show.std = "std2", standardization of estimates follows Gelman’s (2008) suggestion, rescaling the estimates by dividing them by two standard deviations instead of just one. Resulting coefficients are then directly comparable for untransformed binary predictors. For backward compatibility reasons, show.std also may be a logical value; if TRUE, normal standardized estimates are printed (same effect as show.std = "std"). Use show.std = NULL (default) or show.std = FALSE, if no standardization is required.

**How do I use CSS-argument?:** With the CSS-argument, the visual appearance of the tables can be modified. To get an overview of all style-sheet-classnames that are used in this function, see return value page.style for details. Arguments for this list have following syntax:

1. the class-names with "css."-prefix as argument name and
2. each style-definition must end with a semicolon

You can add style information to the default styles by using a + (plus-sign) as initial character for the argument attributes. Examples:

- css.table = 'border:2px solid red;' for a solid 2-pixel table border in red.
- css.summary = 'font-weight:bold;' for a bold fontweight in the summary row.
• css.lasttablerow = 'border-bottom: 1px dotted blue;' for a blue dotted border of the last table row.
• css.colnames = '+color: green' to add green color formatting to column names.
• css.arc = 'color: blue;' for a blue text color each 2nd row.
• css.caption = '+color: red;' to add red font-color to the default table caption style.

Value

Invisibly returns
• the web page style sheet (page.style),
• the web page content (page.content),
• the complete html-output (page.complete) and
• the html-table with inline-css for use with knitr (knitr)

for further use.

Note

The HTML tables can either be saved as file and manually opened (use argument file) or they can be saved as temporary files and will be displayed in the RStudio Viewer pane (if working with RStudio) or opened with the default web browser. Displaying resp. opening a temporary file is the default behaviour (i.e. file = NULL).

Examples are shown in these three vignettes: Summary of Regression Models as HTML Table, Summary of Mixed Models as HTML Table and Summary of Bayesian Models as HTML Table.

---

**tab_pca**  
*Summary of principal component analysis as HTML table*

**Description**

Performes a principle component analysis on a data frame or matrix (with varimax or oblimin rotation) and displays the factor solution as HTML table, or saves them as file.

In case a data frame is used as parameter, the Cronbach’s Alpha value for each factor scale will be calculated, i.e. all variables with the highest loading for a factor are taken for the reliability test. The result is an alpha value for each factor dimension.

**Usage**

```r
tab_pca(
  data,
  rotation = c("varimax", "quartimax", "promax", "oblimin", "simplimax", "cluster", "none"),
  nmbr.fctr = NULL,
  fctr.load.tlrn = 0.1,
)```
Arguments

**data**
A data frame that should be used to compute a PCA, or a \texttt{prcomp} object.

**rotation**
Rotation of the factor loadings. May be one of \texttt{"varimax"}, \texttt{"quartimax"}, \texttt{"promax"}, \texttt{"oblimin"}, \texttt{"simplimax"}, \texttt{"cluster"} or \texttt{"none"}.

**nmbr.fctr**
Number of factors used for calculating the rotation. By default, this value is \texttt{NULL} and the amount of factors is calculated according to the Kaiser-criteria.

**fctr.load.tlrn**
Specifies the minimum difference a variable needs to have between factor loadings (components) in order to indicate a clear loading on just one factor and not diffusing over all factors. For instance, a variable with 0.8, 0.82 and 0.84 factor loading on 3 possible factors can not be clearly assigned to just one factor and thus would be removed from the principal component analysis. By default, the minimum difference of loading values between the highest and 2nd highest factor should be 0.1

**title**
String, will be used as table caption.

**var.labels**
Character vector with variable names, which will be used to label variables in the output.

**wrap.labels**
Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

**show.cronb**
Logical, if \texttt{TRUE} (default), the cronbach’s alpha value for each factor scale will be calculated, i.e. all variables with the highest loading for a factor are taken for the reliability test. The result is an alpha value for each factor dimension. Only applies when \texttt{data} is a data frame.

**show.msa**
Logical, if \texttt{TRUE}, shows an additional column with the measure of sampling adequacy according dor each component.

**show.var**
Logical, if \texttt{TRUE}, the proportions of variances for each component as well as cumulative variance are shown in the table footer.

**alternate.rows**
Logical, if \texttt{TRUE}, rows are printed in alternating colors (white and light grey by default).
digits Amount of decimals for estimates

string.pov String for the table row that contains the proportions of variances. By default, "Proportion of Variance" will be used.

string.cpv String for the table row that contains the cumulative variances. By default, "Cumulative Proportion" will be used.

CSS A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

encoding Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

file Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

use.viewer Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

remove.spaces Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.

Value

Invisibly returns

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (page.complete),
- the html-table with inline-css for use with knitr (knitr),
- the factor.index, i.e. the column index of each variable with the highest factor loading for each factor and
- the removed.items, i.e. which variables have been removed because they were outside of the fctr.load.tlrn’s range.

for further use.

Examples

## Not run:
# Data from the EUROFAMCARE sample dataset
library(sjmisc)
data(efc)

# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")
# auto-detection of labels
if (interactive()) {
  tab_pca(efc[, start:end])
}
## End(Not run)

---

**tab_stackfreq**

*Summary of stacked frequencies as HTML table*

**Description**

Shows the results of stacked frequencies (such as likert scales) as HTML table. This function is useful when several items with identical scale/categories should be printed as table to compare their distributions (e.g. when plotting scales like SF, Barthel-Index, Quality-of-Life-scales etc.).

**Usage**

```r
tab_stackfreq(
  items,
  weight.by = NULL,
  title = NULL,
  var.labels = NULL,
  value.labels = NULL,
  wrap.labels = 20,
  sort.frq = NULL,
  alternate.rows = FALSE,
  digits = 2,
  string.total = "N",
  string.na = "NA",
  show.n = FALSE,
  show.total = FALSE,
  show.na = FALSE,
  show.skew = FALSE,
  show.kurtosis = FALSE,
  digits.stats = 2,
  file = NULL,
  encoding = NULL,
  CSS = NULL,
  use.viewer = TRUE,
  remove.spaces = TRUE
)
```

**Arguments**

- **items**: Data frame, or a grouped data frame, with each column representing one item.
- **weight.by**: Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.
- **title**: String, will be used as table caption.
var.labels: Character vector with variable names, which will be used to label variables in the output.

value.labels: Character vector (or list of character vectors) with value labels of the supplied variables, which will be used to label variable values in the output.

wrap.labels: Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.

sort.frq: logical, indicates whether the items should be ordered by by highest count of first or last category of items.
- Use "first.asc" to order ascending by lowest count of first category,
- "first.desc" to order descending by lowest count of first category,
- "last.asc" to order ascending by lowest count of last category,
- "last.desc" to order descending by lowest count of last category,
- or NULL (default) for no sorting.

alternate.rows: Logical, if TRUE, rows are printed in alternating colors (white and light grey by default).

digits: Numeric, amount of digits after decimal point when rounding values.

string.total: label for the total N column.

string.na: label for the missing column/row.

show.n: logical, if TRUE, adds total number of cases for each group or category to the labels.

show.total: logical, if TRUE, an additional column with each item’s total N is printed.

show.na: logical, if TRUE, NA’s (missing values) are added to the output.

show.skew: logical, if TRUE, an additional column with each item’s skewness is printed. The skewness is retrieved from the describe-function of the psych-package.

show.kurtosis: Logical, if TRUE, the kurtosis for each item will also be shown (see kurtosi and describe in the psych-package for more details).

digits.stats: amount of digits for rounding the skewness and kurtosis values. Default is 2, i.e. skewness and kurtosis values have 2 digits after decimal point.

file: Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

encoding: Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

CSS: A list with user-defined style-sheet-definitions, according to the official CSS syntax. See ’Details’ or this package-vignette.

use.viewer: Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

remove.spaces: Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.
Value

Invisibly returns

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (page.complete) and
- the html-table with inline-css for use with knitr (knitr)

for further use.

Examples

```r
# random sample
# prepare data for 4-category likert scale, 5 items
likert_4 <- data.frame(
  as.factor(sample(1:4, 500, replace = TRUE, prob = c(0.2, 0.3, 0.1, 0.4))),
  as.factor(sample(1:4, 500, replace = TRUE, prob = c(0.5, 0.25, 0.15, 0.1))),
  as.factor(sample(1:4, 500, replace = TRUE, prob = c(0.25, 0.1, 0.4, 0.25))),
  as.factor(sample(1:4, 500, replace = TRUE, prob = c(0.1, 0.4, 0.4, 0.1))),
  as.factor(sample(1:4, 500, replace = TRUE, prob = c(0.35, 0.25, 0.15, 0.25)))
)
#
# create labels
levels_4 <- c("Independent", "Slightly dependent",
  "Dependent", "Severely dependent")
#
# create item labels
items <- c("Q1", "Q2", "Q3", "Q4", "Q5")

# plot stacked frequencies of 5 (ordered) item-scales
## Not run:
if (interactive()) {
  tab_stackfrq(likert_4, value.labels = levels_4, var.labels = items)
}
```

```r
# Data from the EUROFAMCARE sample dataset
# Auto-detection of labels
# --------------------------------

data(efc)
# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive first item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

# Data from the EUROFAMCARE sample dataset
# Auto-detection of labels
# --------------------------------
data(efc)
# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive first item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

# Data from the EUROFAMCARE sample dataset
# Auto-detection of labels
# --------------------------------

data(efc)
# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive first item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

tab_stackfrq(efc[, c(start:end)], alternate.rows = TRUE)

tab_stackfrq(efc[, c(start:end)], alternate.rows = TRUE,
  show.n = TRUE, show.na = TRUE)
```
tab_xtab

# User defined style sheet
# --------------------------------
tab_stackfrq(ec[, c(start:end)], alternate.rows = TRUE, show.total = TRUE, show.skew = TRUE, show.kurtosis = TRUE, CSS = list(css.ncol = "border-left:1px dotted black;", css.summary = "font-style:italic;"))

## End(Not run)

---

**tab_xtab**

*Summary of contingency tables as HTML table*

**Description**

Shows contingency tables as HTML file in browser or viewer pane, or saves them as file.

**Usage**

```r
tab_xtab(
  var.row,
  var.col,
  weight.by = NULL,
  title = NULL,
  var.labels = NULL,
  value.labels = NULL,
  wrap.labels = 20,
  show.obs = TRUE,
  show.cell.prc = FALSE,
  show.row.prc = FALSE,
  show.col.prc = FALSE,
  show.exp = FALSE,
  show.legend = FALSE,
  show.na = FALSE,
  show.summary = TRUE,
  drop.empty = TRUE,
  statistics = c("auto", "cramer", "phi", "spearman", "kendall", "pearson", "fisher"),
  string.total = "Total",
  digits = 1,
  tdcol.n = "black",
  tdcol.expected = "#339999",
  tdcol.cell = "#993333",
  tdcol.row = "#333399",
  tdcol.col = "#339933",
  emph.total = FALSE,
  emph.color = "#f8f8f8",
  prc.sign = "&nbsp;&#37;",
)```
Arguments

var.row Variable that should be displayed in the table rows.
### tab_xtab

- **var.col**: Variable that should be displayed in the table columns.
- **weight.by**: Vector of weights that will be applied to weight all cases. Must be a vector of same length as the input vector. Default is NULL, so no weights are used.
- **title**: String, will be used as table caption.
- **var.labels**: Character vector with variable names, which will be used to label variables in the output.
- **value.labels**: Character vector (or list of character vectors) with value labels of the supplied variables, which will be used to label variable values in the output.
- **wrap.labels**: Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.
- **show.obs**: Logical, if TRUE, observed values are shown.
- **show.cell.prc**: Logical, if TRUE, cell percentage values are shown.
- **show.row.prc**: Logical, if TRUE, row percentage values are shown.
- **show.col.prc**: Logical, if TRUE, column percentage values are shown.
- **show.exp**: Logical, if TRUE, expected values are also shown.
- **show.legend**: Logical, if TRUE, and depending on plot type and function, a legend is added to the plot.
- **show.na**: Logical, if TRUE, NA’s (missing values) are added to the output.
- **show.summary**: Logical, if TRUE, a summary row with chi-squared statistics, degrees of freedom and Cramer’s V or Phi coefficient and p-value for the chi-squared statistics.
- **drop.empty**: Logical, if TRUE and the variable’s values are labeled, values / factor levels with no occurrence in the data are omitted from the output. If FALSE, labeled values that have no observations are still printed in the table (with frequency 0).
- **statistics**: Name of measure of association that should be computed. May be one of "auto", "cramer", "phi", "spearman", "kendall", "pearson" or "fisher". See xtab_statistics.
- **string.total**: Character label for the total column / row header.
- **digits**: Amount of decimals for estimates.
- **tdcol.n**: Color for highlighting count (observed) values in table cells. Default is black.
- **tdcol.expected**: Color for highlighting expected values in table cells. Default is cyan.
- **tdcol.cell**: Color for highlighting cell percentage values in table cells. Default is red.
- **tdcol.row**: Color for highlighting row percentage values in table cells. Default is blue.
- **tdcol.col**: Color for highlighting column percentage values in table cells. Default is green.
- **emph.total**: Logical, if TRUE, the total column and row will be emphasized with a different background color. See emph.color.
- **emph.color**: Logical, if emph.total = TRUE, this color value will be used for painting the background of the total column and row. Default is a light grey.
- **prc.sign**: The percentage sign that is printed in the table cells, in HTML-format. Default is "&nbsp;%", hence the percentage sign has a non-breaking-space after the percentage value.
hundred  Default value that indicates the 100-percent column-sums (since rounding values may lead to non-exact results). Default is "100.0".

CSS  A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

encoding  String, indicating the charset encoding used for variable and value labels. Default is NULL, so encoding will be auto-detected depending on your platform (e.g., "UTF-8" for Unix and "Windows-1252" for Windows OS). Change encoding if specific chars are not properly displayed (e.g. German umlauts).

file  Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

use.viewer  Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

remove.spaces  Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.

...  Other arguments, currently passed down to the test statistics functions chisq.test() or fisher.test().

Value

Invisibly returns

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (page.complete) and
- the html-table with inline-css for use with knitr (knitr)

for further use.

Examples

# prepare sample data set
data(efc)

# print simple cross table with labels
## Not run:
if (interactive()) {
  tab_xtab(efc$e16sex, efc$e42dep)
}

# print cross table with manually set
# labels and expected values
tab_xtab(
  efc$e16sex,
  efc$e42dep,
  var.labels = c("Elder's gender", "Elder's dependency"),
  show.exp = TRUE

# print minimal cross table with labels, total col/row highlighted
tab_xtab(efc$e16sex, efc$e42dep, show.cell.prc = FALSE, emph.total = TRUE)

# User defined style sheet
tab_xtab(efc$e16sex, efc$e42dep,
    CSS = list(css.table = "border: 2px solid;",
               css.tdata = "border: 1px solid;",
               css.horline = "border-bottom: double blue;"))

# ordinal data, use Kendall's tau
tab_xtab(efc$e42dep, efc$quol_5, statistics = "kendall")

# calculate Spearman's rho, with continuity correction
tab_xtab(
    efc$e42dep,
    efc$quol_5,
    statistics = "spearman",
    exact = FALSE,
    continuity = TRUE
)

## End(Not run)

---

**view_df**

View structure of labelled data frames

**Description**

Save (or show) content of an imported SPSS, SAS or Stata data file, or any similar labelled data.frame, as HTML table. This quick overview shows variable ID number, name, label, type and associated value labels. The result can be considered as "codeplan" of the data frame.

**Usage**

```r
view_df(x, weight.by = NULL, alternate.rows = TRUE, show.id = TRUE, show.type = FALSE, show.values = TRUE, show.string.values = FALSE, show.labels = TRUE, show.frq = FALSE, show.prc = FALSE, show.wtd.frq = FALSE,)
```
show.wtd.prc = FALSE,
show.na = FALSE,
max.len = 15,
sort.by.name = FALSE,
wrap.labels = 50,
verbose = FALSE,
CSS = NULL,
encoding = NULL,
file = NULL,
use.viewer = TRUE,
remove.spaces = TRUE
)

Arguments

x
A (labelled) data frame, imported by read_spss, read_sas or read_stata function, or any similar labelled data frame (see set_label and set_labels).

weight.by
Name of variable in x that indicated the vector of weights that will be applied to weight all observations. Default is NULL, so no weights are used.

alternate.rows
Logical, if TRUE, rows are printed in alternating colors (white and light grey by default).

show.id
Logical, if TRUE (default), the variable ID is shown in the first column.

show.type
Logical, if TRUE, adds information about the variable type to the variable column.

show.values
Logical, if TRUE (default), the variable values are shown as additional column.

show.string.values
Logical, if TRUE (default), elements of character vectors are also shown. By default, these are omitted due to possibly overlengthy tables.

show.labels
Logical, if TRUE (default), the value labels are shown as additional column.

show.frq
Logical, if TRUE, an additional column with frequencies for each variable is shown.

show.prc
Logical, if TRUE, an additional column with percentage of frequencies for each variable is shown.

show.wtd.frq
Logical, if TRUE, an additional column with weighted frequencies for each variable is shown. Weights stem from weight by.

show.wtd.prc
Logical, if TRUE, an additional column with weighted percentage of frequencies for each variable is shown. Weights stem from weight by.

show.na
logical, if TRUE, NA’s (missing values) are added to the output.

max.len
Numeric, indicates how many values and value labels per variable are shown. Useful for variables with many different values, where the output can be truncated.

sort.by.name
Logical, if TRUE, rows are sorted according to the variable names. By default, rows (variables) are ordered according to their order in the data frame.

wrap.labels
Numeric, determines how many chars of the value, variable or axis labels are displayed in one line and when a line break is inserted.
verbose Logical, if TRUE, a progress bar is displayed while creating the output.

CSS A list with user-defined style-sheet-definitions, according to the official CSS syntax. See 'Details' or this package-vignette.

encoding Character vector, indicating the charset encoding used for variable and value labels. Default is "UTF-8". For Windows Systems, encoding = "Windows-1252" might be necessary for proper display of special characters.

file Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and opened either in the IDE’s viewer pane or the default web browser.

use.viewer Logical, if TRUE, the HTML table is shown in the IDE’s viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

remove.spaces Logical, if TRUE, leading spaces are removed from all lines in the final string that contains the html-data. Use this, if you want to remove parantheses for html-tags. The html-source may look less pretty, but it may help when exporting html-tables to office tools.

Value

Invisibly returns

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (page.complete) and
- the html-table with inline-css for use with knitr (knitr)

for further use.

Examples

## Not run:
# init dataset
data(efc)

# view variables
view_df(efc)

# view variables w/o values and value labels
view_df(efc, show.values = FALSE, show.labels = FALSE)

# view variables including variable typed, orderd by name
view_df(efc, sort.by.name = TRUE, show.type = TRUE)

# User defined style sheet
view_df(efc,
   CSS = list(css.table = "border: 2px solid;",
               css.tdata = "border: 1px solid;",
               css.arc = "color:blue;"))

## End(Not run)
Index

* data
  efc, 8

chisq.test, 15, 20, 52, 63
ci, 33, 42
colorRampPalette, 72
cor, 65, 66, 73, 75
css_theme (sjPlot-themes), 70
data.frame, 66, 73, 75
degrees_of_freedom, 91
describe, 83, 97
dist_chisq, 4
dist_f, 5
dist_norm, 6
dist_t, 7
efc, 8
fa, 81
fa.parallel, 81
facet_grid, 21
facet_wrap, 21
factor, 88
fisher.test, 20, 52, 63
font_size (sjPlot-themes), 70

get_model_data (plot_model), 28
get_variance, 88
ggeffect, 30, 36
ggmeans, 31
ggpredict, 30, 32, 36
gls, 44
ggrid.arrange, 21
group_by, 70
group_var, 12, 20
guide_legend, 27

icc, 88
item_intercor, 84
item_reliability, 83, 84

kurtosi, 83, 97

label_angle (sjPlot-themes), 70
labs, 10, 33, 40, 62, 67
legend_style (sjPlot-themes), 70
list, 74, 77, 80, 83, 92, 95, 97, 102, 105
lm, 22, 67
lmer, 67

model_parameters (), 34, 42, 90

NA, 11, 20, 97, 101, 104

p.adjust, 35, 42, 91
p_value, 88
plot.ggeffects, 36
plot_frq, 9
plot_gpt, 13
plot_grid, 16, 21, 27
plot_kfold_cv, 22
plot_likert, 23, 69
plot_model, 15, 28, 62
plot_models, 38
plot_residuals, 43
plot_scatter, 45
plot_stackfraq, 48
plot_xtab, 50, 70
png, 55
poly, 68
prcomp, 79, 94

r2, 88
ranef, 33
read_sas, 104
read_spss, 8, 104
read_stata, 104
response_labels, 10, 15, 33, 40, 46

save_plot, 54
scale, 83
scale_color_sjplot (sjPlot-themes), 70
scale_fill_sjplot (sjPlot-themes), 70
set_label, 104
set_labels, 104
set_theme, 26, 56
shapiro.test, 83
show_sjplot_pals (sjPlot-themes), 70
sjp.aov1, 61, 69
sjp.chi2, 63
sjp.corr, 65
sjp.poly, 66
sjPlot (sjPlot-package), 3
sjplot, 69
sjPlot-package, 3
sjPlot-themes, 70
sjplot_pal (sjPlot-themes), 70
sjt.itemanalysis (tab_itemscale), 81
sjt.xtab (tab_xtab), 99
sjtab (sjplot), 69
standard_error, 34, 88

tab_corr, 73
tab_df, 76
tab_dfs (tab_df), 76
tab_fa, 79
tab_itemscale, 81
tab_model, 85
tab_pca, 82, 93
tab_stackfreq, 70, 96
tab_xtab, 70, 99
term_labels, 33, 35, 40, 42, 89, 92
theme_538 (sjPlot-themes), 70
theme_blank (sjPlot-themes), 70
theme_sjplot (sjPlot-themes), 70
theme_sjplot2 (sjPlot-themes), 70
typical_value, 35, 92

values_at, 32
view_df, 103

xtab_statistics, 101