Package 'sjPlot'

February 5, 2018

Type Package

Encoding UTF-8

Title Data Visualization for Statistics in Social Science

Version 2.4.1

Date 2018-02-05

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 (>= 3.2), graphics, grDevices, stats, utils

Imports arm, broom (>= 0.4.2), dplyr (>= 0.7.1), effects, forcats, ggeffects (>= 0.3.1), glmmTMB, ggplot2 (>= 2.2.1), knitr, lme4 (>= 1.1-12), magrittr, MASS, merTools (>= 0.3.0), modelr, nlme, psych, purrr, rlang, scales, sjlabelled (>= 1.0.7), sjmisc (>= 2.6.3), sjstats (>= 0.14.0), tidyselect, tibble (>= 1.3.3), tidyr (>= 0.7.0)

Suggests AICcmodavg, car, cluster, GPArotation, gridExtra, ggrepel, ggridges, lmerTest, lmtest, rstanarm, survey, viridis, wesanderson, Zelig

URL https://github.com/strengejacke/sjPlot

BugReports https://github.com/strengejacke/sjPlot/issues

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Lüdecke [aut, cre],
Carsten Schwemmer [ctb]

Repository CRAN

Date/Publication 2018-02-05 16:57:18 UTC

R topics documented:

`sjPlot-package` *Data Visualization for Statistics in Social Science*

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 `http://www.strengejacke.de/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

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

Arguments

`chi2`	Numeric, optional. If specified, a chi-squared distribution with `deg.f` degrees of freedom is plotted and a shaded area at `chi2` value position is plotted that indicates whether or not the specified value is significant or not. If both `chi2` and `p` are not specified, a distribution without shaded area is plotted.
`deg.f`	Numeric. The degrees of freedom for the chi-squared distribution. Needs to be specified.
`p`	Numeric, optional. If specified, a chi-squared 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 `chi2` 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 `sjp.grpfrq`.
`geom.alpha`	Specifies the alpha-level of the shaded area. Default is 0.7, range between 0 to 1.

Examples

```
# 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.
`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 `sjp.grpfrq`.
`geom.alpha`	Specifies the alpha-level of the shaded area. Default is 0.7, range between 0 to 1.

Examples

```
# a simple F distribution for 6 and 45 degrees of freedom
dist_f(deg.f1 = 6, deg.f2 = 45)

# F distribution for 6 and 45 degrees of freedom,
# and a shaded area starting at F value of two.
# F-values equal or greater than 2.31 are "significant"
dist_f(f = 2, deg.f1 = 6, deg.f2 = 45)

# F distribution for 6 and 45 degrees of freedom,
# and a shaded area starting at a p-level of 0.2
# (F-Value about 1.5).
dist_f(p = 0.2, deg.f1 = 6, deg.f2 = 45)
```

---

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

```
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.
mean	Numeric. Mean value for normal distribution. By default 0.
sd	Numeric. Standard deviation for normal distribution. By default 1.
p	Numeric, optional. If specified, a normal distribution with mean and sd is plotted and a shaded area at the position where the specified p-level starts is plotted. If both norm 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 sjp.grpfrq.
geom.alpha	Specifies the alpha-level of the shaded area. Default is 0.7, range between 0 to 1.

Examples

```
# a simple normal distribution
dist_norm()

# a simple normal distribution with different mean and sd.
# note that curve looks similar to above plot, but axis range
# has changed.
dist_norm(mean = 2, sd = 4)

# a simple normal distribution
dist_norm(norm = 1)

# a simple normal distribution
dist_norm(p = 0.2)
```

`dist_t` *Plot t-distributions*

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

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

Examples

```
# 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_grid *Arrange list of plots as grid*

Description

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

Usage

```
plot_grid(x, margin = c(1, 1, 1, 1))
```

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.

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

```
data(efc)
# fit model
fit <- lm(tot_sc_e ~ c12hour + e17age + e42dep + neg_c_7, data = efc)
# plot marginal effects for each predictor, each as single plot
p <- sjp.lm(fit, type = "eff", facet.grid = FALSE, prnt.plot = FALSE)

# plot grid
plot_grid(p$plot.list)

# or
plot_grid(p)
```

---

plot_model *Plot regression models*

Description

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

Usage

```
plot_model(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"), mdrt.values = c("minmax", "meansd", "zeromax",
  "quart", "all"), ri.nr = NULL, title = NULL, axis.title = NULL,
  axis.labels = NULL, wrap.title = 50, wrap.labels = 25,
  axis.lim = NULL, grid.breaks = NULL, ci.lvl = NULL, se = NULL,
  colors = "Set1", show.intercept = FALSE, show.values = FALSE,
  show.p = TRUE, show.data = FALSE, show.legend = TRUE,
  value.offset = NULL, value.size, digits = 2, dot.size = NULL,
  line.size = NULL, vline.color = NULL, grid, case = "parsed",
  auto.label = TRUE, bpe = "median", bpe.style = "line", ...)

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 beta values.

`type = "std2"` Forest-plot of standardized beta values, 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.

`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).

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

`type = "diag"` Check model assumptions.

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.

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]")`. 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 highedt 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. Note that this 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).

`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, get_dv_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, get_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.

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

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 hdi (High Density Interval) that is plotted. By default, stanreg-models are printed with two intervals: the "inner" interval, which defaults to the 50%-HDI; and the "outer" interval, which defaults to the 89%-HDI. ci.lvl affects only the outer interval in such cases. See prob.inner and prob.outer under the ...-argument for more details.

se — Either a logical, and if TRUE, error bars indicate standard errors, not confidence intervals. Or a character vector with a specification of the covariance matrix to compute robust standard errors (see argument vcov of link[sjstats]{robust} for valid values; robust standard errors are only supported for models that work with coeftest). se overrides ci.lvl: if not NULL, arguments ci.lvl and transform will be ignored. Currently, se only applies to *Coefficients* plots.

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 display.brewer.all to view all available palette names.
- If **wesanderson** is installed, you may also specify a name of a palette from that package.
- If **viridis** is installed, use colors = "v" to get the viridis color palette.
- There are some pre-defined color palettes in this package: "aqua", "warm", "dust", "blambus",
- Else specify own color values or names as vector (e.g. colors = "#00ff00" or colors = c("firebrick", "blue")).

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

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

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

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

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

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.

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.

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 to_any_case for more details on this argument.

auto.label Logical, if TRUE (the default), plot-labels are based on value and variable labels, if the data is labelled. See get_label and get_term_labels for details. If FALSE, original variable names and value labels (factor levels) are used.

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 Bayesion 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 For **Stan**-models (fitted with the **rstanarm**- or **brms**-package), the Bayesian point estimate is indicated as a small, vertical line by default. Use bpe.style = "dot" to plot a dot instead of a line for the point estimate.

... 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`.

`width`, `alpha` **and** `scale` Passed down to `geom_errorbar()` or `geom_density_ridges()`, for forest or diagnostic plots; or 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 `to_any_case`. This only applies to automatically retrieved term labels, *not* if term labels are provided by the `axis.labels`-argument.

Details

`get_model_data` simply calls `plot_model()` and returns the data from the ggplot-object. Hence, it is rather inefficient and should be used as alternative to **brooms** `tidy()`-function only in specific situations.

Some notes on the different plot-types:

`type = "std2"` Plots standardized beta values, 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. This standardization uses the `standardize`-function from the **arm**-package.

`type = "pred"` Plots marginal effects. Simply wraps `ggpredict`.

`type = "eff"` Plots marginal effects. Simply wraps `ggeffect`.

`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.

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.

Note

`plot_model()` replaces the functions `sjp.lm`, `sjp.glm`, `sjp.lmer`, `sjp.glmer` and `sjp.int`. These are becoming softly deprecated and will be removed in a future update.

References

Gelman A (2008) "Scaling regression inputs by dividing by two standard deviations." *Statistics in Medicine 27: 2865–2873.* http://www.stat.columbia.edu/~gelman/research/published/standardizing7.pdf

Aiken and West (1991). Multiple Regression: Testing and Interpreting Interactions.

Examples

```
# prepare data
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, 3, 4, 4))

# 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
library(lme4)
m <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
plot_model(m, type = "re")

# plot marginal effects
plot_model(m, type = "eff", 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)
```

`plot_models` *Forest plot of multiple regression models*

Description

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

Usage

```
plot_models(..., transform, std.est = NULL, 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,
  spacing = 0.4, colors = "Set1", show.values = FALSE,
  show.legend = TRUE, show.intercept = FALSE, show.p = TRUE,
  p.shape = FALSE, ci.lvl = 0.95, vline.color = NULL, digits = 2,
  grid = FALSE)
```

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` For linear models, choose whether standardized coefficients should be used for plotting. Default is no standardization.

`NULL` (default) no standardization, returns original estimates.

`"std"` standardized beta values.

`"std2"` standardized beta values, however, standardization is done by rescaling estimates by dividing them by two sd (see `std_beta`).

`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. Note that this argument does not apply to *Marginal Effects* plots.

`title` Character vector, used as plot title. By default, `get_dv_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 title for the plot legend. Note that only some plot types have legends (e.g. `type = "pred"` or when grouping estimates with `group.estimates`).

`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, `get_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.

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

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

`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 `display.brewer.all` to view all available palette names.
- If **wesanderson** is installed, you may also specify a name of a palette from that package.
- If **viridis** is installed, use `colors = "v"` to get the viridis color palette.
- There are some pre-defined color palettes in this package: `"aqua", "warm", "dust", "blambus",`
- 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`.

`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 `hdi` (High Density Interval) that is plotted. By default, `stanreg`-models are printed with two intervals: the "inner" interval, which defaults to the 50%-HDI; and the "outer" interval, which defaults to the 89%-HDI. `ci.lvl` affects only the outer interval in such cases. See `prob.inner` and `prob.outer` under the `...`-argument for more details.

`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.

Value

A ggplot-object.

Examples

```
data(efc)

# 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
)

# 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")
```

---

save_plot *Save ggplot-figure for print publication*

Description

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

Usage

```
save_plot(filename, fig = ggplot2::last_plot(), width = 12, height = 9,
  dpi = 300, theme = ggplot2::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)
```

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 `png`).

For adjusting plot appearance, see also `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.angle = NULL, 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,
  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 transparancy (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.tickscol`
: Color of axis tick marks

`axis.ticksmar`
: 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.backcol`
: Color of the diagram's background

`panel.bordercol`
: Color of whole diagram border (panel border)

`panel.col`
: Color of both diagram's border and background. If set, overrides both `panel.bordercol` and `panel.backcol`.

`panel.major.gridcol`
: Color of the major grid lines of the diagram background

`panel.minor.gridcol`
: Color of the minor grid lines of the diagram background

`panel.gridcol`
: Color for both minor and major grid lines of the diagram background. If set, overrides both `panel.major.gridcol` and `panel.minor.gridcol`.

`panel.gridcol.x`
: See `panel.gridcol`.

`panel.gridcol.y`
: See `panel.gridcol`.

`panel.major.linetype`
: line type for major grid lines

`panel.minor.linetype`
: line type for minor grid lines

`plot.backcol`
: Color of the plot's background

`plot.bordercol`
: Color of whole plot's border (panel border)

`plot.col`
: Color of both plot's region border and background. If set, overrides both `plot.backcol` and `plot.bordercol`.

`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

```
## 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)
sjp.xtab(efc$e42dep, efc$e16sex)

# Use classic-theme. you may need to
# load the ggplot2-library.
library(ggplot2)
set_theme(base = theme_classic())
sjp.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))
sjp.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"
)
sjp.frq(efc$e42dep)
## End(Not run)
```

`sjc.cluster` *Compute hierarchical or kmeans cluster analysis*

Description

Compute hierarchical or kmeans cluster analysis and return the group association for each observation as vector.

Usage

```
sjc.cluster(data, groupcount = NULL, method = c("hclust", "kmeans"),
  distance = c("euclidean", "maximum", "manhattan", "canberra", "binary",
  "minkowski"), agglomeration = c("ward", "ward.D", "ward.D2", "single",
  "complete", "average", "mcquitty", "median", "centroid"), iter.max = 20,
  algorithm = c("Hartigan-Wong", "Lloyd", "MacQueen"))
```

Arguments

`data`	A data frame with variables that should be used for the cluster analysis.

groupcount — Amount of groups (clusters) used for the cluster solution. May also be a set of initial (distinct) cluster centres, in case method = "kmeans" (see kmeans for details on centers argument). If groupcount = NULL and method = "kmeans", the optimal amount of clusters is calculated using the gap statistics (see sjc.kgap). For method = "hclust", groupcount needs to be specified. Following functions may be helpful for estimating the amount of clusters:

- Use sjc.elbow to determine the group-count depending on the elbow-criterion.
- If method = "kmeans", use sjc.kgap to determine the group-count according to the gap-statistic.
- If method = "hclust" (hierarchical clustering, default), use sjc.dend to inspect different cluster group solutions.
- Use sjc.grpdisc to inspect the goodness of grouping (accuracy of classification).

method — Method for computing the cluster analysis. By default ("kmeans"), a kmeans cluster analysis will be computed. Use "hclust" to compute a hierarchical cluster analysis. You can specify the initial letters only.

distance — Distance measure to be used when method = "hclust" (for hierarchical clustering). Must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist. If is method = "kmeans" this argument will be ignored.

agglomeration — Agglomeration method to be used when method = "hclust" (for hierarchical clustering). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). If method = "kmeans" this argument will be ignored. See 'Note'.

iter.max — Maximum number of iterations allowed. Only applies, if method = "kmeans". See kmeans for details on this argument.

algorithm — Algorithm used for calculating kmeans cluster. Only applies, if method = "kmeans". May be one of "Hartigan-Wong" (default), "Lloyd" (used by SPSS), or "MacQueen". See kmeans for details on this argument.

Value

The group classification for each observation as vector. This group classification can be used for sjc.grpdisc-function to check the goodness of classification. The returned vector includes missing values, so it can be appended to the original data frame data.

Note

Since R version > 3.0.3, the "ward" option has been replaced by either "ward.D" or "ward.D2", so you may use one of these values. When using "ward", it will be replaced by "ward.D2".

To get similar results as in SPSS Quick Cluster function, following points have to be considered:

1. Use the /PRINT INITIAL option for SPSS Quick Cluster to get a table with initial cluster centers.

2. Create a matrix of this table, by consecutively copying the values, one row after another, from the SPSS output into a matrix and specify nrow and ncol arguments.
3. Use algorithm="Lloyd".
4. Use the same amount of iter.max both in SPSS and this sjc.qclus.

This ensures a fixed initial set of cluster centers (as in SPSS), while kmeans in R always selects initial cluster sets randomly.

References

Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2014) cluster: Cluster Analysis Basics and Extensions. R package.

Examples

```
# Hierarchical clustering of mtcars-dataset
groups <- sjc.cluster(mtcars, 5)

# K-means clustering of mtcars-dataset
groups <- sjc.cluster(mtcars, 5, method="k")
```

---

sjc.dend *Compute hierarchical cluster analysis and visualize group classification*

Description

Computes a hierarchical cluster analysis and plots a hierarchical dendrogram with highlighted rectangles around the classified groups. Can be used, for instance, as visual tool to verify the elbow-criterion (see sjc.elbow).

Usage

```
sjc.dend(data, groupcount, distance = "euclidean", agglomeration = "ward")
```

Arguments

data	A data frame with variables that should be used for the cluster analysis.
groupcount	The amount of groups (clusters) that should be used. • Use sjc.elbow-function to determine the group-count depending on the elbow-criterion. • Use sjc.grpdisc-function to inspect the goodness of grouping (accuracy of classification). Solutions for multiple cluster groups can be plotted, for instance with "groupcount = c(3:6)".

distance	Distance measure to be used when method = "hclust" (for hierarchical clustering). Must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist. If is method = "kmeans" this argument will be ignored.
agglomeration	Agglomeration method to be used when method = "hclust" (for hierarchical clustering). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). If method = "kmeans" this argument will be ignored. See 'Note'.

Note

Since R version > 3.0.3, the "ward" option has been replaced by either "ward.D" or "ward.D2", so you may use one of these values. When using "ward", it will be replaced by "ward.D2".

Examples

```
# Plot dendrogram of hierarchical clustering of mtcars-dataset
# and show group classification
sjc.dend(mtcars, 5)

# Plot dendrogram of hierarchical clustering of mtcars-dataset
# and show group classification for 2 to 4 groups
sjc.dend(mtcars, 2:4)
```

---

sjc.elbow *Compute elbow values of a k-means cluster analysis*

Description

Plot elbow values of a k-means cluster analysis. This function computes a k-means cluster analysis on the provided data frame and produces two plots: one with the different elbow values and a second plot that maps the differences between each "step" (i.e. between elbow values) on the y-axis. An increase in the second plot may indicate the elbow criterion.

Usage

```
sjc.elbow(data, steps = 15, show.diff = FALSE)
```

Arguments

data	data frame containing all variables that should be used for determining the elbow criteria
steps	maximum group-count for the k-means cluster analysis for which the elbow-criterion should be displayed. Default is 15.
show.diff	logical, if TRUE, an additional plot with the differences between each fusion step of the Elbow criterion calculation is shown. This plot may help identifying the "elbow". Default for this argument is FALSE.

Examples

```
# plot elbow values of mtcars dataset
sjc.elbow(mtcars)
```

`sjc.grpdisc` *Compute a linear discriminant analysis on classified cluster groups*

Description

Computes linear discriminant analysis on classified cluster groups. This function plots a bar graph indicating the goodness of classification for each group.

Usage

```
sjc.grpdisc(data, groups, groupcount, clss.fit = TRUE, prnt.plot = TRUE)
```

Arguments

`data`	A data frame with variables that should be used for the cluster analysis.
`groups`	group classification of the cluster analysis that was returned from the `sjc.cluster`-function
`groupcount`	amount of groups (clusters) that should be used. Use `sjc.elbow` to determine the group-count depending on the elbow-criterion.
`clss.fit`	logical, if `TRUE` (default), a vertical line indicating the overall goodness of classification is added to the plot, so one can see whether a certain group is below or above the average classification goodness.
`prnt.plot`	logical, if `TRUE` (default), plots the results as graph. Use `FALSE` if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Invisibly) returns an object with

- `data`: the used data frame for plotting,
- `plot`: the ggplot object,
- `accuracy`: a vector with the accuracy of classification for each group,
- `total.accuracy`: the total accuracy of group classification.

Examples

```
# retrieve group classification from hierarchical cluster analysis
# on the mtcars data set (5 groups)
groups <- sjc.cluster(mtcars, 5)

# plot goodness of group classificatoin
sjc.grpdisc(mtcars, groups, 5)
```

sjc.kgap *Compute gap statistics for k-means-cluster*

Description

An implementation of the gap statistic algorithm from Tibshirani, Walther, and Hastie's "Estimating the number of clusters in a data set via the gap statistic". This function calls the clusGap-function of the **cluster**-package to calculate the data for the plot.

Usage

```
sjc.kgap(x, max = 10, B = 100, SE.factor = 1, method = "Tibs2001SEmax",
  plotResults = TRUE)
```

Arguments

x	matrix, where rows are observations and columns are individual dimensions, to compute and plot the gap statistic (according to a uniform reference distribution).
max	maximum number of clusters to consider, must be at least two. Default is 10.
B	integer, number of Monte Carlo ("bootstrap") samples. Default is 100.
SE.factor	[When method contains "SE"] Determining the optimal number of clusters, Tibshirani et al. proposed the "1 S.E."-rule. Using an SE.factor f, the "f S.E."-rule is used, more generally.
method	character string indicating how the "optimal" number of clusters, k^, is computed from the gap statistics (and their standard deviations), or more generally how the location k^ of the maximum of f[k] should be determined. Default is "Tibs2001SEmax". Possible value are: "globalmax" simply corresponds to the global maximum, i.e., is which.max(f). "firstmax" gives the location of the first local maximum. "Tibs2001SEmax" uses the criterion, Tibshirani et al(2001) proposed: "the smallest k such that f(k) >= f(k+1) - s_k+1". Note that this chooses k = 1 when all standard deviations are larger than the differences f(k+1) - f(k). "firstSEmax" is the location of the first f() value which is not larger than the first local maximum minus SE.factor * SE.f[], i.e, within an "f S.E." range of that maximum (see also SE.factor). "globalSEmax" (used in Dudoit and Fridlyand (2002), supposedly following Tibshirani's proposition) is the location of the first f() value which is not larger than the global maximum minus SE.factor * SE.f[], i.e, within an "f S.E." range of that maximum (see also SE.factor).
plotResults	logical, if TRUE (default), a graph visualiting the gap statistic will be plotted. Use FALSE to omit the plot.

Value

An object containing the used data frame for plotting, the ggplot object and the number of found cluster.

References

- Tibshirani R, Walther G, Hastie T (2001) Estimating the number of clusters in a data set via gap statistic. J. R. Statist. Soc. B, 63, Part 2, pp. 411-423
- Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K.(2013). cluster: Cluster Analysis Basics and Extensions. R package version 1.14.4. (web)

See Also

sjc.elbow

Examples

```
## Not run:
# plot gap statistic and determine best number of clusters
# in mtcars dataset
sjc.kgap(mtcars)

# and in iris dataset
sjc.kgap(iris[,1:4])
## End(Not run)
```

---

sjc.qclus *Compute quick cluster analysis*

Description

Compute a quick kmeans or hierarchical cluster analysis and displays "cluster characteristics" as plot.

Usage

```
sjc.qclus(data, groupcount = NULL, groups = NULL, method = c("kmeans",
  "hclust"), distance = c("euclidean", "maximum", "manhattan", "canberra",
  "binary", "minkowski"), agglomeration = c("ward", "ward.D", "ward.D2",
  "single", "complete", "average", "mcquitty", "median", "centroid"),
  iter.max = 20, algorithm = c("Hartigan-Wong", "Lloyd", "MacQueen"),
  show.accuracy = FALSE, title = NULL, axis.labels = NULL,
  wrap.title = 40, wrap.labels = 20, wrap.legend.title = 20,
  wrap.legend.labels = 20, facet.grid = FALSE, geom.colors = "Paired",
  geom.size = 0.5, geom.spacing = 0.1, show.legend = TRUE,
  show.grpcnt = TRUE, legend.title = NULL, legend.labels = NULL,
  coord.flip = FALSE, reverse.axis = FALSE, prnt.plot = TRUE)
```

Arguments

data	A data frame with variables that should be used for the cluster analysis.
groupcount	Amount of groups (clusters) used for the cluster solution. May also be a set of initial (distinct) cluster centres, in case method = "kmeans" (see kmeans for details on centers argument). If groupcount = NULL and method = "kmeans", the optimal amount of clusters is calculated using the gap statistics (see sjc.kgap). For method = "hclust", groupcount needs to be specified. Following functions may be helpful for estimating the amount of clusters: • Use sjc.elbow to determine the group-count depending on the elbow-criterion. • If method = "kmeans", use sjc.kgap to determine the group-count according to the gap-statistic. • If method = "hclust" (hierarchical clustering, default), use sjc.dend to inspect different cluster group solutions. • Use sjc.grpdisc to inspect the goodness of grouping (accuracy of classification).
groups	Optional, by default, this argument is NULL and will be ignored. However, to plot existing cluster groups, specify groupcount and groups. groups is a vector of same length as nrow(data) and indicates the group classification of the cluster analysis. The group classification can be computed with the sjc.cluster function. See 'Examples'.
method	Method for computing the cluster analysis. By default ("kmeans"), a kmeans cluster analysis will be computed. Use "hclust" to compute a hierarchical cluster analysis. You can specify the initial letters only.
distance	Distance measure to be used when method = "hclust" (for hierarchical clustering). Must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist. If is method = "kmeans" this argument will be ignored.
agglomeration	Agglomeration method to be used when method = "hclust" (for hierarchical clustering). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). If method = "kmeans" this argument will be ignored. See 'Note'.
iter.max	Maximum number of iterations allowed. Only applies, if method = "kmeans". See kmeans for details on this argument.
algorithm	Algorithm used for calculating kmeans cluster. Only applies, if method = "kmeans". May be one of "Hartigan-Wong" (default), "Lloyd" (used by SPSS), or "MacQueen". See kmeans for details on this argument.
show.accuracy	Logical, if TRUE, the sjc.grpdisc function will be called, which computes a linear discriminant analysis on the classified cluster groups and plots a bar graph indicating the goodness of classification for each 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.

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.

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.

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

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

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

show.grpcnt — Logical, if TRUE (default), the count within each cluster group is added to the legend labels (e.g. "Group 1 (n=87)").

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

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

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

reverse.axis — Logical, if TRUE, the values on the x-axis are reversed.

prnt.plot — logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Details

Following steps are computed in this function:

1. If method = "kmeans", this function first determines the optimal group count via gap statistics (unless argument groupcount is specified), using the sjc.kgap function.
2. A cluster analysis is performed by running the sjc.cluster function to determine the cluster groups.
3. Then, all variables in data are scaled and centered. The mean value of these z-scores within each cluster group is calculated to see how certain characteristics (variables) in a cluster group differ in relation to other cluster groups.
4. These results are plotted as graph.

This method can also be used to plot existing cluster solution as graph witouth computing a new cluster analysis. See argument groups for more details.

Value

(Invisibly) returns an object with

- `data`: the used data frame for plotting,
- `plot`: the ggplot object,
- `groupcount`: the number of found cluster (as calculated by `sjc.kgap`)
- `classification`: the group classification (as calculated by `sjc.cluster`), including missing values, so this vector can be appended to the original data frame.
- `accuracy`: the accuracy of group classification (as calculated by `sjc.grpdisc`).

Note

See 'Note' in `sjc.cluster`

References

Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2014) cluster: Cluster Analysis Basics and Extensions. R package.

Examples

```
## Not run:
# k-means clustering of mtcars-dataset
sjc.qclus(mtcars)

# k-means clustering of mtcars-dataset with 4 pre-defined
# groups in a faceted panel
sjc.qclus(airquality, groupcount = 4, facet.grid = TRUE)
## End(Not run)

# k-means clustering of airquality data
# and saving the results. most likely, 3 cluster
# groups have been found (see below).
airgrp <- sjc.qclus(airquality)

# "re-plot" cluster groups, without computing
# new k-means cluster analysis.
sjc.qclus(airquality, groupcount = 3, groups = airgrp$classification)
```

---

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 = 0.1, show.p = TRUE,
  show.summary = FALSE, prnt.plot = TRUE)
```

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 (see also sjt.grpmean). 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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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.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 sjp.glm), 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 sjp.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 and 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 excact test (see fisher.test) is computed instead of chi-squared test. If the table's matrix is larger than 2x2, Fisher's excact test with Monte Carlo simulation is computed.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) 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).

See Also

sjt.grpmean

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, prnt.plot = TRUE)
```

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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) returns the ggplot-object with the complete plot (plot) as well as the data frame that was used for setting up the ggplot-object (mydf).

See Also

Tales of R.

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)
```

sjp.corr *Plot correlation matrix*

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, prnt.plot = 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.
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 "spearman" (default), "pearson" or "kendall". May be abbreviated.
geom.colors	user defined color for geoms. See 'Details' in sjp.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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

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

(Insisibily) 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.

See Also

sjt.corr

Examples

```
# create data frame with 5 random variables
mydf <- data.frame(cbind(runif(10), runif(10), runif(10),
                         runif(10), runif(10)))

# plot correlation matrix
sjp.corr(mydf)

# ------------------------------
# Data from the EUROFAMCARE sample dataset
# ------------------------------
library(sjlabelled)
data(efc)

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

# create data frame
vars.index <- c(1, 4, 15, 19, 20, 21, 22, 24, 25)
mydf <- data.frame(efc[, vars.index])
colnames(mydf) <- varlabs[vars.index]

# show legend
```

```
sjp.corr(mydf, show.legend = TRUE)

# ------------------------------
# auto-detection of labels
# ------------------------------
sjp.corr(efc[, vars.index])
```

`sjp.fa` *Plot FA results*

Description

Performes a maximum likelihood factor analysis on a data frame or matrix and plots the factor solution as ellipses or tiles.
In case a data frame is used as argument, 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

```
sjp.fa(data, rotation = c("promax", "varimax"), method = c("ml", "minres",
  "wls", "gls", "pa", "minchi", "minrank"), nmbr.fctr = NULL,
  fctr.load.tlrn = 0.1, digits = 2, title = NULL, axis.labels = NULL,
  type = c("bar", "circle", "tile"), geom.size = 0.6,
  geom.colors = "RdBu", wrap.title = 50, wrap.labels = 30,
  show.values = TRUE, show.cronb = TRUE, prnt.plot = TRUE)
```

Arguments

`data`	A data frame that should be used to compute a FA, or a `fa` object.
`rotation`	Rotation of the factor loadings. May be `"varimax"` for orthogonal rotation or `"promax"` for oblique transformation (default). Requires the `"GPArotation"` package.
`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.
`nmbr.fctr`	Number of factors used for calculating the rotation. By default, this value is `NULL` and the amount of factors is calculated according to a parallel analysis.

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
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
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.
type	Plot type resp. geom type. May be one of following: `"circle"` or `"tile"` circular or tiled geoms, or `"bar"` for a bar plot. You may use initial letter only for this argument.
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 sjp.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.values	Logical, whether values should be plotted or not.
show.cronb	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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Invisibly) returns a structure with

- the rotated factor loading matrix (rotate)
- the column indices of removed variables (for more details see next list item) (removed.colindex)
- an updated data frame containing all factors that have a clear loading on a specific scale in case data was a data frame (See argument fctr.load.tlrn for more details) (removed.df)
- the factor.index, i.e. the column index of each variable with the highest factor loading for each factor,
- the ggplot-object (plot),
- the data frame that was used for setting up the ggplot-object (df).

Note

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

See Also

- sjPlot manual: sjp.pca
- sjt.pca

Examples

```
library(GPArotation)
data(efc)
# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# recveive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

# use data frame as argument, let sjp.fa() compute FA
sjp.fa(efc[, start:end])
sjp.fa(efc[, start:end], type = "tile")
```

---

sjp.frq *Plot frequencies of variables*

Description

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

Usage

```
sjp.frq(var.cnt, 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,
  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, prnt.plot = TRUE)
```

Arguments

var.cnt	Vector of counts, for which frequencies or means will be plotted or printed.
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.
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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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.

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 insie 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
: Transparancy 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).

prnt.plot
: logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) returns the ggplot-object with the complete plot (plot) as well as the data frame that was used for setting up the ggplot-object (data).

Note

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

See Also

- sjPlot manual: sjp.frq
- sjt.frq

Examples

```
library(sjlabelled)
data(efc)

# boxplot
sjp.frq(efc$e17age, type = "box")

# histogram
sjp.frq(efc$e17age, type = "hist", show.mean = TRUE)

# violin plot
sjp.frq(efc$e17age, type = "v")
```

```
# bar plot
sjp.frq(efc$e42dep)

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

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

# Simulate ggplot-default histogram
sjp.frq(efc$c160age, type = "h", geom.size = 3)

# histogram with overlayed normal curve
sjp.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))
```

---

sjp.glm *Plot estimates, predictions or effects of generalized linear models*

Description

Plot odds or incident rate ratios with confidence intervalls as dot plot. Depending on the type argument, this function may also plot model assumptions for generalized linear models, or marginal effects (predicted probabilities or events).

Usage

```
sjp.glm(fit, type = "dots", vars = NULL, group.estimates = NULL,
  remove.estimates = NULL, sort.est = TRUE, title = NULL,
  legend.title = NULL, axis.labels = NULL, axis.title = NULL,
  geom.size = NULL, geom.colors = "Set1", wrap.title = 50,
  wrap.labels = 25, axis.lim = NULL, grid.breaks = 0.5,
  trns.ticks = TRUE, show.intercept = FALSE, show.values = TRUE,
  show.p = TRUE, show.ci = FALSE, show.legend = FALSE,
  show.summary = FALSE, show.scatter = TRUE, point.alpha = 0.2,
  point.color = NULL, jitter.ci = FALSE, digits = 2, vline.type = 2,
  vline.color = "grey70", coord.flip = TRUE, y.offset = 0.15,
  facet.grid = TRUE, prnt.plot = TRUE, ...)
```

Arguments

fit	Fitted generalized linear model (glm- or logistf-object).
type	Type of plot. Use one of following: "dots" (or "glm" or "or" (default)) for odds or incident rate ratios (forest plot). Note that this type plots the exponentiated estimates, thus being appropriate only for specific link-functions. "eff" to plot marginal effects of predicted probabilities or incidents for each model term, where all remaining co-variates are set to the mean (see 'Details'). Use facet.grid to decide whether to plot each coefficient as separate plot or as integrated faceted plot. "pred" to plot predicted values for the response, related to specific model predictors. See 'Details'. "ma" to check model assumptions. Note that the only relevant argument for this option is fit. All other arguments are ignored. "vif" to plot Variance Inflation Factors.
vars	Numeric vector with column indices of selected variables or a character vector with variable names of selected variables from the fitted model, which should be used to plot - depending on type - estimates, fixed effects slopes or predicted values (mean, probabilities, incidents rates, ...). See 'Examples'.
group.estimates	Numeric or character vector, indicating a group identifier for each estimate. Dots and confidence intervals of estimates are coloured according to their group association. See 'Examples'.
remove.estimates	Character vector with coefficient names that indicate which estimates should be removed from the plot. remove.estimates = "est_name" would remove the estimate *est_name*. Default is NULL, i.e. all estimates are printed.
sort.est	Logical, determines whether estimates should be sorted according to their values. If group.estimates is *not* NULL, estimates are sorted according to their group assignment.
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. Note that only some plot types have legends (e.g. type = "pred" or when grouping estimates with group.estimates).
axis.labels	character vector with labels used as axis labels. Optional argument, since in most cases, axis labels are set automatically.
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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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 = ...)

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 palette for geoms. If group.estimates is *not* specified, must either be vector with two color values or a specific color palette code (see 'Details' in sjp.grpfrq). Else, if group.estimates is specified, geom.colors must be a vector of same length as groups. See 'Examples'.
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, or both. For multiple plot outputs (e.g., from type = "eff" or type = "slope" in sjp.glm), 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.
trns.ticks	Logical, if TRUE, the grid lines have exponential distances (equidistant), i.e. they visually have the same distance from one panel grid to the next. If FALSE, grids are plotted on every grid.breaks's position, thus the grid lines become narrower with higher odds ratio values.
show.intercept	Logical, if TRUE, the intercept of the fitted model is also plotted. Default is FALSE. For glm's, please note that due to exponential transformation of estimates, the intercept in some cases can not be calculated, thus the function call is interrupted and no plot printed.
show.values	Logical, whether values should be plotted or not.
show.p	Logical, adds significance levels to values, or value and variable labels.
show.ci	Logical, if TRUE, depending on type, a confidence interval or region is added to the plot. For frequency plots, the confidence interval for the relative frequencies are shown.
show.legend	logical, if TRUE, and depending on plot type and function, a legend is added to the plot.
show.summary	Logical, if TRUE, a summary with model statistics is added to the plot.
show.scatter	Logical, if TRUE (default), adds a scatter plot of data points to the plot. Only applies for slope-type or predictions plots. For most plot types, dots are jittered to avoid overplotting, hence the points don't reflect exact values in the data.
point.alpha	Alpha value of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
point.color	Color of of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
jitter.ci	Logical, if TRUE and show.ci = TRUE and confidence bands are displayed as error bars, adds jittering to lines and error bars to avoid overlapping.
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
vline.type	Linetype of the vertical "zero point" line. Default is 2 (dashed line).

vline.color	Color of the vertical "zero point" line. Default value is "grey70".
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).
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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.
...	Other arguments passed down to further functions. Currently, following arguments are supported: ?effects::effect Any arguments accepted by the effect resp. allEffects function, for type = "eff". width The width-argument for error bars. alpha The alpha-argument for confidence bands. level The level-argument confidence bands.

Details

type = "slope" the predicted values are based on the intercept's estimate and each specific term's estimate. All other co-variates are set to zero (i.e. ignored), which corresponds to family(fit)$linkinv(eta = b0 + b (where xi is the estimate). This plot type can be seen as equivalent to type = "slope" for sjp.lm, just for glm objects. This plot type may give similar results as type = "pred", however, type = "slope" does not adjust for other predictors.

type = "eff" computes marginal effects of all higher order terms in the model. The predicted values computed by type = "eff" are adjusted for all other co-variates, by setting them to the mean (as returned by the allEffects function). You can pass further arguments down to allEffects for flexible function call via the ...-argument.

type = "pred" the predicted values of the response are computed, based on the predict.glm method. Corresponds to predict(fit, type = "response"). This plot type requires the vars argument to select specific terms that should be used for the x-axis and - optional - as grouping factor. Hence, vars must be a character vector with the names of one or two model predictors. See 'Examples'.

Value

(Insisibily) returns, depending on the plot type

- The ggplot-object (plot). For multiple plots and if facet.grid = FALSE) a plot.list is returned.
- A data frame data with the data used to build the ggplot-object(s), or a list of data frames (data.list).

See Also

sjPlot manual: sjp.glm

Examples

```
# prepare dichotomous dependent variable
swiss$y <- ifelse(swiss$Fertility < median(swiss$Fertility), 0, 1)

# fit model
fitOR <- glm(y ~ Education + Examination + Infant.Mortality + Catholic,
             family = binomial(link = "logit"), data = swiss)

# print Odds Ratios as dots
sjp.glm(fitOR)

# ------------------------------
# Predictors for negative impact of care. Data from
# the EUROFAMCARE sample dataset
# ------------------------------
library(sjmisc)
library(sjlabelled)
data(efc)
# create binary response
y <- ifelse(efc$neg_c_7 < median(na.omit(efc$neg_c_7)), 0, 1)
# create data frame for fitted model
mydf <- data.frame(y = as.factor(y),
                   sex = to_factor(efc$c161sex),
                   dep = to_factor(efc$e42dep),
                   barthel = efc$barthtot,
                   education = to_factor(efc$c172code))
# fit model
fit <- glm(y ~., data = mydf, family = binomial(link = "logit"))

# plot odds ratios
sjp.glm(fit, title = get_label(efc$neg_c_7))

# plot probability curves (relationship between predictors and response)
sjp.glm(fit, title = get_label(efc$neg_c_7), type = "slope")

# --------------------------
# grouping estimates
# --------------------------
sjp.glm(fit,  group.estimates = c(1, 2, 2, 2, 3, 4, 4))

# --------------------------
# model predictions, with selected model terms.
# 'vars' needs to be a character vector of length 1 to 3
# with names of model terms for x-axis and grouping factor.
# --------------------------
sjp.glm(fit, type = "pred", vars = "barthel")
# faceted, with ci
sjp.glm(fit, type = "pred", vars = c("barthel", "dep"), show.ci = TRUE)
# w/o facets
sjp.glm(fit, type = "pred", vars = c("barthel", "dep"), facet.grid = FALSE)
# with third grouping variable - this type automatically uses grid layout
sjp.glm(fit, type = "pred", vars = c("barthel", "sex", "education"))
```

`sjp.glmer` *Plot estimates, predictions or effects of generalized linear mixed effects models*

Description

By default, this function plots estimates (odds, risk or incidents ratios, i.e. exponentiated coefficients, depending on family and link function) with confidence intervals of either fixed effects or random effects of generalized linear mixed effects models (that have been fitted with the `glmer`-function of the **lme4**-package). Furthermore, this function also plots predicted probabilities / incidents or diagnostic plots.

Usage

```
sjp.glmer(fit, type = "re", vars = NULL, ri.nr = NULL,
  group.estimates = NULL, remove.estimates = NULL, emph.grp = NULL,
  sample.n = NULL, sort.est = NULL, title = NULL, legend.title = NULL,
  axis.labels = NULL, axis.title = NULL, geom.colors = "Set1",
  geom.size = NULL, show.values = TRUE, show.p = TRUE, show.ci = FALSE,
  show.legend = FALSE, show.intercept = FALSE,
  string.interc = "(Intercept)", show.scatter = TRUE, point.alpha = 0.2,
  point.color = NULL, jitter.ci = FALSE, fade.ns = FALSE,
  axis.lim = NULL, digits = 2, vline.type = 2, vline.color = "grey70",
  facet.grid = TRUE, free.scale = FALSE, y.offset = 0.1,
  prnt.plot = TRUE, ...)
```

Arguments

`fit` A fitted model as returned by the `glmer`-function.

`type` Type of plot. Use one of following:

- `"re"` (default) for conditional modes (odds or incidents ratios) of random effects
- `"fe"` for odds or incidents ratios of fixed effects
- `"fe.cor"` for correlation matrix of fixed effects
- `"re.qq"` for a QQ-plot of random effects (random effects quantiles against standard normal quantiles)
- `"ri.slope"` to plot probability or incidents curves (predicted probabilities or incidents) of random intercept variances for all fixed effects coefficients. Use `facet.grid` to decide whether to plot each coefficient as separate plot or as integrated faceted plot. See 'Details'.
- `"rs.ri"` for fitted probability curves (predicted probabilities) indicating the random slope-intercept pairs. Use this to visualize the random parts of random slope-intercept (or repeated measure) models. When having too many groups, use `sample.n` argument.

"eff" to plot marginal effects of predicted probabilities or incidents for each fixed term, where remaining co-variates are set to the mean. Use facet.grid to decide whether to plot each coefficient as separate plot or as integrated faceted plot. See 'Details'.

"pred" to plot predicted probabilities or incidents for the response, related to specific model predictors and conditioned on random effects. See 'Details'.

"pred.fe" to plot predicted probabilities or incidents for the response, related to specific model predictors, only for fixed effects. See 'Details'.

"ma" to check model assumptions. Note that only argument fit applies to this plot type. All other arguments are ignored.

vars Numeric vector with column indices of selected variables or a character vector with variable names of selected variables from the fitted model, which should be used to plot - depending on type - estimates, fixed effects slopes or predicted values (mean, probabilities, incidents rates, ...). See 'Examples'.

ri.nr Numeric vector. If type = "re" or type = "ri.slope", 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.

group.estimates
Numeric or character vector, indicating a group identifier for each estimate. Dots and confidence intervals of estimates are coloured according to their group association. See 'Examples'.

remove.estimates
Character vector with coefficient names that indicate which estimates should be removed from the plot. remove.estimates = "est_name" would remove the estimate *est_name*. Default is NULL, i.e. all estimates are printed.

emph.grp Numeric vector with index numbers of grouping levels (from random effect). If type = "ri.slope" and facet.grid = FALSE, an integrated plot of predicted probabilities of fixed effects resp. fixed effects slopes for each grouping level is plotted. To better find certain groups, use this argument to emphasize these groups in the plot. See 'Examples'.

sample.n Numeric vector. only applies, if type = "rs.ri". If plot has many random intercepts (grouping levels), overplotting of regression lines may occur. In this case, consider random sampling of grouping levels. If sample.n is of length 1, a random sample of sample.n observation is selected to plot random intercepts. If sample.n is of length > 1, random effects indicated by the values in sample.n are selected to plot random effects. Use the latter option to always select a fixed, identical set of random effects for plotting (useful when ecomparing multiple models).

sort.est Determines in which way estimates are sorted in the plot:

- If NULL (default), no sorting is done and estimates are sorted in order of model coefficients.
- If sort.est = "sort.all", estimates are re-sorted for each coefficient (only applies if type = "re" and facet.grid = FALSE), i.e. the estimates of the random effects for each predictor are sorted and plotted to an own plot.

- If `type = "fe"` or `type = "fe.std"`, `TRUE` will sort estimates
- If `type = "re"`, specify a predictor's / coefficient's name to sort estimates according to this coefficient.

See 'Examples'.

`title` Character vector with one or more labels that are used as plot title.

`legend.title` Character vector, used as title for the plot legend. Note that only some plot types have legends (e.g. `type = "pred"` or when grouping estimates with `group.estimates`).

`axis.labels` Character vector with labels for the model terms, used as axis labels. For mixed models, should either be vector of fixed effects variable labels (if `type = "fe"` or `type = "fe.std"`) or a vector of group (value) labels from the random intercept's categories (if `type = "re"`).

`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. To set multiple axis titles (e.g. with `type = "eff"` for many predictors), `axis.title` must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for `type = "eff"`). **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 = ...)`

`geom.colors` User defined color palette for geoms. If `group.estimates` is *not* specified, must either be vector with two color values or a specific color palette code (see 'Details' in `sjp.grpfrq`). Else, if `group.estimates` is specified, `geom.colors` must be a vector of same length as groups. See 'Examples'.

`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.

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

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

`show.ci` Logical, if `TRUE`, depending on `type`, a confidence interval or region is added to the plot. For frequency plots, the confidence interval for the relative frequencies are shown.

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

`show.intercept` Logical, if `TRUE`, the intercept of the fitted model is also plotted. Default is `FALSE`. For `glm`'s, please note that due to exponential transformation of estimates, the intercept in some cases can not be calculated, thus the function call is interrupted and no plot printed.

`string.interc` String, axis label of intercept estimate. Only applies, if `show.intercept = TRUE` and `axis.labels` is not `NULL`.

`show.scatter` Logical, if `TRUE` (default), adds a scatter plot of data points to the plot. Only applies for slope-type or predictions plots. For most plot types, dots are jittered to avoid overplotting, hence the points don't reflect exact values in the data.

`point.alpha` Alpha value of point-geoms in the scatter plots. Only applies, if `show.scatter = TRUE`.

point.color — Color of of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.

jitter.ci — Logical, if TRUE and show.ci = TRUE and confidence bands are displayed as error bars, adds jittering to lines and error bars to avoid overlapping.

fade.ns — Logical, if TRUE, non significant estimates will be printed in slightly faded colors.

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 sjp.glm), axis.lim may also be a list of vectors of length 2, defining axis limits for each plot (only if non-faceted).

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

vline.type — Linetype of the vertical "zero point" line. Default is 2 (dashed line).

vline.color — Color of the vertical "zero point" line. Default value is "grey70".

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.

free.scale — Logical, if TRUE and facet.grid = TRUE, each facet grid gets its own fitted scale. If free.scale = FALSE, each facet in the grid has the same scale range.

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

prnt.plot — logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

... — Other arguments passed down to further functions. Currently, following arguments are supported:

?effects::effect — Any arguments accepted by the effect resp. allEffects function, for type = "eff".

width — The width-argument for error bars.

alpha — The alpha-argument for confidence bands.

level — The level-argument confidence bands.

Details

type = "re" plots the conditional modes of the random effects, incldluing predicion intervals. It basically does the same as dotplot(exp(ranef(fit, condVar = TRUE)[[i]]), where i denotes the random effect index.

type = "fe.slope" the predicted values are based on the fixed effects intercept's estimate and each specific fixed term's estimate. All other fixed effects are set to zero (i.e. ignored), which corresponds to family(fit)$linkinv(eta = b0 + bi * xi) (where xi is the estimate of fixed effects and b0 is the intercept of the fixed effects; the inverse link-function is used). This plot type may give similar results as type = "pred", however, type = "fe.slope" does not adjust for other predictors.

type = "eff" plots the marginal effects of model predictors. Unlike type = "fe.slope", the predicted values computed by type = "eff" are adjusted for all co-variates, which are set to the mean, as returned by the allEffects function. You can pass further arguments down to allEffects for flexible function call via the ...-argument.

`type = "ri.slope"` the predicted values are based on the fixed effects intercept, plus each random intercept and each specific fixed term's estimate. All other fixed effects are set to zero (i.e. ignored), which corresponds to `family(fit)$linkinv(eta = b0 + b0[r1-rn] + bi * xi)` (where `xi` is the estimate of fixed effects, `b0` is the intercept of the fixed effects and `b0[r1-rn]` are all random intercepts).

`type = "rs.ri"` the predicted values are based on the fixed effects intercept, plus each random intercept and random slope. This plot type is intended to plot the random part, i.e. the predicted probabilities or incident rates of each random slope for each random intercept. Since the random intercept specifies the deviance from the gloabl intercept, the global intercept is always included. In case of overplotting, use the `sample.n` argument to randomly sample a limited amount of groups.

`type = "coef"` forest plot of joint fixed and random effect coefficients, as retrieved by `coef.merMod`, it's simply `ranef + fixef`.

`type = "pred"` **or** `type = "pred.fe"` predicted values against response, only fixed effects or conditional on random intercept. It's calling `predict(fit, type = "response", re.form = NA)` resp. `predict(fit, type = "response", re.form = NULL)` to compute the values. This plot type requires the `vars` argument to select specific terms that should be used for the x-axis and - optional - as grouping factor. Hence, `vars` must be a character vector with the names of one or two model predictors. See 'Examples'.

Value

(Insisibily) returns, depending on the plot type

- The ggplot-object (`plot`). For multiple plots and if `facet.grid = FALSE`) a `plot.list` is returned.
- A data frame `data` with the data used to build the ggplot-object(s), or a list of data frames (`data.list`).

Note

- Computation of p-values (if necessary) is based on normal- distribution assumption, treating the t-statistics as Wald z-statistics.
- Plot types use the inverse link-function to calculate predicted probabilites or incidents rates. Thus, this function should work with different model families and link functions; however, the plot or axis title may not use the exact terminology regarding model family or link function.
- Thanks go to Robert Reijntjes from Leiden University Medical Center for sharing R code that is used to compute fixed effects correlation matrices and qq-plots of random effects.

See Also

sjPlot manual: sjp.glmer

Examples

```
library(lme4)
library(sjmisc)
library(sjlabelled)
```

```
# create binary response
sleepstudy$Reaction.dicho <- dicho(sleepstudy$Reaction, dich.by = "median")
# fit model
fit <- glmer(Reaction.dicho ~ Days + (Days | Subject),
             data = sleepstudy, family = binomial("logit"))

# simple plot
sjp.glmer(fit)

# sort by predictor Days
sjp.glmer(fit, sort.est = "Days")

## Not run:
data(efc)
# create binary response
efc$hi_qol <- dicho(efc$quol_5)
# prepare group variable
efc$grp = as.factor(efc$e15relat)
levels(x = efc$grp) <- get_labels(efc$e15relat)
# data frame for fitted model
mydf <- data.frame(hi_qol = to_factor(efc$hi_qol),
                   sex = to_factor(efc$c161sex),
                   education = to_factor(efc$c172code),
                   c12hour = efc$c12hour,
                   neg_c_7 = efc$neg_c_7,
                   grp = efc$grp)

# fit glmer, with categorical predictor with more than 2 levels
fit <- glmer(hi_qol ~ sex + education + c12hour + neg_c_7 + (1|grp),
             data = mydf, family = binomial("logit"))

# plot and sort fixed effects, axis labels automatically retrieved
sjp.glmer(fit, type = "fe", sort.est = TRUE)

# plot probability curves (predicted probabilities)
# for each covariate, grouped by random intercepts
# in integrated plots, emphasizing groups 1 and 4
sjp.glmer(fit, type = "ri.slope", emph.grp = c(1, 4), facet.grid = FALSE)

# plot predicted probabilities for response,
# non faceted, with ci
sjp.glmer(fit, type = "pred.fe", vars = c("neg_c_7", "education"),
          show.ci = TRUE, facet.grid = FALSE)

# predictions by gender and education
sjp.glmer(fit, type = "pred.fe", vars = c("neg_c_7", "sex", "education"))
## End(Not run)
```

---

sjp.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 groups.

Usage

```
sjp.gpt(x, y, groups, geom.colors = "Set1", 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, prnt.plot = TRUE)
```

Arguments

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, dichtomized at the highest category and all remaining categories.
groups	Grouping variable, which will define the y-axis
geom.colors	user defined color for geoms. See 'Details' in sjp.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.
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. 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.
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 title for the plot 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 `sjp.glm`), `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 `groups` 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.

prnt.plot
: logical, if `TRUE` (default), plots the results as graph. Use `FALSE` if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Details

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

Value

(Insisibily) 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`).

Examples

```
data(efc)

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

# proportion of educational levels in highest
# dependency category of elderly, for different
# care levels
sjp.gpt(efc$c172code, efc$e42dep, efc$n4pstu)
```

sjp.grpfrq *Plot grouped or stacked frequencies*

Description

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

Usage

```
sjp.grpfrq(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, 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,
  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", prnt.plot = TRUE)
```

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 sjp.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 excact test (see fisher.test) is computed instead of chi-squared test. If the table's matrix is larger than 2x2, Fisher's excact test with Monte Carlo simulation is computed.

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 insie 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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

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 (like sjp.int() or sjp.glm(type = "pred")), 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 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")).

Value

(Insisibily) 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).

See Also

sjPlot manual: sjp.grpfrq

Examples

```
data(efc)
sjp.grpfrq(efc$e17age, efc$e16sex, show.values = FALSE)

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

# grouped bars
sjp.grpfrq(efc$e42dep, efc$e16sex, title = NULL)

# box plots with interaction variable
sjp.grpfrq(efc$e17age, efc$e42dep, intr.var = efc$e16sex, type = "box")
```

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

# same data as line plot
sjp.grpfrq(efc$neg_c_7, efc$e42dep, type = "line")
```

`sjp.int` *Plot interaction effects of (generalized) linear (mixed) models*

Description

Plot regression (predicted values) or probability lines (predicted probabilities) of significant interaction terms to better understand effects of moderations in regression models. This function accepts following fitted model classes:

- linear models (`lm`)
- generalized linear models (`glm`)
- linear mixed effects models (`lmer`)
- generalized linear mixed effects models (`glmer`)
- non-linear mixed effects models (`nlmer`)
- linear mixed effects models (`lme`, but only for `type = "eff"`)
- generalized least squares models (`gls`, but only for `type = "eff"`)
- panel data estimators (`plm`)

Note that beside interaction terms, also the single predictors of each interaction (main effects) must be included in the fitted model as well. Thus, `lm(dep ~ pred1 * pred2)` will work, but `lm(dep ~ pred1:pred2)` won't!

Usage

```
sjp.int(fit, type = c("eff", "cond"), int.term = NULL,
  int.plot.index = NULL, mdrt.values = c("minmax", "meansd", "zeromax",
  "quart", "all"), swap.pred = FALSE, plevel = 0.1, diff = FALSE,
  title = NULL, axis.title = NULL, legend.title = NULL,
  legend.labels = NULL, wrap.title = 50, wrap.legend.labels = 20,
  wrap.legend.title = 20, geom.colors = "Set1", geom.size = NULL,
  fill.color = "grey", fill.alpha = 0.3, show.values = FALSE,
  show.ci = FALSE, jitter.ci = FALSE, p.kr = TRUE, grid.breaks = NULL,
  xlim = NULL, ylim = NULL, y.offset = 0.07, digits = 2,
  facet.grid = FALSE, prnt.plot = TRUE, ...)
```

Arguments

`fit` A fitted (generalized) linear (mixed) model object, including interaction terms. Accepted model classes are

- linear models (`lm`)
- generalized linear models (`glm`)
- linear mixed effects models (`lmer`)
- generalized linear mixed effects models (`glmer`)
- non-linear mixed effects models (`nlmer`)
- linear mixed effects models (`lme`, but only for `type = "eff"`)
- generalized least squares models (`gls`, but only for `type = "eff"`)
- panel data estimators (`plm`)

`type` Interaction plot type. Use one of following values:

`type = "eff"` (default) plots the overall moderation effect on the response value. See 'Details'.

`type = "cond"` plots the mere *change* of the moderating effect on the response value (conditional effect). See 'Details'.

`int.term` Name of interaction term of `fit` (as character), which should be plotted when using `type = "eff"`. By default, this argument will be ignored (i.e. `int.term = NULL`). See 'Details'.

`int.plot.index` Numeric vector with index numbers that indicate which interaction terms should be plotted in case the `fit` has more than one interaction. By default, this value is `NULL`, hence all interactions are plotted.

`mdrt.values` Indicates which values of the moderator variable should be used when plotting the interaction effects.

`"minmax"` (default) minimum and maximum values (lower and upper bounds) of the moderator are used to plot the interaction between independent variable and moderator.

`"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, 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. Note that this option only applies to `type = "eff"`, for numeric moderator values.

swap.pred — Logical, if TRUE, the predictor on the x-axis and the moderator value in an interaction are swapped. For type = "eff", the first interaction term is used as moderator and the second term is plotted at the x-axis. For type = "cond", the interaction's predictor with less unique values is printed along the x-axis. Default is FALSE, so the second predictor in an interaction, respectively the predictor with more unique values is printed along the x-axis.

plevel — Numeric, indicates at which p-value an interaction term is considered as *significant*, i.e. at which p-level an interaction term will be considered for plotting. Default is 0.1 (10 percent), hence, non-significant interactions are excluded by default. This argument does not apply to type = "eff".

diff — Logical, if FALSE (default), the minimum and maximum interaction effects of the moderating variable is shown (one line each). if TRUE, only the difference between minimum and maximum interaction effect is shown (single line). Only applies to type = "cond".

title — Default title used for the plots. Should be a character vector of same length as interaction plots to be plotted. Default value is NULL, which means that each plot's title includes the dependent variable as well as the names of the interaction terms.

axis.title — Default title used for the x-axis. Should be a character vector of same length as interaction plots to be plotted. Default value is NULL, which means that each plot's x-axis uses the predictor's name as title.

legend.title — Title of the plot's legend. A character vector of same length as amount of interaction plots to be plotted (i.e. one vector element for each plot's legend title).

legend.labels — Labels for the guide/legend. Either a character vector of same length as amount of legend labels of the plot, or a list of character vectors, if more than one interaction plot is plotted (i.e. one vector of legend labels for each interaction plot). Default is NULL, so the name of the predictor with min/max-effect is used as legend label.

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.labels — numeric, determines how many chars of the legend 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.

geom.colors — Vector of color values or name of a valid color brewer palette. If not a color brewer palette name, geom.colors must be of same length as moderator values used in the plot (see mdrt.values). See also 'Details' in sjp.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.

fill.color — Fill color of the shaded area between the minimum and maximum lines. Default is "grey". Either set fill.color to NULL or use 0 for fill.alpha if you want to hide the shaded area.

fill.alpha — Alpha value (transparancy) of the shaded area between the minimum and maximum lines. Default is 0.4. Use either 0 or set fill.color to NULL if you want to hide the shaded area.

show.values — Logical, whether values should be plotted 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.

jitter.ci — Logical, if TRUE and show.ci = TRUE and confidence bands are displayed as error bars, adds jittering to lines and error bars to avoid overlapping.

p.kr — logical, if TRUE, p-value estimation is based on conditional F-tests with Kenward-Roger approximation for the df. Caution: Computation may take very long time for large samples!

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.

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.

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

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

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.

prnt.plot — logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

... — Other arguments passed down to further functions. Currently, following arguments are supported:

?effects::effect Any arguments accepted by the effect resp. allEffects function, for type = "eff".

width The width-argument for error bars.

alpha The alpha-argument for confidence bands.

level The level-argument confidence bands.

Details

type = "eff" plots the overall effects (marginal effects) of the interaction, with all remaining covariates set to the mean. Effects are calculated using the effect- function from the **effects**-package. You can pass further arguments down to allEffects for flexible function call via the ...-argument.

`type = "cond"` plots the effective *change* or *impact* (conditional effect) on a dependent variable of a moderation effect, as described by Grace-Martin, i.e. the difference of the moderation effect on the dependent variable in *presence* and *absence* of the moderating effect (*simple slope* plot or *conditional effect*, see Hayes 2012). All remaining predictors are set to zero (i.e. ignored and not adjusted for). Hence, this plot type may be used especially for *binary or dummy coded* moderator values (see also Esarey and Summer 2015). This type *does not* show the overall effect (marginal mean, i.e. adjusted for all other predictors and covariates) of interactions on the result of Y. Use `type = "eff"` for effect displays similar to the `effect`-function from the **effects**-package.

The argument `int.term` only applies to `type = "eff"` and can be used to select a specific interaction term of the model that should be plotted. The function then calls `effect(int.term, fit)` to compute effects for this specific interaction term only. This approach is recommended, when the fitted model contains many observations and/or variables, which may slow down the effect-computation dramatically. In such cases, consider computing effects for selected interaction terms only with `int.terms`. See 'Examples'.

Value

(Insisibily) returns the ggplot-objects with the complete plot-list (`plot.list`) as well as the data frames that were used for setting up the ggplot-objects (`data.list`).

Note

Note that beside interaction terms, also the single predictors of each interaction (main effects) must be included in the fitted model as well. Thus, `lm(dep ~ pred1 * pred2)` will work, but `lm(dep ~ pred1:pred2)` won't!

For `type = "eff"`, predictors of interactions that are introduced first into the model are used as grouping variable, while the latter predictor is printed along the x-axis (i.e. lm(y~a+b+a:b) means that "a" is used as grouping variable and "b" is plotted along the x-axis).

References

- Aiken and West (1991). Multiple Regression: Testing and Interpreting Interactions.
- Brambor T, Clark WR and Golder M (2006) Understanding Interaction Models: Improving Empirical Analyses. Political Analysis 14: 63-82. download
- Esarey J, Sumner JL (2015) Marginal Effects in Interaction Models: Determining and Controlling the False Positive Rate. download
- Fox J (2003) Effect displays in R for generalised linear models. Journal of Statistical Software 8:15, 1–27, <http://www.jstatsoft.org/v08/i15/>
- Hayes AF (2012) PROCESS: A versatile computational tool for observed variable mediation, moderation, and conditional process modeling [White paper] download
- Grace-Martin K: Interpreting Interactions in Regression
- Grace-Martin K: Clarifications on Interpreting Interactions in Regression

See Also

sjPlot manual: sjp.int

Examples

```
# Note that the data sets used in this example may not be perfectly suitable for
# fitting linear models. I just used them because they are part of the R-software.

# fit "dummy" model. Note that moderator should enter
# first the model, followed by predictor. Else, use
# argument "swap.pred" to change predictor on
# x-axis with moderator
fit <- lm(weight ~ Diet * Time, data = ChickWeight)

# show summary to see significant interactions
summary(fit)

# plot regression line of interaction terms, including value labels
sjp.int(fit, type = "eff", show.values = TRUE)


# load sample data set
library(sjmisc)
library(sjlabelled)
data(efc)
# create data frame with variables that should be included
# in the model
mydf <- data.frame(usage = efc$tot_sc_e,
                   sex = efc$c161sex,
                   education = efc$c172code,
                   burden = efc$neg_c_7,
                   dependency = efc$e42dep)
# convert gender predictor to factor
mydf$sex <- relevel(factor(mydf$sex), ref = "2")
# fit "dummy" model
fit <- lm(usage ~ .*., data = mydf)
summary(fit)

# plot interactions. note that type = "cond" only considers
# significant interactions by default. use "plevel" to
# adjust p-level sensivity
sjp.int(fit, type = "cond")

# plot only selected interaction term for
# type = "eff"
sjp.int(fit, type = "eff", int.term = "sex*education")

# plot interactions, using mean and sd as moderator
# values to calculate interaction effect
sjp.int(fit, type = "eff", mdrt.values = "meansd")
sjp.int(fit, type = "cond", mdrt.values = "meansd")

# plot interactions, including those with p-value up to 0.1
sjp.int(fit, type = "cond", plevel = 0.1)

# ------------------------------
```

```
# Predictors for negative impact of care.
# Data from the EUROFAMCARE sample dataset
# ------------------------------
# create binary response
y <- ifelse(efc$neg_c_7 < median(stats::na.omit(efc$neg_c_7)), 0, 1)
# create data frame for fitted model
mydf <- data.frame(y = as.factor(y),
                   sex = as.factor(efc$c161sex),
                   barthel = as.numeric(efc$barthtot))
# fit model
fit <- glm(y ~ sex * barthel, data = mydf, family = binomial(link = "logit"))
# plot interaction, increase p-level sensivity
sjp.int(fit, type = "eff", legend.labels = get_labels(efc$c161sex), plevel = 0.1)
sjp.int(fit, type = "cond", legend.labels = get_labels(efc$c161sex), plevel = 0.1)

## Not run:
# load sample data set
library(sjmisc)
data(efc)
# create data frame with variables that should be included
# in the model
mydf <- data.frame(burden = efc$neg_c_7,
                   sex = efc$c161sex,
                   education = efc$c172code,
                   barthel = efc$barthtot)
# convert gender predictor to factor
mydf$sex <- factor(mydf$sex)
mydf$education <- factor(mydf$education)
# name factor levels and dependent variable
levels(mydf$sex) <- c("female", "male")
levels(mydf$education) <- c("low", "mid", "high")
mydf$burden <- set_label(mydf$burden, lab = "care burden")
# fit "dummy" model
fit <- lm(burden ~ .*., data = mydf)

# plot effects
sjp.int(fit, type = "eff", show.ci = TRUE)

# plot effects, faceted
sjp.int(fit, type = "eff", int.plot.index = 3, show.ci = TRUE, facet.grid = TRUE)
## End(Not run)
```

`sjp.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

```
sjp.kfold_cv(data, formula, k = 5, fit)
```

Arguments

data	A data frame, used to split the data into k trainig-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 (using the crossv_kfold-function) 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)

sjp.kfold_cv(efc, neg_c_7 ~ e42dep + c172code + c12hour)
sjp.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)
sjp.kfold_cv(efc, fit = fit)

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

`sjp.likert` *Plot likert scales as centered stacked bars*

Description

Plot likert scales as centered stacked bars.

Usage

```
sjp.likert(items, title = NULL, legend.title = NULL, legend.labels = NULL,
  axis.titles = NULL, axis.labels = NULL, catcount = NULL,
  cat.neutral = NULL, sort.frq = NULL, weight.by = NULL,
  title.wtd.suffix = NULL, wrap.title = 50, wrap.labels = 30,
  wrap.legend.title = 30, wrap.legend.labels = 28, geom.size = 0.6,
  geom.colors = "BrBG", cat.neutral.color = "grey70",
  intercept.line.color = "grey50", reverse.colors = FALSE,
  values = "show", show.n = TRUE, show.legend = TRUE,
  show.prc.sign = FALSE, grid.range = 1, grid.breaks = 0.2,
  expand.grid = TRUE, digits = 1, coord.flip = TRUE, prnt.plot = TRUE)
```

Arguments

`items`	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.
`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.

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 sjp.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.outide" 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. 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 and values.
coord.flip	logical, if TRUE, the x and y axis are swapped.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) 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.neg for the negative values, df.pos for the positive values and df.neutral for the neutral category values).

Note

Note that only even numbers of categories are possible to plot, so the "positive" and "negative" values can be splitted into two halfs. 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.

See Also

sjPlot manual: sjp.likert

Examples

```
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
find_var(efc, pattern = "cop", out = "df") %>% sjp.likert()

sjp.likert(
  find_var(efc, pattern = "cop", out = "df"),
  grid.range = 1.2,
```

```
  expand.grid = FALSE,
  values = "sum.outside",
  show.prc.sign = TRUE
)
```

---

sjp.lm *Plot estimates, predictions or effects of linear models*

Description

Depending on the type, this function plots coefficients (estimates) of linear regressions (including panel models fitted with the plm-function from the **plm**-package and generalized least squares models fitted with the gls-function from the **nlme**-package) with confidence intervals as dot plot (forest plot), model assumptions for linear models or slopes and scatter plots for each single coefficient. See type for details.

Usage

```
sjp.lm(fit, type = "lm", vars = NULL, group.estimates = NULL,
  remove.estimates = NULL, sort.est = TRUE, poly.term = NULL,
  title = NULL, legend.title = NULL, axis.labels = NULL,
  axis.title = NULL, resp.label = NULL, geom.size = NULL,
  geom.colors = "Set1", wrap.title = 50, wrap.labels = 25,
  axis.lim = NULL, grid.breaks = NULL, show.values = TRUE,
  show.p = TRUE, show.ci = TRUE, show.legend = FALSE,
  show.loess = FALSE, show.loess.ci = FALSE, show.summary = FALSE,
  show.scatter = TRUE, point.alpha = 0.2, point.color = NULL,
  jitter.ci = FALSE, digits = 2, vline.type = 2, vline.color = "grey70",
  coord.flip = TRUE, y.offset = 0.15, facet.grid = TRUE,
  complete.dgns = FALSE, prnt.plot = TRUE, ...)
```

Arguments

fit	Fitted linear regression model (of class lm, gls or plm).
type	Type of plot. Use one of following:
	"lm" (default) for forest-plot of estimates. If the fitted model only contains one predictor, slope-line is plotted.
	"pred" to plot predicted values (marginal effects) for specific model terms. See 'Details'.
	"eff" to plot marginal effects of all terms in fit. Note that interaction terms are excluded from this plot.
	"std" for forest-plot of standardized beta values.
	"std2" for forest-plot of standardized beta values, however, standardization is done by dividing by two sd (see 'Details').

"resid" to plot regression lines for each single predictor of the fitted model, against the residuals (linear relationship between each model term and residuals). May be used for model diagnostics.

"ma" to check model assumptions.

"vif" to plot Variance Inflation Factors.

vars — Numeric vector with column indices of selected variables or a character vector with variable names of selected variables from the fitted model, which should be used to plot - depending on type - estimates, fixed effects slopes or predicted values (mean, probabilities, incidents rates, ...). See 'Examples'.

group.estimates — Numeric or character vector, indicating a group identifier for each estimate. Dots and confidence intervals of estimates are coloured according to their group association. See 'Examples'.

remove.estimates — Character vector with coefficient names that indicate which estimates should be removed from the plot. remove.estimates = "est_name" would remove the estimate *est_name*. Default is NULL, i.e. all estimates are printed.

sort.est — Logical, determines whether estimates should be sorted according to their values. If group.estimates is *not* NULL, estimates are sorted according to their group assignment.

poly.term — name of a polynomial term in fit as string. Needs to be specified, if type = "poly", in order to plot marginal effects for polynomial terms. See 'Examples'.

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. Note that only some plot types have legends (e.g. type = "pred" or when grouping estimates with group.estimates).

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

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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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 = ...)

resp.label — Name of dependent variable, as string. Only used if fitted model has only one predictor and type = "lm".

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 palette for geoms. If group.estimates is *not* specified, must either be vector with two color values or a specific color palette code (see

	'Details' in sjp.grpfrq). Else, if group.estimates is specified, geom.colors must be a vector of same length as groups. See 'Examples'.
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, or both. For multiple plot outputs (e.g., from type = "eff" or type = "slope" in sjp.glm), 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.values	Logical, whether values should be plotted or not.
show.p	Logical, adds significance levels to values, or value and variable labels.
show.ci	Logical, if TRUE, depending on type, a confidence interval or region is added to the plot. For frequency plots, the confidence interval for the relative frequencies are shown.
show.legend	logical, if TRUE, and depending on plot type and function, a legend is added to the plot.
show.loess	logical, if TRUE, and depending on type, an additional loess-smoothed line is plotted.
show.loess.ci	logical, if TRUE, a confidence region for the loess-smoothed line will be plotted. Default is FALSE. Only applies, if show.loess = TRUE (and for sjp.lmer, only applies if type = "fe.slope" or type = "fe.resid").
show.summary	Logical, if TRUE, a summary with model statistics is added to the plot.
show.scatter	Logical, if TRUE (default), adds a scatter plot of data points to the plot. Only applies for slope-type or predictions plots. For most plot types, dots are jittered to avoid overplotting, hence the points don't reflect exact values in the data.
point.alpha	Alpha value of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
point.color	Color of of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
jitter.ci	Logical, if TRUE and show.ci = TRUE and confidence bands are displayed as error bars, adds jittering to lines and error bars to avoid overlapping.
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
vline.type	Linetype of the vertical "zero point" line. Default is 2 (dashed line).
vline.color	Color of the vertical "zero point" line. Default value is "grey70".
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).
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.

`complete.dgns` Logical, if `TRUE`, additional tests are performed. Default is `FALSE` Only applies if `type = "ma"`.

`prnt.plot` logical, if `TRUE` (default), plots the results as graph. Use `FALSE` if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

`...` Other arguments passed down to further functions. Currently, following arguments are supported:

`?effects::effect` Any arguments accepted by the `effect` resp. `allEffects` function, for `type = "eff"`.

`width` The `width`-argument for error bars.

`alpha` The `alpha`-argument for confidence bands.

`level` The `level`-argument confidence bands.

Details

`type = "lm"` if fitted model only has one predictor, no forest plot is shown. Instead, a regression line with confidence interval (in blue) is plotted by default, and a loess-smoothed line without confidence interval (in red) can be added if argument `show.loess = TRUE`.

`type = "std2"` plots standardized beta values, 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. This standardization uses the `standardize`-function from the **arm**-package.

`type = "slope"` regression lines (slopes) with confidence intervals for each single predictor of the fitted model are plotted, i.e. all predictors of the fitted model are extracted and for each of them, the linear relationship is plotted against the response variable. Other predictors are omitted, so this plot type is intended to check the linear relationship between a predictor and the response.

`type = "resid"` is similar to the `type = "slope"` option, however, each predictor is plotted against the residuals (instead of response).

`type = "pred"` plots predicted values of the response, related to specific model predictors. This plot type calls `predict(fit, newdata = model.frame, type = "response")` and requires the `vars` argument to select specific terms that should be used for the x-axis and - optional - as grouping factor. Hence, `vars` must be a character vector with the names of one or two model predictors. See 'Examples'.

`type = "eff"` computes the marginal effects for all predictors, using the `allEffects` function. I.e. for each predictor, the predicted values towards the response are plotted, with all remaining co-variates set to the mean. Due to possible different scales of predictors, a faceted plot is printed (instead of plotting all lines in one plot). You can pass further arguments down to `allEffects` for flexible function call via the `...`-argument.

`type = "poly"` plots the marginal effects of polynomial terms in `fit`, using the `effect` function, but only for a selected polynomial term, which is specified with `poly.term`. This function helps undertanding the effect of polynomial terms by plotting the curvilinear relationships of response and quadratic, cubic etc. terms. This function accepts following argument.

`type = "ma"` checks model assumptions. Please note that only three arguments are relevant: `fit` and `complete.dgns`. All other arguments are ignored.

`type = "vif"` Variance Inflation Factors (check for multicollinearity) are plotted. As a rule of thumb, values below 5 are considered as good and indicate no multicollinearity, values between 5 and 10 may be tolerable. Values greater than 10 are not acceptable and indicate multicollinearity between model's predictors.

Value

Depending on the `type`, in most cases (insisibily) 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`). For `type = "ma"`, an updated model with removed outliers is returned.

References

Gelman A (2008) "Scaling regression inputs by dividing by two standard deviations." *Statistics in Medicine 27: 2865–2873.* http://www.stat.columbia.edu/~gelman/research/published/standardizing7.pdf

Hyndman RJ, Athanasopoulos G (2013) "Forecasting: principles and practice." OTexts; accessed from https://www.otexts.org/fpp/5/4.

See Also

sjPlot manual: sjp.lm for more details and examples of this function; use `sjp.poly` to see which polynomial degree fits best for possible polynomial terms.

Examples

```
# ------------------------------------------------
# plotting estimates of linear models as forest plot
# ------------------------------------------------
# fit linear model
fit <- lm(airquality$Ozone ~ airquality$Wind + airquality$Temp + airquality$Solar.R)

# plot estimates with CI
sjp.lm(fit, grid.breaks = 2)

# plot estimates with CI
# and with narrower tick marks
# (because "grid.breaks" was not specified)
sjp.lm(fit)

# ------------------------------------------------
# plotting regression line of linear model (done
# automatically if fitted model has only 1 predictor)
# ------------------------------------------------
library(sjmisc)
data(efc)
# fit model
fit <- lm(neg_c_7 ~ quol_5, data=efc)
# plot regression line with label strings
sjp.lm(fit, resp.label = "Burden of care",
```

```
       axis.labels = "Quality of life", show.loess = TRUE)

# ------------------------------------------------
# plotting regression lines of each single predictor
# of a fitted model
# ------------------------------------------------
library(sjmisc)
data(efc)
# fit model
fit <- lm(tot_sc_e ~ c12hour + e17age + e42dep, data=efc)

# reression line and scatter plot
sjp.lm(fit, type = "slope")

# reression line w/o scatter plot
sjp.lm(fit, type = "slope", show.scatter = FALSE)

# --------------------------
# plotting model assumptions
# --------------------------
sjp.lm(fit, type = "ma")

## Not run:
# --------------------------
# grouping estimates
# --------------------------
library(sjmisc)
data(efc)
fit <- lm(barthtot ~ c160age + e17age + c12hour + e16sex + c161sex + c172code,
          data = efc)

# order estimates according to coefficient's order
sjp.lm(fit, group.estimates = c(1, 1, 2, 3, 3, 4),
       geom.colors = c("green", "red", "blue", "grey"), sort.est = FALSE)

fit <- lm(barthtot ~ c160age + c12hour + e17age+ c161sex + c172code + e16sex,
          data = efc)

# force order of estimates according to group assignment
sjp.lm(fit, group.estimates = c(1, 2, 1, 3, 4, 3),
       geom.colors = c("green", "red", "blue", "grey"), sort.est = TRUE)

# --------------------------
# predicted values for response
# --------------------------
library(sjmisc)
data(efc)
efc$education <- to_label(to_factor(efc$c172code))
efc$gender <- to_label(to_factor(efc$c161sex))
fit <- lm(barthtot ~ c160age + c12hour + e17age + gender + education,
          data = efc)

sjp.lm(fit, type = "pred", vars = "c160age")
```

```
# with loess
sjp.lm(fit, type = "pred", vars = "e17age", show.loess = TRUE)

# grouped
sjp.lm(fit, type = "pred", vars = c("c12hour", "education"))

# grouped, non-facet
sjp.lm(fit, type = "pred", vars = c("c12hour", "education"),
       facet.grid = FALSE)

# two groupings
sjp.lm(fit, type = "pred", vars = c("c12hour", "gender", "education"))

# --------------------------
# plotting polynomial terms
# --------------------------
library(sjmisc)
data(efc)
# fit sample model
fit <- lm(tot_sc_e ~ c12hour + e17age + e42dep, data = efc)
# "e17age" does not seem to be linear correlated to response
# try to find appropiate polynomial. Grey line (loess smoothed)
# indicates best fit. Looks like x^3 has a good fit.
# (not checked for significance yet).
sjp.poly(fit, "e17age", 2:4, show.scatter = FALSE)
# 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
sjp.lm(fit, type = "poly", poly.term = "e17age")

library(splines)
# fit new model with "splines"-package, "bs"
fit <- lm(tot_sc_e ~ c12hour + e42dep + bs(e17age, 3), data = efc)
# plot marginal effects of polynomial term, same call as above
sjp.lm(fit, type = "poly", poly.term = "e17age")
## End(Not run)
```

`sjp.lmer` *Plot estimates, predictions or effects of linear mixed effects models*

Description

By default, this function plots estimates (coefficients) with confidence intervalls of either fixed effects or random effects of linear mixed effects models (that have been fitted with the `lmer`-function of the **lme4**-package). Furhermore, this function also plot predicted values or diagnostic plots.

Usage

```
sjp.lmer(fit, type = "re", vars = NULL, ri.nr = NULL,
  group.estimates = NULL, remove.estimates = NULL, emph.grp = NULL,
  sample.n = NULL, poly.term = NULL, sort.est = NULL, title = NULL,
  legend.title = NULL, axis.labels = NULL, axis.title = NULL,
  geom.size = NULL, geom.colors = "Set1", show.values = TRUE,
  show.p = TRUE, show.ci = FALSE, show.legend = FALSE,
  show.loess = FALSE, show.loess.ci = FALSE, show.intercept = FALSE,
  string.interc = "(Intercept)", p.kr = TRUE, show.scatter = TRUE,
  point.alpha = 0.2, point.color = NULL, jitter.ci = FALSE,
  fade.ns = FALSE, axis.lim = NULL, digits = 2, vline.type = 2,
  vline.color = "grey70", facet.grid = TRUE, free.scale = FALSE,
  y.offset = 0.1, prnt.plot = TRUE, ...)
```

Arguments

`fit`	a fitted model as returned by the `lmer`-function.
`type`	type of plot. Use one of following:

- `"re"` (default) for conditional modes of random effects as forest plot
- `"fe"` for estimates of fixed effects as forest plot
- `"fe.std"` for standardized estimates of fixed effects as forest plot
- `"fe.resid"` to plot regression lines (slopes) with confidence intervals for each single fixed effect (against residuals), i.e. all fixed terms are extracted and each is plotted against the model residuals (linear relationship between each fixed term and residuals)
- `"fe.cor"` for correlation matrix of fixed effects
- `"re.qq"` for a QQ-plot of random effects (random effects quantiles against standard normal quantiles)
- `"ri.slope"` for fixed effects slopes depending on the random intercept.
- `"rs.ri"` for fitted regression lines indicating the random slope-intercept pairs. Use this to visualize the random parts of random slope-intercept (or repeated measure) models. When having too many groups, use `sample.n` argument.
- `"coef"` for joint (sum of) random and fixed effects coefficients for each explanatory variable for each level of each grouping factor as forest plot.
- `"pred"` to plot predicted values for the response, related to specific model predictors and conditioned on random effects. See 'Details'.
- `"pred.fe"` to plot predicted values for the response, related to specific model predictors and conditioned on fixed effects only. See 'Details'.
- `"eff"` to plot marginal effects of all fixed terms in `fit`. Note that interaction terms are excluded from this plot; use `sjp.int` to plot effects of interaction terms. See also 'Details' of `sjp.lm`.
- `"eff.ri"` to plot marginal effects of all fixed terms in `fit`, varying by the random intercepts.

"poly" to plot predicted values (marginal effects) of polynomial terms in fit. Use poly.term to specify the polynomial term in the fitted model (see 'Examples' here and 'Details' of sjp.lm).

"ma" to check model assumptions. Note that no further arguments except fit are relevant for this option. All other arguments are ignored.

vars
: Numeric vector with column indices of selected variables or a character vector with variable names of selected variables from the fitted model, which should be used to plot - depending on type - estimates, fixed effects slopes or predicted values (mean, probabilities, incidents rates, ...). See 'Examples'.

ri.nr
: Numeric vector. If type = "re" or type = "ri.slope", 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.

group.estimates
: Numeric or character vector, indicating a group identifier for each estimate. Dots and confidence intervals of estimates are coloured according to their group association. See 'Examples'.

remove.estimates
: Character vector with coefficient names that indicate which estimates should be removed from the plot. remove.estimates = "est_name" would remove the estimate *est_name*. Default is NULL, i.e. all estimates are printed.

emph.grp
: Numeric vector with index numbers of grouping levels (from random effect). If type = "ri.slope" and facet.grid = FALSE, an integrated plot of predicted probabilities of fixed effects resp. fixed effects slopes for each grouping level is plotted. To better find certain groups, use this argument to emphasize these groups in the plot. See 'Examples'.

sample.n
: Numeric vector. only applies, if type = "rs.ri". If plot has many random intercepts (grouping levels), overplotting of regression lines may occur. In this case, consider random sampling of grouping levels. If sample.n is of length 1, a random sample of sample.n observation is selected to plot random intercepts. If sample.n is of length > 1, random effects indicated by the values in sample.n are selected to plot random effects. Use the latter option to always select a fixed, identical set of random effects for plotting (useful when ecomparing multiple models).

poly.term
: name of a polynomial term in fit as string. Needs to be specified, if type = "poly", in order to plot marginal effects for polynomial terms. See 'Examples'.

sort.est
: Determines in which way estimates are sorted in the plot:

- If NULL (default), no sorting is done and estimates are sorted in order of model coefficients.
- If sort.est = "sort.all", estimates are re-sorted for each coefficient (only applies if type = "re" and facet.grid = FALSE), i.e. the estimates of the random effects for each predictor are sorted and plotted to an own plot.
- If type = "fe" or type = "fe.std", TRUE will sort estimates
- If type = "re", specify a predictor's / coefficient's name to sort estimates according to this coefficient.

	See 'Examples'.
title	Character vector with one or more labels that are used as plot title.
legend.title	character vector, used as title for the plot legend.
axis.labels	Character vector with labels for the model terms, used as axis labels. For mixed models, should either be vector of fixed effects variable labels (if type = "fe" or type = "fe.std") or a vector of group (value) labels from the random intercept's categories (if type = "re").
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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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 = ...)
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 sjp.grpfrq.
show.values	Logical, whether values should be plotted or not.
show.p	Logical, adds significance levels to values, or value and variable labels.
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.legend	logical, if TRUE, and depending on plot type and function, a legend is added to the plot.
show.loess	logical, if TRUE, and depending on type, an additional loess-smoothed line is plotted.
show.loess.ci	logical, if TRUE, a confidence region for the loess-smoothed line will be plotted. Default is FALSE. Only applies, if show.loess = TRUE (and for sjp.lmer, only applies if type = "fe.slope" or type = "fe.resid").
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.
string.interc	String, axis label of intercept estimate. Only applies, if show.intercept = TRUE and axis.labels is not NULL.
p.kr	logical, if TRUE, p-value estimation is based on conditional F-tests with Kenward-Roger approximation for the df. Caution: Computation may take very long time for large samples!
show.scatter	Logical, if TRUE (default), adds a scatter plot of data points to the plot. Only applies for slope-type or predictions plots. For most plot types, dots are jittered to avoid overplotting, hence the points don't reflect exact values in the data.

point.alpha — Alpha value of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.

point.color — Color of of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.

jitter.ci — Logical, if TRUE and show.ci = TRUE and confidence bands are displayed as error bars, adds jittering to lines and error bars to avoid overlapping.

fade.ns — Logical, if TRUE, non significant estimates will be printed in slightly faded colors.

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 sjp.glm), axis.lim may also be a list of vectors of length 2, defining axis limits for each plot (only if non-faceted).

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

vline.type — Linetype of the vertical "zero point" line. Default is 2 (dashed line).

vline.color — Color of the vertical "zero point" line. Default value is "grey70".

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.

free.scale — Logical, if TRUE and facet.grid = TRUE, each facet grid gets its own fitted scale. If free.scale = FALSE, each facet in the grid has the same scale range.

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

prnt.plot — logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

... — Other arguments passed down to further functions. Currently, following arguments are supported:

?effects::effect Any arguments accepted by the effect resp. allEffects function, for type = "eff".

width The width-argument for error bars.

alpha The alpha-argument for confidence bands.

level The level-argument confidence bands.

Details

type = "re" plots the conditional modes of the random effects, inclduing predicion intervals. It basically does the same as dotplot(ranef(fit, condVar = TRUE)[[i]]), where i denotes the random effect index.

type = "fe.slope" plots the linear relationship between each fixed effect and the response. The regression lines are *not* based on the fitted model's fixed effects estimates (though they may be similar). This plot type just computes a simple linear model for each fixed effect and response. Hence, it's intended for checking model assumptions, i.e. if predictor and respone are in a linear relationship. You may use the show.loess argument to see whether the linear line differs from the best fitting line.

type = "fe.resid" Similar to type = "fe.slope", this this type is intended for checking model assumptions. However, fitted values are plotted against the residuals instead of response.

`type = "eff"` plots the adjusted (marginal) effects for each fixed effect, with all co-variates set to the mean, as returned by the `allEffects` function. You can pass further arguments down to `allEffects` for flexible function call via the `...`-argument.

`type = "eff.ri"` plots the adjusted (marginal) effects for each fixed effect, with all co-variates set to the mean, varying by the random intercepts. This plot type basically does the same as `type = "ri.slope"`, except that the co-variates are not set to zero, but adjusted for. This plot type differs from `type = "ri.slope"` only in the adjusted y-axis-scale

`type = "rs.ri"` plots regression lines for the random parts of the model, i.e. all random slopes for each random intercept. As the random intercepts describe the deviation from the global intercept, the regression lines are computed as global intercept + random intercept + random slope. In case of overplotting, use the `sample.n` argument to randomly sample a limited amount of groups.

`type = "ri.slope"` plots regression lines for each fixed effect (slopes) within each random intercept. Lines are based on the fixed effects intercept, plus each random intercept and each specific fixed term's estimate. All other fixed effects are set to zero (i.e. ignored), which corresponds to `b0 + b0[r1-rn] + bi * xi)` (where `xi` is the estimate of fixed effects, `b0` is the intercept of the fixed effects and `b0[r1-rn]` are all random intercepts).

`type = "coef"` forest plot of joint fixed and random effect coefficients, as retrieved by `coef.merMod`, it's simply `ranef + fixef`.

`type = "pred"` **or** `type = "pred.fe"` predicted values for response, conditional on fixed effects only or on random intercept. It's calling `predict(fit, type = "response", re.form = NA)` resp. `predict(fit, type = "response", re.form = NULL)` to compute the values. This plot type requires the `vars` argument to select specific terms that should be used for the x-axis and - optional - as grouping factor. Hence, `vars` must be a character vector with the names of one or two model predictors. See 'Examples'.

Value

(Insisibily) returns

- the ggplot-object (`plot`), if `type = "fe"` or if `type = "re"` and `facet.grid = TRUE`). Multiple plots (`type = "re"` and if `facet.grid = FALSE`) are returned in the object `plot.list`.
- a list of ggplot-objects (`plot.list`). see `plot` for details.
- a data frame `data` with the data used to build the ggplot-object(s).

Note

Computation of p-values (if necessary and if `p.kr = TRUE`) are based on conditional F-tests with Kenward-Roger approximation for the df, using the **pbkrtest**-package. If **pbkrtest** is not available or `p.kr = FALSE`, computation of p-values is based on normal-distribution assumption, treating the t-statistics as Wald z-statistics. See 'Details' in `p_value`.

See Also

sjPlot manual: sjp.lmer

Examples

```
# fit model
library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)

# simple plot
sjp.lmer(fit)

# plot fixed effects
sjp.lmer(fit, type = "fe")

# sort by predictor Days
sjp.lmer(fit, sort.est = "Days")

# plot each predictor as own plot
# sort each plot
sjp.lmer(fit, facet.grid = FALSE, sort.est = "sort.all")

library(sjmisc)
library(sjlabelled)
data(efc)
# prepare group variable
efc$grp = as.factor(efc$e15relat)
levels(x = efc$grp) <- get_labels(efc$e15relat)
# data frame for fitted model
mydf <- data.frame(neg_c_7 = as.numeric(efc$neg_c_7),
                   sex = as.factor(efc$c161sex),
                   c12hour = as.numeric(efc$c12hour),
                   barthel = as.numeric(efc$barthtot),
                   grp = efc$grp)
# fit lmer
fit <- lmer(neg_c_7 ~ sex + c12hour + barthel + (1|grp), data = mydf)

sjp.lmer(fit, type = "fe.std", sort.est = TRUE)

# highlight specific grouping levels, in this case we compare
# spouses, children and children-in-law
sjp.lmer(fit, type = "ri.slope", emph.grp = c(1, 2, 4), vars = "c12hour")

## Not run:
# plotting polynomial terms
# check linear relation between predictors and response
sjp.lmer(fit, type = "fe.slope", show.loess = TRUE)

# "barthel" does not seem to be linear correlated to response
# try to find appropiate polynomial. Grey line (loess smoothed)
# indicates best fit. Looks like x^4 has the best fit,
# however, x^2 seems to be suitable according to p-values.
sjp.poly(fit, "barthel", 2:4, show.scatter = FALSE)

# fit new model
fit <- lmer(neg_c_7 ~ sex + c12hour + barthel +
```

```
            I(barthel^2) + (1|grp), data = mydf)

# plot marginal effects of polynomial term
sjp.lmer(fit, type = "poly", poly.term = "barthel")

# lme4 complaints about scale of polynomial term, so
# try centering this predictor
mydf$barthel_s <- sjmisc::std(mydf$barthel)

# re-fit model
fit_s <- lmer(neg_c_7 ~ sex + c12hour + barthel_s +
              I(barthel_s^2) + (1|grp), data = mydf)

# plot marginal effects of centered, scaled polynomial term
sjp.lmer(fit_s, type = "poly", poly.term = "barthel_s")

# scaling also improved p-values
sjt.lmer(fit, fit_s)

# plotting predicted values for response
# conditioned on random effects
sjp.lmer(fit, type = "pred", vars = "c12hour")

# grouped, for fixed effects only
sjp.lmer(fit, type = "pred.fe", vars = c("c12hour", "sex"))

# grouped, for fixed effects only, non-facted
sjp.lmer(fit, type = "pred.fe", vars = c("c12hour", "sex"),
         facet.grid = FALSE, show.ci = FALSE)
## End(Not run)
```

---

sjp.pca *Plot PCA results*

Description

Performs a principle component analysis on a data frame or matrix (with varimax or oblimin rotation) and plots the factor solution as ellipses or tiles.

In case a data frame is used as argument, 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

```
sjp.pca(data, rotation = c("varimax", "oblimin"), nmbr.fctr = NULL,
  fctr.load.tlrn = 0.1, plot.eigen = FALSE, digits = 2, title = NULL,
  axis.labels = NULL, type = c("bar", "circle", "tile"), geom.size = 0.6,
  geom.colors = "RdBu", wrap.title = 50, wrap.labels = 30,
  show.values = TRUE, show.cronb = TRUE, prnt.plot = 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 "varimax" for orthogonal rotation or "oblimin" for oblique transformation.
nmbr.fctr	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.
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
plot.eigen	If TRUE, a plot showing the Eigenvalues according to the Kaiser criteria is plotted to determine the number of factors.
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
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.
type	Plot type resp. geom type. May be one of following: "circle" or "tile" circular or tiled geoms, or "bar" for a bar plot. You may use initial letter only for this argument.
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 sjp.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.values	Logical, whether values should be plotted or not.
show.cronb	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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Invisibly) returns a structure with

- the rotated factor loading matrix (varim)
- the column indices of removed variables (for more details see next list item) (removed.colindex)
- an updated data frame containing all factors that have a clear loading on a specific scale in case data was a data frame (See argument fctr.load.tlrn for more details) (removed.df)
- the factor.index, i.e. the column index of each variable with the highest factor loading for each factor,
- the ggplot-object (plot),
- the data frame that was used for setting up the ggplot-object (df).

See Also

- sjPlot manual: sjp.pca
- sjt.pca

Examples

```
library(sjmisc)
data(efc)
# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# recveive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")

# manually compute PCA
pca <- prcomp(
  na.omit(efc[, start:end]),
  retx = TRUE,
  center = TRUE,
  scale. = TRUE
)
# plot results from PCA as circles, including Eigenvalue-diagnostic.
# note that this plot does not compute the Cronbach's Alpha
sjp.pca(pca, plot.eigen = TRUE, type = "circle", geom.size = 10)

# use data frame as argument, let sjp.pca() compute PCA
sjp.pca(efc[, start:end])
sjp.pca(efc[, start:end], type = "tile")
```

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, poly.degree, poly.scale = FALSE, fun = NULL,
  axis.title = NULL, geom.colors = NULL, geom.size = 0.8,
  show.loess = TRUE, show.loess.ci = TRUE, show.p = TRUE,
  show.scatter = TRUE, point.alpha = 0.2, point.color = "#404040",
  loess.color = "#808080", prnt.plot = TRUE)
```

Arguments

x	A vector, representing the response variable of a linear (mixed) model; or a linear (mixed) model as returned by lm or lmer.
poly.term	If x is a vector, poly.term should also be a vector, representing the polynomial term (independent variabl) in the model; if x is a fitted model, poly.term should be the polynomial term's name as character string. See 'Examples'.
poly.degree	Numeric, or numeric vector, indicating the degree of the polynomial. If poly.degree is a numeric vector, multiple polynomial curves for each degree are plotted. See 'Examples'.
poly.scale	Logical, if TRUE, poly.term will be scaled before linear regression is computed. Default is FALSE. Scaling the polynomial term may have an impact on the resulting p-values.
fun	Linear function when modelling polynomial terms. Use fun = "lm" for linear models, or fun = "glm" for generalized linear models. When x is not a vector, but a fitted model object, the function is detected automatically. If x is a vector, fun defaults to "lm".
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. To set multiple axis titles (e.g. with type = "eff" for many predictors), axis.title must be a character vector of same length of plots that are printed. In this case, each plot gets an own axis title (applying, for instance, to the y-axis for type = "eff"). **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 = ...)
geom.colors	User defined color palette for geoms. If group.estimates is *not* specified, must either be vector with two color values or a specific color palette code (see 'Details' in sjp.grpfrq). Else, if group.estimates is specified, geom.colors must be a vector of same length as groups. See 'Examples'.

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.
show.loess	Logical, if TRUE, an additional loess-smoothed line is plotted.
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. Only applies for slope-type or predictions plots. For most plot types, dots are jittered to avoid overplotting, hence the points don't reflect exact values in the data.
point.alpha	Alpha value of point-geoms in the scatter plots. Only applies, if show.scatter = TRUE.
point.color	Color of 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.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

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, lm(y ~ x + I(x^2) + ... + I(x^i)) is repeatedly computed for all values in poly.degree, and the predicted values of the reponse 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 poly function, with argument 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

(Insisibily) returns

plot the ggplot-object with the complete plot

df the data frame that was used for setting up the ggplot-object

cutpoints a data frame that indicates x-values and predicted y-values of each direction change in the loess curvature

See Also

To plot marginal effects of polynomial terms, call sjp.lm with type = "poly", or sjp.lmer respectively for linear mixed models.

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 <- lm(tot_sc_e ~ c12hour + e17age + e42dep, data = efc)
# inspect relationship between predictors and response
sjp.lm(fit, type = "slope", show.loess = TRUE, show.scatter = FALSE)
# "e17age" does not seem to be linear correlated to response
# try to find appropiate 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
sjp.lm(fit, type = "poly", poly.term = "e17age")
## End(Not run)
```

`sjp.resid` *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.

Usage

```
sjp.resid(fit, geom.size = 2, remove.estimates = NULL, show.lines = TRUE,
  show.resid = TRUE, show.pred = TRUE, show.ci = F, prnt.plot = TRUE)
```

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	Character vector with coefficient names that indicate which estimates should be removed from the plot. 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, depending on type, a confidence interval or region is added to the plot. For frequency plots, the confidence interval for the relative frequencies are shown.
prnt.plot	logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) returns the ggplot-object with the complete plot (plot), the residual pattern (pattern) as well as the data frame that was used for setting up the ggplot-object (mydf).

Note

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

Examples

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

# plot residuals for all independent variables
sjp.resid(fit)

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

# show pattern
sjp.resid(fit, remove.estimates = c("e17age", "e42dep"))$pattern
```

`sjp.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

```
sjp.scatter(x = NULL, y = NULL, grp = NULL, title = "",
  legend.title = NULL, legend.labels = NULL, dot.labels = NULL,
  axis.titles = NULL, wrap.title = 50, wrap.legend.title = 20,
  wrap.legend.labels = 20, geom.size = 2, label.size = 3,
  geom.colors = NULL, show.axis.values = TRUE, fit.line.grps = FALSE,
  fit.line = FALSE, show.ci = FALSE, fitmethod = "lm",
  jitter.dots = FALSE, emph.dots = FALSE, auto.jitter = TRUE,
  jitter.ratio = 0.15, show.rug = FALSE, show.legend = TRUE,
  facet.grid = FALSE, prnt.plot = TRUE)
```

Arguments

`x`	Vector indicating the x positions. If not specified (i.e. if `NULL`), a range from 1 to length of `y` is used to spread the dots along the x axis.
`y`	Vector indicating the y positions. If not specified (i.e. if `NULL`), a range from 1 to length of `x` is used to spread the dots along the y axis.
`grp`	Grouping variable. If not `NULL`, the scatter plot will be grouped. See 'Examples'. Default is `NULL`, i.e. not grouping is done.
`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.
`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.
`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.

`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.

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

`geom.colors`
: user defined color for geoms. See 'Details' in `sjp.grpfrq`.

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

`fit.line.grps`
: Logical, if `TRUE`, a fitted line for each group is drawn. See `fitmethod` to change the fit method of the fitted lines.

`fit.line`
: Logical, if `TRUE`, a fitted line for the overall scatterplot is drawn. See `fitmethod` to change the fit method of the fitted line.

`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.

`fitmethod`
: By default, a linear method (`"lm"`) is used for fitting the fit lines. Possible values are for instance `"lm"`, `"glm"`, `"loess"` or `"auto"`.

`jitter.dots`
: Logical, if `TRUE`, points will be jittered (to avoid overplotting).

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

`auto.jitter`
: Logical, if `TRUE`, points will be jittered according to an overlap-estimation. A matrix of `x` and `y` values is created and the amount of cells (indicating a unique point position) is calculated. If more than 15% (see `jitter.ratio`) of the approximated amount of unique point coordinates seem to overlap, they are automatically jittered.

`jitter.ratio`
: Ratio of tolerated overlapping (see `auto.jitter`). If approximated amount of overlapping points exceed this ratio, they are automatically jittered. Default is 0.15. Valid values range between 0 and 1.

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

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

`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`.

`prnt.plot`
: logical, if `TRUE` (default), plots the results as graph. Use `FALSE` if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) returns the ggplot-object with the complete plot (`plot`) as well as the data frame that was used for setting up the ggplot-object (`data`).

See Also

sjPlot manual: sjp.scatter

Examples

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

# simple scatter plot, auto-jittering
sjp.scatter(efc$e16sex, efc$neg_c_7)

# simple scatter plot, no jittering needed
sjp.scatter(efc$c160age, efc$e17age)

# grouped scatter plot
sjp.scatter(efc$c160age, efc$e17age, efc$e42dep)

# grouped and jittered scatter plot with marginal rug plot
sjp.scatter(efc$e16sex,efc$neg_c_7, efc$c172code, show.rug = TRUE)

# grouped and labelled scatter plot, not using the auto-detection
# of labels, but instead pass labels as arguments
sjp.scatter(efc$c160age, efc$e17age, efc$e42dep,
            title = "Scatter Plot", legend.title = get_label(efc)['e42dep'],
            legend.labels = get_labels(efc)[['e42dep']],
            axis.titles = c(get_label(efc)['c160age'], get_label(efc)['e17age']),
            fit.line.grps = TRUE)

# grouped and labelled scatter plot as facets
sjp.scatter(efc$c160age,efc$e17age, efc$e42dep, fit.line.grps = TRUE,
            facet.grid = TRUE, show.ci = TRUE)

# plot residuals of fitted models
fit <- lm(neg_c_7 ~ quol_5, data = efc)
sjp.scatter(y = fit$residuals, fit.line = TRUE)

# "hide" axis titles
sjp.scatter(efc$c160age, efc$e17age, efc$e42dep, title = "",
            axis.titles = c("", ""))

# plot text labels
pl <- c(1:10)
for (i in 1:10)
  pl[i] <- paste(sample(c(0:9, letters, LETTERS), 8, replace = TRUE), collapse = "")
sjp.scatter(runif(10), runif(10), dot.labels = pl)
```

`sjp.stackfrq` *Plot stacked proportional bars*

Description

Plot items (variables) of a scale as stacked proportional bars. This function is useful when several items with identical scale/categoroies should be plotted to compare the distribution of answers.

Usage

```
sjp.stackfrq(items, title = NULL, legend.title = NULL,
  legend.labels = NULL, axis.titles = NULL, axis.labels = NULL,
  weight.by = NULL, sort.frq = NULL, wrap.title = 50, wrap.labels = 30,
  wrap.legend.title = 30, wrap.legend.labels = 28, geom.size = 0.5,
  geom.colors = "Blues", show.values = TRUE, show.n = TRUE,
  show.prc = TRUE, show.legend = TRUE, grid.breaks = 0.2,
  expand.grid = FALSE, digits = 1, vjust = "center", coord.flip = TRUE,
  prnt.plot = TRUE)
```

Arguments

`items`	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 sjp.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), 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 and 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.

prnt.plot
: logical, if TRUE (default), plots the results as graph. Use FALSE if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) 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).

Note

Thanks to Forrest Stevens for bug fixes.

See Also

- sjPlot manual: sjp.stackfrq
- sjt.stackfrq

Examples

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

---

sjp.xtab *Plot contingency tables*

Description

Plot proportional crosstables (contingency tables) of two variables as ggplot diagram.

Usage

```
sjp.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, axis.labels = NULL, legend.title = NULL,
  legend.labels = NULL, weight.by = NULL, rev.order = FALSE,
  show.values = TRUE, show.n = TRUE, show.prc = TRUE, show.total = TRUE,
  show.legend = TRUE, show.summary = FALSE, summary.pos = "r",
  string.total = "Total", wrap.title = 50, wrap.labels = 15,
  wrap.legend.title = 20, wrap.legend.labels = 20, geom.size = 0.7,
  geom.spacing = 0.1, geom.colors = "Paired", dot.size = 3,
  smooth.lines = FALSE, grid.breaks = 0.2, expand.grid = FALSE,
  ylim = NULL, vjust = "bottom", hjust = "center", y.offset = NULL,
  coord.flip = FALSE, prnt.plot = TRUE)
```

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 excact test (see fisher.test) is computed instead of chi-squared test. If the table's matrix is larger than 2x2, Fisher's excact 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.

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)

`geom.colors`
: user defined color for geoms. See 'Details' in `sjp.grpfrq`.

`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.

`prnt.plot`
: logical, if `TRUE` (default), plots the results as graph. Use `FALSE` if you don't want to plot any graphs. In either case, the ggplot-object will be returned as value.

Value

(Insisibily) returns the ggplot-object with the complete plot (`plot`) as well as the data frame that was used for setting up the ggplot-object (`mydf`).

See Also

- sjPlot manual: sjp.xtab
- `sjt.xtab`

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
sjp.xtab(x, grp)

# plot "cross tablulation" of x and y, including labels
sjp.xtab(x, grp, axis.labels = c("low", "mid", "high"),
         legend.labels = c("Grp 1", "Grp 2", "Grp 3", "Grp 4"))

# plot "cross tablulation" of x and grp
# as stacked proportional bars
sjp.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)
sjp.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)

sjp.xtab(efc$e42dep, efc$e16sex, title = efc.var['e42dep'],
         axis.labels = efc.val[['e42dep']], legend.title = efc.var['e16sex'],
         legend.labels = efc.val[['e16sex']])

sjp.xtab(efc$e16sex, efc$e42dep, title = efc.var['e16sex'],
         axis.labels = efc.val[['e16sex']], legend.title = efc.var['e42dep'],
         legend.labels = efc.val[['e42dep']])

# ------------------------------
# auto-detection of labels works here
# so no need to specify labels. For
# title-auto-detection, use NULL
# ------------------------------
sjp.xtab(efc$e16sex, efc$e42dep, title = NULL)

sjp.xtab(efc$e16sex, efc$e42dep, margin = "row",
         bar.pos = "stack", coord.flip = TRUE)
```

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("frq", "grpfrq", "xtab", "gpt", "scatter", "aov1",
  "likert", "stackfrq"))

sjtab(data, ..., fun = c("frq", "xtab", "grpmean", "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.
- `"frq"` calls `sjp.frq` or `sjt.frq`. If `data` has more than one variable, a plot for each variable in `data` is plotted.
- `"gpt"` calls `sjp.gpt`. The first three variables in `data` are used (and required) to create the plot.
- `"grpfrq"` calls `sjp.grpfrq`. The first two variables in `data` are used (and required) to create the plot.
- `"grpmean"` calls `sjt.grpmean`. The first two variables in `data` are used (and required) to create the table.
- `"likert"` calls `sjp.likert`. `data` must be a data frame with items to plot.
- `"scatter"` calls `sjp.scatter`. The first two variables in `data` are used (and required) to create the plot; if `data` also has a third variable, this is used as grouping- variable in `sjp.scatter`.
- `"stackfrq"` calls `sjp.stackfrq` or `sjt.stackfrq`. `data` must be a data frame with items to create the plot or table.
- `"xtab"` calls `sjp.xtab` or `sjt.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

```
library(dplyr)
data(efc)

# Frequency plot
sjplot(efc, e42dep, c172code, fun = "frq")

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

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

# scatter plot, grouped
efc %>%
  select(e42dep, c172code, c161sex) %>%
  sjplot(fun = "scatter")

# frequencies, as plot grid
efc %>%
  select(e42dep, c172code, e16sex, c161sex) %>%
  sjplot() %>%
  plot_grid()

# plot grouped data frame
efc %>%
  group_by(e16sex, c172code) %>%
  select(e42dep, e16sex, c172code) %>%
  sjplot(wrap.title = 100) # no line break for subtitles

## 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 theme plots or modify plot 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)
```

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.
base.theme	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.x	Angle for x-axis labels.
angle.y	Angle for y-axis labels.
inside	Logical, use TRUE to put legend inside the plotting area. See also pos.
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.

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

Examples

```
# prepare data
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
p + font_size(axis_title.x = 30)
```

---

sjt.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

```
sjt.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, no.output = FALSE, remove.spaces = TRUE)
```

Arguments

data	A vector or a data frame, for which frequencies should be printed as table.
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 "spearman" (default), "pearson" or "kendall". May be abbreviated.
title	Table caption, as character vector.
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, 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.
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 shown 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	Numeric, amount of digits after decimal point when rounding estimates and values.
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 specifiy the inital 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 defauilt, this argument is NULL and the diagnal cells are empty.
CSS	A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.
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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

Details

See 'Details' in sjt.frq.

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.

Additionally, see 'Note' in sjt.frq.

See Also

- sjPlot manual: sjt.corr
- sjp.corr

Examples

```
## Not run:
# plot correlation matrix using circles
sjt.corr(mydf)

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

# retrieve variable and value labels
```

```
varlabs <- get_label(efc)

# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c83cop2")
# recveive 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. See example from "sjp.pca".
sjt.corr(mydf, p.numeric = TRUE)

# auto-detection of labels, only lower triangle
sjt.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
sjt.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
sjt.corr(efc[, c(start:end)], triangle = "lower",val.rm = 0.3,
         CSS = list(css.valueremove = 'color:blue;'))
## End(Not run)
```

`sjt.fa` *Summary of factor analysis as HTML table*

Description

Performes 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

```
sjt.fa(data, rotation = c("promax", "varimax"), method = c("ml", "minres",
  "wls", "gls", "pa", "minchi", "minrank"), nmbr.fctr = NULL,
  fctr.load.tlrn = 0.1, title = "Factor Analysis", var.labels = NULL,
  wrap.labels = 40, show.cronb = TRUE, show.comm = FALSE,
  altr.row.col = FALSE, digits = 2, CSS = NULL, encoding = NULL,
  file = NULL, use.viewer = TRUE, no.output = FALSE,
  remove.spaces = TRUE)
```

Arguments

data	A data frame that should be used to compute a FA, or a fa object.
rotation	Rotation of the factor loadings. May be "varimax" for orthogonal rotation or "promax" for oblique transformation (default). Requires the "GPArotation" package.
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.
nmbr.fctr	Number of factors used for calculating the rotation. By default, this value is NULL and the amount of factors is calculated according to a parallel analysis.
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	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.
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 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.
show.comm	Logical, if TRUE, show the communality column in the table.
altr.row.col	Logical, if TRUE, alternating rows are highlighted with a light gray background color.
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
CSS	A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.
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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

Details

See 'Details' in sjt.frq.

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.

Note

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

Examples

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

# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# recveive last item of COPE-index scale
```

```
end <- which(colnames(efc) == "c90cop9")
# auto-detection of labels
sjt.fa(efc[, start:end])
## End(Not run)
```

---

sjt.frq *Summary of frequencies as HTML table*

Description

Shows (multiple) frequency tables as HTML file, or saves them as file.

Usage

```
sjt.frq(data, weight.by = NULL, title.wtd.suffix = " (weighted)",
  title = NULL, value.labels = NULL, sort.frq = c("none", "asc", "desc"),
  altr.row.col = FALSE, string.val = "value", string.cnt = "N",
  string.prc = "raw %", string.vprc = "valid %",
  string.cprc = "cumulative %", string.na = "missings", emph.md = FALSE,
  emph.quart = FALSE, show.summary = TRUE, show.skew = FALSE,
  show.kurtosis = FALSE, skip.zero = "auto", ignore.strings = TRUE,
  auto.group = NULL, auto.grp.strings = TRUE, max.string.dist = 3,
  digits = 2, CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, remove.spaces = TRUE)
```

Arguments

data	A vector or a data frame, for which frequencies should be printed as table.
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.
title	Table caption, as character vector.
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.
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.
altr.row.col	Logical, if TRUE, alternating rows are highlighted with a light gray background color.
string.val	Character label for the very first table column containing the values (see value.labels).
string.cnt	Character label for the first table data column containing the counts. Default is "N".

`string.prc` Character label for the second table data column containing the raw percentages. Default is `"raw %"`.

`string.vprc` Character label for the third data table column containing the valid percentages, i.e. the count percentage value exluding possible missing values.

`string.cprc` Character label for the last table data column containing the cumulative percentages.

`string.na` Character label for the last table data row containing missing values.

`emph.md` Logical, if `TRUE`, the table row indicating the median value will be emphasized.

`emph.quart` Logical, if `TRUE`, the table row indicating the lower and upper quartiles will be emphasized.

`show.summary` Logical, if `TRUE` (default), a summary row with total and valid N as well as mean and standard deviation is shown.

`show.skew` Logical, if `TRUE`, the variable's skewness is added to the summary. The skewness is retrieved from the `describe`-function of the **psych**-package and indicated by a lower case Greek gamma.

`show.kurtosis` Logical, if `TRUE`, the variable's kurtosis is added to the summary. The kurtosis is retrieved from the `describe`-function of the **psych**-package and indicated by a lower case Greek omega.

`skip.zero` Logical, if `TRUE`, rows with only zero-values are not printed (e.g. if a variable has values or levels 1 to 8, and levels / values 4 to 6 have no counts, these values would not be printed in the table). Use `FALSE` to print also zero-values, or use `"auto"` (default) to detect whether it makes sense or not to print zero-values (e.g., a variable "age" with values from 10 to 100, where at least 25 percent of all possible values have no counts, zero-values would be skipped automatically).

`ignore.strings` Logical, if `TRUE` (default), character vectors / string variables will be removed from `data` before frequency tables are computed.

`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.

`auto.grp.strings` Logical, if `TRUE` (default), string values in character vectors (string variables) are automatically grouped based on their similarity. The similarity is estimated with the **stringdist**-package. You can specify a distance-measure via `max.string.dist` argument. This argument only applies if `ignore.strings` is `FALSE`.

`max.string.dist` Numeric, the allowed distance of string values in a character vector, which indicates when two string values are merged because they are considered as close enough. See `auto.grp.strings`.

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

`CSS` A `list` with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in `sjt.frq`.

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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

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-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.

See further examples in this package-vignette.

Value

Invisibly returns

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

- each frequency table as web page content (`page.content.list`),
- 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`).

See Also

- sjPlot manual: sjt.frq
- `sjp.frq`

Examples

```
## Not run:
# load sample data
library(sjmisc)
data(efc)

# show frequencies of "e42dep" in RStudio Viewer Pane
# or default web browser
sjt.frq(efc$e42dep)

# plot and show frequency table of "e42dep" with labels
sjt.frq(efc$e42dep, title = "Dependency",
        value.labels = c("independent", "slightly dependent",
                         "moderately dependent", "severely dependent"))

# plot frequencies of e42dep, e16sex and c172code in one HTML file
# and show table in RStudio Viewer Pane or default web browser
# Note that value.labels of multiple variables have to be
# list-objects
sjt.frq(data.frame(efc$e42dep, efc$e16sex, efc$c172code),
        title = c("Dependency", "Gender", "Education"),
        value.labels = list(c("independent", "slightly dependent",
                              "moderately dependent", "severely dependent"),
                            c("male", "female"), c("low", "mid", "high")))

# auto-detection of labels
sjt.frq(data.frame(efc$e42dep, efc$e16sex, efc$c172code))

# plot larger scale including zero-counts
# indicating median and quartiles
sjt.frq(efc$neg_c_7, emph.md = TRUE, emph.quart = TRUE)
```

```
# sort frequencies
sjt.frq(efc$e42dep, sort.frq = "desc")

# User defined style sheet
sjt.frq(efc$e42dep,
        CSS = list(css.table = "border: 2px solid;",
                   css.tdata = "border: 1px solid;",
                   css.firsttablecol = "color:#003399; font-weight:bold;"))
## End(Not run)
```

`sjt.glm` *Summary of generalized linear models as HTML table*

Description

Summarizes (multiple) fitted generalized linear models (odds ratios, ci, p-values...) as HTML table, or saves them as file. The fitted models may have different predictors, e.g. when comparing different stepwise fitted models.

Usage

```
sjt.glm(..., pred.labels = NULL, depvar.labels = NULL,
  remove.estimates = NULL, group.pred = TRUE, exp.coef = TRUE,
  p.numeric = TRUE, emph.p = FALSE, p.zero = FALSE, robust = FALSE,
  separate.ci.col = TRUE, newline.ci = TRUE, show.ci = TRUE,
  show.se = FALSE, show.header = FALSE, show.col.header = TRUE,
  show.r2 = FALSE, show.icc = FALSE, show.re.var = FALSE,
  show.loglik = FALSE, show.aic = FALSE, show.aicc = FALSE,
  show.dev = FALSE, show.hoslem = FALSE, show.family = FALSE,
  show.chi2 = FALSE, string.pred = "Predictors",
  string.dv = "Dependent Variables", string.interc = "(Intercept)",
  string.obs = "Observations", string.est = NULL, string.ci = "CI",
  string.se = "std. Error", string.p = "p",
  ci.hyphen = " &ndash; ", digits.est = 2, digits.p = 3,
  digits.ci = 2, digits.se = 2, digits.summary = 3, cell.spacing = 0.2,
  cell.gpr.indent = 0.6, sep.column = TRUE, CSS = NULL, encoding = NULL,
  file = NULL, use.viewer = TRUE, no.output = FALSE,
  remove.spaces = TRUE)
```

Arguments

`...`	One or more fitted generalized linear (mixed) models.
`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. If `NULL`, variable labels are set based on label attributes (see `get_label`). If `pred.labels = ""`, column names (vector names) are used as predictor labels. See 'Examples'.

depvar.labels Character vector with labels of dependent variables of all fitted models. See 'Examples'.

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. *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.

group.pred Logical, if `TRUE` (default), automatically groups table rows with factor levels of same factor, i.e. predictors of type `factor` will be grouped, if the factor has more than two levels. Grouping means that a separate headline row is inserted to the table just before the predictor values.

exp.coef Logical, if `TRUE` (default), regression coefficients and confidence intervals are exponentiated. Use `FALSE` for non-exponentiated coefficients (log-odds) as provided by the `summary` function.

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

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

p.zero logical, if `TRUE`, p-values have a leading 0 before the period (e.g. *0.002*), else p-values start with a period and without a zero (e.g. *.002*).

robust Logical, if `TRUE`, robust standard errors and confidence intervals will be reported. Computation of robust standard errors is based on the `robust`-function in the **sjstats**-package.

separate.ci.col
Logical, if `TRUE`, the CI values are shown in a separate table column. Default is `FALSE`.

newline.ci Logical, if `TRUE` and `separate.ci.col = FALSE`, inserts a line break between estimate and CI values. If `FALSE`, CI values are printed in the same line as estimate values.

show.ci Logical, if `TRUE` (default), the confidence intervall is also printed to the table. Use `FALSE` to omit the CI in the table.

show.se Logical, if `TRUE`, the standard errors are also printed. Default is `FALSE`.

show.header Logical, if `TRUE`, the header strings `string.pred` and `string.dv` are shown. By default, they're hidden.

show.col.header
Logical, if `TRUE` (default), the table data columns have a headline with abbreviations for estimates, std. beta-values, confidence interval and p-values.

show.r2 Logical, if `TRUE` (default), the pseudo R2 values for each model are printed in the model summary. R2cs is the Cox-Snell-pseudo R-squared value, R2n is Nagelkerke's pseudo R-squared value and `D` is Tjur's Coefficient of Discrimination (see `cod`).

show.icc	Logical, if TRUE, the intra-class-correlation for each model is printed in the model summary. Only applies to mixed models.
show.re.var	Logical, if TRUE, the variance parameters for the random effects for each model are printed in the model summary. Only applies to mixed models. For details output, see 'Note' in icc.
show.loglik	Logical, if TRUE, the Log-Likelihood for each model is printed in the model summary. Default is FALSE.
show.aic	Logical, if TRUE, the AIC value for each model is printed in the model summary. Default is FALSE.
show.aicc	Logical, if TRUE, the second-order AIC value for each model is printed in the model summary. Default is FALSE.
show.dev	Logical, if TRUE, the deviance for each model is printed in the model summary.
show.hoslem	Logical, if TRUE, a Hosmer-Lemeshow-Goodness-of-fit-test is performed. A well-fitting model shows no significant difference between the model and the observed data, i.e. the reported p-values should be greater than 0.05.
show.family	Logical, if TRUE, the family object and link function for each fitted model are printed. Can be used in case you want to compare models with different link functions and same predictors and response, to decide which model fits best. See family for more details. It is recommended to inspect the model AIC (see show.aic) to get a decision help for which model to choose.
show.chi2	Logical, if TRUE, the p-value of the chi-squared value for each model's residual deviance against the null deviance is printed in the model summary. Default is FALSE. A well-fitting model with predictors should significantly differ from the null-model (without predictors), thus, a p-value less than 0.05 indicates a good model-fit.
string.pred	Character vector,used as headline for the predictor column. Default is "Predictors".
string.dv	Character vector, used as headline for the dependent variable columns. Default is "Dependent Variables".
string.interc	Character vector, used as headline for the Intercept row. Default is "Intercept".
string.obs	character vector, used in the summary row for the count of observation (cases). Default is "Observations".
string.est	Character vector, used for the column heading of estimates.
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.p	Character vector, used for the column heading of p values. Default is "p".
ci.hyphen	Character vector, indicating the hyphen for confidence interval range. May be an HTML entity. See 'Examples'.
digits.est	Amount of decimals for estimates
digits.p	Amount of decimals for p-values
digits.ci	Amount of decimals for confidence intervals

digits.se — Amount of decimals for standard error

digits.summary — Amount of decimals for values in model summary

cell.spacing — Numeric, inner padding of table cells. By default, this value is 0.2 (unit is cm), which is suitable for viewing the table. Decrease this value (0.05 to 0.1) if you want to import the table into Office documents. This is a convenient argument for the CSS argument for changing cell spacing, which would be: `CSS = list(css.thead = "padding:0.2cm;", css.tdata = "padding:0.2cm;")`.

cell.gpr.indent — Indent for table rows with grouped factor predictors. Only applies if `group.pred = TRUE`.

sep.column — Logical, if TRUE, an empty table column is added after each model column, to add margins between model columns. By default, this column will be added to the output; however, when copying tables to office applications, it might be helpful not to add this separator column when modifying the table layout.

CSS — A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.

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 openend 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.

no.output — Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.

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.

Details

See 'Details' in sjt.frq.

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 `exp.coef = TRUE` and Odds Ratios are reported, standard errors for generalized linear (mixed) models are *not* on the untransformed scale, as shown in the `summary()`-method. Rather, `sjt.glm()` uses adjustments according to the delta method for approximating standard errors of transformed regression parameters (see `se`). If `exp.coef = FALSE` and log-Odds Ratios are reported, the standard errors are untransformed.

Futhermore, see 'Notes' in `sjt.frq`.

Examples

```
# prepare dummy variables for binary logistic regression
swiss$y1 <- ifelse(swiss$Fertility < median(swiss$Fertility), 0, 1)
swiss$y2 <- ifelse(swiss$Infant.Mortality < median(swiss$Infant.Mortality), 0, 1)
swiss$y3 <- ifelse(swiss$Agriculture < median(swiss$Agriculture), 0, 1)

# Now fit the models. Note that both models share the same predictors
# and only differ in their dependent variable (y1, y2 and y3)
fitOR1 <- glm(y1 ~ Education + Examination + Catholic, data = swiss,
              family = binomial(link = "logit"))
fitOR2 <- glm(y2 ~ Education + Examination + Catholic, data = swiss,
              family = binomial(link = "logit"))
fitOR3 <- glm(y3 ~ Education + Examination + Catholic, data = swiss,
              family = binomial(link = "logit"))

## Not run:
# open HTML-table in RStudio Viewer Pane or web browser
sjt.glm(fitOR1, fitOR2,
        depvar.labels = c("Fertility", "Infant Mortality"),
        pred.labels = c("Education", "Examination", "Catholic"),
        ci.hyphen = " to ")

# open HTML-table in RStudio Viewer Pane or web browser,
# integrate CI in OR column
sjt.glm(fitOR1, fitOR2, fitOR3,
        pred.labels = c("Education", "Examination", "Catholic"),
        separate.ci.col = FALSE)

# open HTML-table in RStudio Viewer Pane or web browser,
# indicating p-values as numbers and printing CI in a separate column
sjt.glm(fitOR1, fitOR2, fitOR3,
        depvar.labels = c("Fertility", "Infant Mortality", "Agriculture"),
        pred.labels = c("Education", "Examination", "Catholic"))

# ------------------------------------------
# User defined style sheet
# ------------------------------------------
sjt.glm(fitOR1, fitOR2, fitOR3,
        depvar.labels = c("Fertility", "Infant Mortality", "Agriculture"),
        pred.labels = c("Education", "Examination", "Catholic"),
        show.header = TRUE,
```

```
        CSS = list(css.table = "border: 2px solid;",
                   css.tdata = "border: 1px solid;",
                   css.depvarhead = "color:#003399;"))

# ------------------------------------------
# Compare models with different link functions,
# but same predictors and response
# ------------------------------------------
library(sjmisc)
# load efc sample data
data(efc)
# dichtomozize service usage by "service usage yes/no"
efc$services <- sjmisc::dicho(efc$tot_sc_e, dich.by = 0, as.num = TRUE)
# fit 3 models with different link-functions
fit1 <- glm(services ~ neg_c_7 + c161sex + e42dep,
            data = efc, family = binomial(link = "logit"))
fit2 <- glm(services ~ neg_c_7 + c161sex + e42dep,
            data = efc, family = binomial(link = "probit"))
fit3 <- glm(services ~ neg_c_7 + c161sex + e42dep,
            data = efc, family = poisson(link = "log"))

# compare models
sjt.glm(fit1, fit2, fit3, string.est = "Estimate",
        show.aic = TRUE, show.family = TRUE)

# ------------------------------------------
# Change style of p-values and CI-appearance
# ------------------------------------------
# open HTML-table in RStudio Viewer Pane or web browser,
# table indicating p-values as stars
sjt.glm(fit1, fit2, fit3, p.numeric = FALSE,
        show.aic = TRUE, show.family = TRUE)

# open HTML-table in RStudio Viewer Pane or web browser,
# indicating p-values as stars and integrate CI in OR column
sjt.glm(fit1, fit2, fit3, p.numeric = FALSE, separate.ci.col = FALSE,
        show.aic = TRUE, show.family = TRUE, show.r2 = TRUE)

# ---------------------------------
# automatic grouping of predictors
# ---------------------------------
library(sjmisc)
# load efc sample data
data(efc)
# dichtomozize service usage by "service usage yes/no"
efc$services <- sjmisc::dicho(efc$tot_sc_e, dich.by = 0, as.num = TRUE)
# make dependency categorical
efc$e42dep <- to_factor(efc$e42dep)
# fit model with "grouped" predictor
fit <- glm(services ~ neg_c_7 + c161sex + e42dep, data = efc)

# automatic grouping of categorical predictors
sjt.glm(fit)
```

```
# ---------------------------------
# compare models with different predictors
# ---------------------------------
fit2 <- glm(services ~ neg_c_7 + c161sex + e42dep + c12hour, data = efc)
fit3 <- glm(services ~ neg_c_7 + c161sex + e42dep + c12hour + c172code,
            data = efc)

# print models with different predictors
sjt.glm(fit, fit2, fit3)

efc$c172code <- to_factor(efc$c172code)
fit2 <- glm(services ~ neg_c_7 + c161sex + c12hour, data = efc)
fit3 <- glm(services ~ neg_c_7 + c161sex + c172code, data = efc)

# print models with different predictors
sjt.glm(fit, fit2, fit3, group.pred = FALSE)
## End(Not run)
```

---

sjt.glmer *Summary of generalized linear mixed models as HTML table*

Description

Summarizes (multiple) fitted generalized linear mixed models (odds ratios, ci, p-values...) as HTML table, or saves them as file. The fitted models may have different predictors, e.g. when comparing different stepwise fitted models.

Usage

```
sjt.glmer(..., pred.labels = NULL, depvar.labels = NULL,
  remove.estimates = NULL, group.pred = FALSE, exp.coef = TRUE,
  p.numeric = TRUE, emph.p = FALSE, p.zero = FALSE,
  separate.ci.col = TRUE, newline.ci = TRUE, show.ci = TRUE,
  show.se = FALSE, show.header = FALSE, show.col.header = TRUE,
  show.r2 = FALSE, show.icc = TRUE, show.re.var = TRUE,
  show.loglik = FALSE, show.aic = FALSE, show.aicc = FALSE,
  show.dev = TRUE, show.hoslem = FALSE, show.family = FALSE,
  string.pred = "Predictors", string.dv = "Dependent Variables",
  string.interc = "(Intercept)", string.obs = "Observations",
  string.est = NULL, string.ci = "CI", string.se = "std. Error",
  string.p = "p", ci.hyphen = " &ndash; ", digits.est = 2,
  digits.p = 3, digits.ci = 2, digits.se = 2, digits.summary = 3,
  cell.spacing = 0.2, cell.gpr.indent = 0.6, sep.column = TRUE,
  CSS = NULL, encoding = NULL, file = NULL, use.viewer = TRUE,
  no.output = FALSE, remove.spaces = TRUE)
```

Arguments

...	One or more fitted generalized linear (mixed) models.
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. If NULL, variable labels are set based on label attributes (see get_label). If pred.labels = "", column names (vector names) are used as predictor labels. See 'Examples'.
depvar.labels	Character vector with labels of dependent variables of all fitted models. See 'Examples'.
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. *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.
group.pred	Logical, if TRUE (default), automatically groups table rows with factor levels of same factor, i.e. predictors of type factor will be grouped, if the factor has more than two levels. Grouping means that a separate headline row is inserted to the table just before the predictor values.
exp.coef	Logical, if TRUE (default), regression coefficients and confidence intervals are exponentiated. Use FALSE for non-exponentiated coefficients (log-odds) as provided by the summary function.
p.numeric	Logical, if TRUE, the p-values are printed as numbers. If FALSE (default), asterisks are used.
emph.p	Logical, if TRUE, significant p-values are shown bold faced.
p.zero	logical, if TRUE, p-values have a leading 0 before the period (e.g. *0.002*), else p-values start with a period and without a zero (e.g. *.002*).
separate.ci.col	Logical, if TRUE, the CI values are shown in a separate table column. Default is FALSE.
newline.ci	Logical, if TRUE and separate.ci.col = FALSE, inserts a line break between estimate and CI values. If FALSE, CI values are printed in the same line as estimate values.
show.ci	Logical, if TRUE (default), the confidence intervall is also printed to the table. Use FALSE to omit the CI in the table.
show.se	Logical, if TRUE, the standard errors are also printed. Default is FALSE.
show.header	Logical, if TRUE, the header strings string.pred and string.dv are shown. By default, they're hidden.
show.col.header	Logical, if TRUE (default), the table data columns have a headline with abbreviations for estimates, std. beta-values, confidence interval and p-values.

show.r2	Logical, if TRUE (default), the pseudo R2 values for each model are printed in the model summary. R2cs is the Cox-Snell-pseudo R-squared value, R2n is Nagelkerke's pseudo R-squared value and D is Tjur's Coefficient of Discrimination (see cod).
show.icc	Logical, if TRUE, the intra-class-correlation for each model is printed in the model summary. Only applies to mixed models.
show.re.var	Logical, if TRUE, the variance parameters for the random effects for each model are printed in the model summary. Only applies to mixed models. For details output, see 'Note' in icc.
show.loglik	Logical, if TRUE, the Log-Likelihood for each model is printed in the model summary. Default is FALSE.
show.aic	Logical, if TRUE, the AIC value for each model is printed in the model summary. Default is FALSE.
show.aicc	Logical, if TRUE, the second-order AIC value for each model is printed in the model summary. Default is FALSE.
show.dev	Logical, if TRUE, the deviance for each model is printed in the model summary.
show.hoslem	Logical, if TRUE, a Hosmer-Lemeshow-Goodness-of-fit-test is performed. A well-fitting model shows no significant difference between the model and the observed data, i.e. the reported p-values should be greater than 0.05.
show.family	Logical, if TRUE, the family object and link function for each fitted model are printed. Can be used in case you want to compare models with different link functions and same predictors and response, to decide which model fits best. See family for more details. It is recommended to inspect the model AIC (see show.aic) to get a decision help for which model to choose.
string.pred	Character vector,used as headline for the predictor column. Default is "Predictors".
string.dv	Character vector, used as headline for the dependent variable columns. Default is "Dependent Variables".
string.interc	Character vector, used as headline for the Intercept row. Default is "Intercept".
string.obs	character vector, used in the summary row for the count of observation (cases). Default is "Observations".
string.est	Character vector, used for the column heading of estimates.
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.p	Character vector, used for the column heading of p values. Default is "p".
ci.hyphen	Character vector, indicating the hyphen for confidence interval range. May be an HTML entity. See 'Examples'.
digits.est	Amount of decimals for estimates
digits.p	Amount of decimals for p-values
digits.ci	Amount of decimals for confidence intervals
digits.se	Amount of decimals for standard error

digits.summary Amount of decimals for values in model summary

cell.spacing Numeric, inner padding of table cells. By default, this value is 0.2 (unit is cm), which is suitable for viewing the table. Decrease this value (0.05 to 0.1) if you want to import the table into Office documents. This is a convenient argument for the CSS argument for changing cell spacing, which would be: CSS = list(css.thead = "padding:0.2cm;", css.tdata = "padding:0.2cm;").

cell.gpr.indent Indent for table rows with grouped factor predictors. Only applies if group.pred = TRUE.

sep.column Logical, if TRUE, an empty table column is added after each model column, to add margins between model columns. By default, this column will be added to the output; however, when copying tables to office applications, it might be helpful not to add this separator column when modifying the table layout.

CSS A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.

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 openend 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.

no.output Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.

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.

Details

See 'Details' in sjt.frq.

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

Computation of p-values (if necessary) is based on normal-distribution assumption, treating the t-statistics as Wald z-statistics.

The variance components of the random parts (see `show.re.var`) are denoted like:

- between-group-variance: tau-zero-zero
- random-slope-intercept-correlation: rho-zero-one

Standard errors for generalized linear (mixed) models are *not* the regular standard errors on the untransformed scale, as shown in the `summary()`-method. Rather, `sjt.glmer()` uses adjustments according to the delta method for approximating standard errors of transformed regression parameters (see `se`).

Futhermore, see 'Notes' in `sjt.frq`.

Examples

```
## Not run:
library(lme4)
library(sjmisc)
data(efc)

# create binary response
efc$hi_qol <- dicho(efc$quol_5)
# prepare group variable
efc$grp = as.factor(efc$e15relat)
levels(x = efc$grp) <- get_labels(efc$e15relat)
# data frame for fitted model
mydf <- data.frame(hi_qol = to_factor(efc$hi_qol),
                   sex = to_factor(efc$c161sex),
                   c12hour = efc$c12hour,
                   neg_c_7 = efc$neg_c_7,
                   education = to_factor(efc$c172code),
                   grp = efc$grp)

# fit glmer
fit1 <- glmer(hi_qol ~ sex + c12hour + neg_c_7 + (1|grp),
              data = mydf, family = binomial("logit"))
fit2 <- glmer(hi_qol ~ sex + c12hour + neg_c_7 + education + (1|grp),
              data = mydf, family = binomial("logit"))

# print summary table
sjt.glmer(fit1, fit2, ci.hyphen = " to ")

# print summary table, using different table layout
sjt.glmer(fit1, fit2, show.aic = TRUE, show.ci = FALSE,
          show.se = TRUE, p.numeric = FALSE)

# print summary table
sjt.glmer(fit1, fit2, pred.labels = c("Elder's gender (female)",
```

```
              "Hours of care per week", "Negative Impact",
              "Educational level (mid)", "Educational level (high)"))

# use vector names as predictor labels
sjt.glmer(fit1, fit2, pred.labels = "")
## End(Not run)
```

`sjt.grpmean` *Deprecated functions*

Description

A list of deprecated functions.

Usage

```
sjt.grpmean(...)

sjt.df(...)

sjt.mwu(...)
```

Arguments

`...`	Not used.

Value

Nothing.

`sjt.itemanalysis` *Summary of item analysis of an item scale as HTML table*

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:

- kurstosis
- Shapiro-Wilk Normality Test

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

Usage

```
sjt.itemanalysis(df, factor.groups = NULL, factor.groups.titles = "auto",
  scale = FALSE, min.valid.rowmean = 2, altr.row.col = TRUE,
  sort.column = NULL, show.shapiro = FALSE, show.kurtosis = FALSE,
  show.corr.matrix = TRUE, CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, remove.spaces = TRUE)
```

Arguments

`df`	A data frame with items.
`factor.groups`	If not `NULL`, `df` will be splitted 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`. 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 `reliab_test`). 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.
`altr.row.col`	Logical, if `TRUE`, alternating rows are highlighted with a light gray background color.
`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.

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. For more details, see this package-vignette, or 'Details' in sjt.frq.

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 openend 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.

no.output — Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.

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.

Details

See 'Details' in sjt.frq.

Value

Invisibly returns

- df.list: List of data frames with the item analysis for each sub.group (or complete, if factor.groups was NULL)
- 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.
- 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.

- `cronbach.values`: List of Cronbach's Alpha values for the overall item scale for each sub-group.
- `knitr.list`: List of html-tables with inline-css for use with knitr for each table (sub-group)
- `knitr`: html-table of all complete output with inline-css for use with knitr
- `complete.page`: Complete html-output.

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

Note

- The *Shapiro-Wilk Normality Test* (see column `W(p)`) tests if an item has a distribution that is significantly different from normal.
- *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 *item discrimination*, acceptable values are 0.20 or higher; the closer to 1.00 the better. See `reliab_test` for more details.
- In case the total *Cronbach's Alpha* value is below the acceptable cut-off of 0.7 (mostly if an index has few items), the *mean inter-item-correlation* is an alternative measure to indicate acceptability. Satisfactory range lies between 0.2 and 0.4. See also `mic`.

See 'Notes' in `sjt.frq`.

References

- Jorion N, Self B, James K, Schroeder L, DiBello L, Pellegrino J (2013) Classical Test Theory Analysis of the Dynamics Concept Inventory. (web)
- Briggs SR, Cheek JM (1986) The role of factor analysis in the development and evaluation of personality scales. Journal of Personality, 54(1), 106-148. doi: 10.1111/j.14676494.1986.tb00391.x
- McLean S et al. (2013) Stigmatizing attitudes and beliefs about bulimia nervosa: Gender, age, education and income variability in a community sample. International Journal of Eating Disorders. doi: 10.1002/eat.22227
- Trochim WMK (2008) Types of Reliability. (web)

See Also

sjPlot manual: sjt.itemanalysis

Examples

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

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

```
# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# recveive 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:
sjt.itemanalysis(mydf)

# auto-detection of labels
sjt.itemanalysis(efc[, start:end])

# Compute PCA on Cope-Index, and perform a
# item analysis for each extracted factor.
factor.groups <- sjt.pca(mydf, no.output = TRUE)$factor.index
sjt.itemanalysis(mydf, factor.groups)
## End(Not run)
```

---

sjt.lm *Summary of linear regression as HTML table*

Description

Summarizes (multiple) fitted linear models (coefficients, std. beta values etc.) as HTML table, or saves them as file. The fitted models may have different predictors, e.g. when comparing different stepwise fitted models. This function also supports panel models fitted with the plm-function from the **plm**-package and generalized least squares models fitted with the gls-function from the **nlme**-package.

Usage

```
sjt.lm(..., pred.labels = NULL, depvar.labels = NULL,
  remove.estimates = NULL, group.pred = TRUE, p.numeric = TRUE,
  emph.p = FALSE, p.zero = FALSE, p.kr = TRUE, robust = FALSE,
  separate.ci.col = TRUE, newline.ci = TRUE, show.est = TRUE,
  show.std = NULL, show.ci = TRUE, show.se = FALSE, show.header = FALSE,
  show.col.header = TRUE, show.r2 = TRUE, show.icc = FALSE,
  show.re.var = FALSE, show.fstat = FALSE, show.aic = FALSE,
  show.aicc = FALSE, show.dev = FALSE, string.pred = "Predictors",
  string.dv = "Dependent Variables", string.interc = "(Intercept)",
  string.obs = "Observations", string.est = "B", string.std = "std. Beta",
  string.ci = "CI", string.se = "std. Error", string.p = "p",
  ci.hyphen = " &ndash; ", minus.sign = "&#45;", digits.est = 2,
  digits.std = 2, digits.p = 3, digits.ci = 2, digits.se = 2,
```

```
  digits.summary = 3, cell.spacing = 0.2, cell.gpr.indent = 0.6,
  sep.column = TRUE, CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, remove.spaces = TRUE)
```

Arguments

`...`	One or more fitted linear (mixed) models.
`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. If `NULL`, variable labels are set based on label attributes (see `get_label`). If `pred.labels = ""`, column names (vector names) are used as predictor labels. See 'Examples'.
`depvar.labels`	Character vector with labels of dependent variables of all fitted models. See 'Examples'.
`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. *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.
`group.pred`	Logical, if `TRUE` (default), automatically groups table rows with factor levels of same factor, i.e. predictors of type `factor` will be grouped, if the factor has more than two levels. Grouping means that a separate headline row is inserted to the table just before the predictor values.
`p.numeric`	Logical, if `TRUE`, the p-values are printed as numbers. If `FALSE` (default), asterisks are used.
`emph.p`	Logical, if `TRUE`, significant p-values are shown bold faced.
`p.zero`	logical, if `TRUE`, p-values have a leading 0 before the period (e.g. *0.002*), else p-values start with a period and without a zero (e.g. *.002*).
`p.kr`	logical, if `TRUE`, p-value estimation is based on conditional F-tests with Kenward-Roger approximation for the df. Caution: Computation may take very long time for large samples!
`robust`	Logical, if `TRUE`, robust standard errors and confidence intervals will be reported. Computation of robust standard errors is based on the `robust`-function in the **sjstats**-package.
`separate.ci.col`	Logical, if `TRUE`, the CI values are shown in a separate table column. Default is `FALSE`.
`newline.ci`	Logical, if `TRUE` and `separate.ci.col = FALSE`, inserts a line break between estimate and CI values. If `FALSE`, CI values are printed in the same line as estimate values.
`show.est`	Logical, if `TRUE` (default), the estimates are printed.
`show.std`	Indicates whether standardized beta-coefficients should also printed, and if yes, which type of standardization is done. See 'Details'.

show.ci — Logical, if TRUE (default), the confidence intervall is also printed to the table. Use FALSE to omit the CI in the table.

show.se — Logical, if TRUE, the standard errors are also printed. Default is FALSE.

show.header — Logical, if TRUE, the header strings string.pred and string.dv are shown. By default, they're hidden.

show.col.header — Logical, if TRUE (default), the table data columns have a headline with abbreviations for estimates, std. beta-values, confidence interval and p-values.

show.r2 — Logical, if TRUE (default), the R2 and adjusted R2 values for each model are printed in the model summary. For linear mixed models, the R2 and Omega-squared values are printed (see r2 for details).

show.icc — Logical, if TRUE, the intra-class-correlation for each model is printed in the model summary. Only applies to mixed models.

show.re.var — Logical, if TRUE, the variance parameters for the random effects for each model are printed in the model summary. Only applies to mixed models. For details output, see 'Note' in icc.

show.fstat — Logical, if TRUE, the F-statistics for each model is printed in the model summary. Default is FALSE. This argument does not apply to sjt.lmer.

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

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

show.dev — Logical, if TRUE, the deviance for each model is printed in the model summary.

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

string.dv — Character vector, used as headline for the dependent variable columns. Default is "Dependent Variables".

string.interc — Character vector, used as headline for the Intercept row. Default is "Intercept".

string.obs — character vector, used in the summary row for the count of observation (cases). Default is "Observations".

string.est — Character vector, used for the column heading of estimates.

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

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'.

digits.est — Amount of decimals for estimates

digits.std	Amount of decimals for standardized beta
digits.p	Amount of decimals for p-values
digits.ci	Amount of decimals for confidence intervals
digits.se	Amount of decimals for standard error
digits.summary	Amount of decimals for values in model summary
cell.spacing	Numeric, inner padding of table cells. By default, this value is 0.2 (unit is cm), which is suitable for viewing the table. Decrease this value (0.05 to 0.1) if you want to import the table into Office documents. This is a convenient argument for the CSS argument for changing cell spacing, which would be: CSS = list(css.thead = "padding:0.2cm;", css.tdata = "padding:0.2cm;").
cell.gpr.indent	Indent for table rows with grouped factor predictors. Only applies if group.pred = TRUE.
sep.column	Logical, if TRUE, an empty table column is added after each model column, to add margins between model columns. By default, this column will be added to the output; however, when copying tables to office applications, it might be helpful not to add this separator column when modifying the table layout.
CSS	A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.
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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

Details

Concerning the show.std argument, show.std = "std" will print normal standardized estimates. For show.std = "std2", however, 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. This type of standardization uses the standardize-function from the **arm**-package. 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

standardized estimats should not be printed.

Furthermore, see 'Details' in sjt.frq.

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

See 'Note' in sjt.frq.

See Also

sjPlot manual: sjt.lm

Examples

```
## Not run:
# Now fit the models. Note that both models share the same predictors
# and only differ in their dependent variable. See examples of stepwise
# models below at the end.
library(sjmisc)
data(efc)

# fit first model
fit1 <- lm(barthtot ~ c160age + c12hour + c161sex + c172code, data = efc)
# fit second model
fit2 <- lm(neg_c_7 ~ c160age + c12hour + c161sex + c172code, data = efc)

# create and open HTML-table in RStudio Viewer Pane or web browser
# note that we don't need to specify labels for the predictors,
# because these are automatically read
sjt.lm(fit1, fit2)

# create and open HTML-table in RStudio Viewer Pane or web browser
# in the following examples, we set labels via argument
sjt.lm(fit1, fit2,
       depvar.labels = c("Barthel-Index", "Negative Impact"),
       pred.labels = c("Carer's Age", "Hours of Care",
                       "Carer's Sex", "Educational Status"))

# use vector names as labels
sjt.lm(fit1, fit2, pred.labels = "")
```

```
# show HTML-table, indicating p-values as asterisks
sjt.lm(fit1, fit2, show.std = TRUE, p.numeric = FALSE)

# create and open HTML-table in RStudio Viewer Pane or web browser,
# integrate CI in estimate column
sjt.lm(fit1, fit2, separate.ci.col = FALSE)

# show HTML-table, indicating p-values as numbers
# and printing CI in a separate column
sjt.lm(fit1, fit2, show.std = TRUE)

# show HTML-table, indicating p-values as stars
# and integrate CI in estimate column
sjt.lm(fit1, fit2, show.std = TRUE, ci.hyphen = " to ",
       minus.sign = "&minus;", p.numeric = FALSE,
       separate.ci.col = FALSE)

# ----------------------------------
# connecting two html-tables
# ----------------------------------
# fit two more models
fit3 <- lm(tot_sc_e ~ c160age + c12hour + c161sex + c172code, data=efc)
fit4 <- lm(e42dep ~ c160age + c12hour + c161sex + c172code, data=efc)

# create and save first HTML-table
part1 <- sjt.lm(fit1, fit2)

# create and save second HTML-table
part2 <- sjt.lm(fit3, fit4)

# browse temporary file
htmlFile <- tempfile(fileext=".html")
write(sprintf("<html><head>%s</head><body>%s<p></p>%s</body></html>",
              part1$page.style, part1$page.content, part2$page.content),
      file = htmlFile)
viewer <- getOption("viewer")
if (!is.null(viewer)) viewer(htmlFile) else utils::browseURL(htmlFile)

# ----------------------------------
# User defined style sheet
# ----------------------------------
sjt.lm(fit1, fit2,
       CSS = list(css.table = "border: 2px solid;",
                  css.tdata = "border: 1px solid;",
                  css.depvarhead = "color:#003399;"))

# ----------------------------------
# automatic grouping of predictors
# ----------------------------------
library(sjmisc)
data(efc)
```

```
# make education categorical
efc$c172code <- to_factor(efc$c172code)

# fit first model again (with c172code as factor)
fit1 <- lm(barthtot ~ c160age + c12hour + c172code + c161sex, data=efc)
# fit second model again (with c172code as factor)
fit2 <- lm(neg_c_7 ~ c160age + c12hour + c172code + c161sex, data=efc)

# plot models, but group by predictors
sjt.lm(fit1, fit2, group.pred = TRUE)

# ----------------------------------------
# compare models with different predictors
# ----------------------------------------
library(sjmisc)
data(efc)

# make education categorical
efc$c172code <- to_factor(efc$c172code)
# make education categorical
efc$e42dep <- to_factor(efc$e42dep)

# fit first model
fit1 <- lm(neg_c_7 ~ c160age + c172code + c161sex, data = efc)
# fit second model
fit2 <- lm(neg_c_7 ~ c160age + c172code + c161sex + c12hour, data = efc)
# fit second model
fit3 <- lm(neg_c_7 ~ c160age + c172code + e42dep + tot_sc_e, data = efc)

sjt.lm(fit1, fit2, fit3)

# ----------------------------------------
# compare models with different predictors
# and grouping
# ----------------------------------------
# make cope-index categorical
efc$c82cop1 <- to_factor(efc$c82cop1)
# fit another model
fit4 <- lm(neg_c_7 ~ c160age + c172code + e42dep + tot_sc_e + c82cop1,
           data = efc)

sjt.lm(fit1, fit2, fit4, fit3)

# show standardized beta only
sjt.lm(fit1, fit2, fit4, fit3, show.est = FALSE, show.std = TRUE,
       show.aic = TRUE, show.fstat = TRUE)

# ------------------------------------------------------------
# color insanity. just to show that each column has an own
# CSS-tag, so - depending on the stats and values you show -
# you can define column spaces / margins, border etc. to
# visually separate your models in the table
# ------------------------------------------------------------
```

```
sjt.lm(fit1, fit2, fit4, fit3, show.std = TRUE, show.aic = TRUE,
       show.fstat = TRUE, show.se = TRUE,
       CSS = list(css.modelcolumn1 = 'color:blue;',
                  css.modelcolumn2 = 'color:red;',
                  css.modelcolumn3 = 'color:green;',
                  css.modelcolumn4 = 'color:#ffff00;',
                  css.modelcolumn5 = 'color:#777777;',
                  css.modelcolumn6 = 'color:#3399cc;',
                  css.modelcolumn7 = 'color:#cc9933;'))

sjt.lm(fit1, fit2, fit4, fit3, show.est = FALSE, show.std = TRUE,
       p.numeric = FALSE, group.pred = FALSE,
       CSS = list(css.modelcolumn4 = 'border-left:1px solid black;',
                  css.modelcolumn5 = 'padding-right:50px;'))
## End(Not run)
```

---

sjt.lmer *Summary of linear mixed effects models as HTML table*

Description

Summarizes (multiple) fitted linear mixed effects models (estimates, std. beta values etc.) as HTML table, or saves them as file. The fitted models may have different predictors, e.g. when comparing different stepwise fitted models.

Usage

```
sjt.lmer(..., pred.labels = NULL, depvar.labels = NULL,
  remove.estimates = NULL, group.pred = FALSE, p.numeric = TRUE,
  emph.p = FALSE, p.zero = FALSE, p.kr = TRUE, separate.ci.col = TRUE,
  newline.ci = TRUE, show.est = TRUE, show.std = NULL, show.ci = TRUE,
  show.se = FALSE, show.header = FALSE, show.col.header = TRUE,
  show.r2 = TRUE, show.icc = TRUE, show.re.var = TRUE,
  show.fstat = FALSE, show.aic = FALSE, show.aicc = FALSE,
  show.dev = FALSE, string.pred = "Predictors",
  string.dv = "Dependent Variables", string.interc = "(Intercept)",
  string.obs = "Observations", string.est = "B", string.std = "std. Beta",
  string.ci = "CI", string.se = "std. Error", string.p = "p",
  ci.hyphen = " &ndash; ", minus.sign = "&#45;", digits.est = 2,
  digits.std = 2, digits.p = 3, digits.ci = 2, digits.se = 2,
  digits.summary = 3, cell.spacing = 0.2, cell.gpr.indent = 0.6,
  sep.column = TRUE, CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, remove.spaces = TRUE)
```

Arguments

...	One or more fitted linear (mixed) models.

`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. If `NULL`, variable labels are set based on label attributes (see `get_label`). If `pred.labels = ""`, column names (vector names) are used as predictor labels. See 'Examples'.

`depvar.labels` Character vector with labels of dependent variables of all fitted models. See 'Examples'.

`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. *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.

`group.pred` Logical, if `TRUE` (default), automatically groups table rows with factor levels of same factor, i.e. predictors of type `factor` will be grouped, if the factor has more than two levels. Grouping means that a separate headline row is inserted to the table just before the predictor values.

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

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

`p.zero` logical, if `TRUE`, p-values have a leading 0 before the period (e.g. *0.002*), else p-values start with a period and without a zero (e.g. *.002*).

`p.kr` logical, if `TRUE`, p-value estimation is based on conditional F-tests with Kenward-Roger approximation for the df. Caution: Computation may take very long time for large samples!

`separate.ci.col`
Logical, if `TRUE`, the CI values are shown in a separate table column. Default is `FALSE`.

`newline.ci` Logical, if `TRUE` and `separate.ci.col = FALSE`, inserts a line break between estimate and CI values. If `FALSE`, CI values are printed in the same line as estimate values.

`show.est` Logical, if `TRUE` (default), the estimates are printed.

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

`show.ci` Logical, if `TRUE` (default), the confidence intervall is also printed to the table. Use `FALSE` to omit the CI in the table.

`show.se` Logical, if `TRUE`, the standard errors are also printed. Default is `FALSE`.

`show.header` Logical, if `TRUE`, the header strings `string.pred` and `string.dv` are shown. By default, they're hidden.

`show.col.header`
Logical, if `TRUE` (default), the table data columns have a headline with abbreviations for estimates, std. beta-values, confidence interval and p-values.

show.r2	Logical, if TRUE (default), the R2 and adjusted R2 values for each model are printed in the model summary. For linear mixed models, the R2 and Omega-squared values are printed (see r2 for details).
show.icc	Logical, if TRUE, the intra-class-correlation for each model is printed in the model summary. Only applies to mixed models.
show.re.var	Logical, if TRUE, the variance parameters for the random effects for each model are printed in the model summary. Only applies to mixed models. For details output, see 'Note' in icc.
show.fstat	Logical, if TRUE, the F-statistics for each model is printed in the model summary. Default is FALSE. This argument does not apply to sjt.lmer.
show.aic	Logical, if TRUE, the AIC value for each model is printed in the model summary. Default is FALSE.
show.aicc	Logical, if TRUE, the second-order AIC value for each model is printed in the model summary. Default is FALSE.
show.dev	Logical, if TRUE, the deviance for each model is printed in the model summary.
string.pred	Character vector,used as headline for the predictor column. Default is "Predictors".
string.dv	Character vector, used as headline for the dependent variable columns. Default is "Dependent Variables".
string.interc	Character vector, used as headline for the Intercept row. Default is "Intercept".
string.obs	character vector, used in the summary row for the count of observation (cases). Default is "Observations".
string.est	Character vector, used for the column heading of estimates.
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.p	Character vector, used for the column heading of p values. Default is "p".
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'.
digits.est	Amount of decimals for estimates
digits.std	Amount of decimals for standardized beta
digits.p	Amount of decimals for p-values
digits.ci	Amount of decimals for confidence intervals
digits.se	Amount of decimals for standard error
digits.summary	Amount of decimals for values in model summary

`cell.spacing` Numeric, inner padding of table cells. By default, this value is 0.2 (unit is cm), which is suitable for viewing the table. Decrease this value (0.05 to 0.1) if you want to import the table into Office documents. This is a convenient argument for the `CSS` argument for changing cell spacing, which would be: `CSS = list(css.thead = "padding:0.2cm;", css.tdata = "padding:0.2cm;")`.

`cell.gpr.indent`
Indent for table rows with grouped factor predictors. Only applies if `group.pred = TRUE`.

`sep.column` Logical, if `TRUE`, an empty table column is added after each model column, to add margins between model columns. By default, this column will be added to the output; however, when copying tables to office applications, it might be helpful not to add this separator column when modifying the table layout.

`CSS` A `list` with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in `sjt.frq`.

`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 openend 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.

`no.output` Logical, if `TRUE`, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in `knitr` documents. The html output can be accessed via the return value.

`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.

Details

Concerning the `show.std` argument, `show.std = "std"` will print normal standardized estimates. For `show.std = "std2"`, however, 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. This type of standardization uses the `standardize`-function from the **arm**-package. 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 standardized estimats should not be printed.

Computation of p-values (if necessary and if `p.kr = TRUE`) are based on conditional F-tests with Kenward-Roger approximation for the df, using the **pbkrtest**-package. If **pbkrtest** is not available or `p.kr = FALSE`, computation of p-values is based on normal-distribution assumption, treating the t-statistics as Wald z-statistics. See 'Details' in `p_value`.

The confidence intervals stem from **broom**'s tidy-function. For linear mixed models, the computation method is "Wald" (`lme4::confint.merMod(fit, method = "Wald")`).

Furthermore, see 'Details' in `sjt.frq`.

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 variance components of the random parts (see `show.re.var`) are denoted like:

- within-group variance: sigma-squared
- between-group-variance: tau-zero-zero
- random-slope-intercept-correlation: rho-zero-one

See Also

sjPlot manual: sjt.lmer and `sjt.lm` for further examples.

Examples

```
## Not run:
library(lme4)
library(sjmisc)
data(efc)

# prepare group variable
efc$grp = as.factor(efc$e15relat)
levels(x = efc$grp) <- get_labels(efc$e15relat)
efc$care.level <- sjmisc::rec(efc$n4pstu,
                              rec = "0=0;1=1;2=2;3:4=3",
                              as.num = FALSE,
                              append = FALSE)
levels(x = efc$care.level) <- c("none", "I", "II", "III")

# data frame for fitted model
mydf <- data.frame(neg_c_7 = efc$neg_c_7,
                   sex = efc$c161sex,
                   c12hour = efc$c12hour,
                   barthel = efc$barthtot,
                   education = to_factor(efc$c172code),
```

```
                  grp = efc$grp,
                  carelevel = efc$care.level)

# fit three sample models
fit1 <- lmer(neg_c_7 ~ sex + c12hour + barthel + (1|grp), data = mydf)
fit2 <- lmer(neg_c_7 ~ sex + c12hour + education + barthel + (1|grp), data = mydf)
fit3 <- lmer(neg_c_7 ~ sex + c12hour + education + barthel +
             (1|grp) + (1|carelevel), data = mydf)

# print summary table... automatic grouping does not work here,
# barthel-index is printed as category of education (values are
# correct, however, indentation is wrong)
sjt.lmer(fit1, fit2, ci.hyphen = " to ", group.pred = TRUE)

# either change order of models
sjt.lmer(fit2, fit1, group.pred = TRUE)
# or turn off automatic grouping of categorical predictors
sjt.lmer(fit1, fit2, group.pred = FALSE)

# print table, using vector names as labels
sjt.lmer(fit1, fit2, fit3, pred.labels = "")

# show other statistics
sjt.lmer(fit1, fit2, show.aic = TRUE, show.ci = FALSE,
         show.se = TRUE, p.numeric = FALSE)

sjt.lmer(fit1, fit2, fit3, show.aic = TRUE,
         separate.ci.col = FALSE, newline.ci = FALSE)

# user defined predictor labels
sjt.lmer(fit1, fit2, fit3, pred.labels = c("Elder's gender (female)",
         "Hours of care per week", "Barthel Index", "Educational level (mid)",
         "Educational level (high)"))
## End(Not run)
```

`sjt.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

```
sjt.pca(data, rotation = c("varimax", "oblimin"), nmbr.fctr = NULL,
  fctr.load.tlrn = 0.1, title = "Principal Component Analysis",
  var.labels = NULL, wrap.labels = 40, show.cronb = TRUE,
  show.msa = FALSE, show.var = FALSE, altr.row.col = FALSE, digits = 2,
  string.pov = "Proportion of Variance",
  string.cpov = "Cumulative Proportion", CSS = NULL, encoding = NULL,
  file = NULL, use.viewer = TRUE, no.output = FALSE,
  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 "varimax" for orthogonal rotation or "oblimin" for oblique transformation.
nmbr.fctr	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.
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	Table caption, as character vector.
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 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.
show.msa	Logical, if TRUE, shows an additional column with the measure of sampling adequacy according dor each component.
show.var	Logical, if TRUE, the proportions of variances for each component as well as cumulative variance are shown in the table footer.
altr.row.col	Logical, if TRUE, alternating rows are highlighted with a light gray background color.
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
string.pov	String for the table row that contains the proportions of variances. By default, *"Proportion of Variance"* will be used.
string.cpov	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. For more details, see this package-vignette, or 'Details' in sjt.frq.
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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

Details

See 'Details' in sjt.frq.

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.

Note

See 'Notes' in sjt.frq.

See Also

- sjPlot manual: sjt.pca
- sjp.pca

Examples

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

# recveive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# recveive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")
# auto-detection of labels
sjt.pca(efc[, start:end])
## End(Not run)
```

---

sjt.stackfrq *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

```
sjt.stackfrq(items, weight.by = NULL, title = NULL, var.labels = NULL,
  value.labels = NULL, wrap.labels = 20, sort.frq = NULL,
  altr.row.col = 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,
  no.output = FALSE, remove.spaces = TRUE)
```

Arguments

items	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	Table caption, as character vector.
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.
altr.row.col	Logical, if TRUE, alternating rows are highlighted with a light gray background color.
digits	Numeric, amount of digits after decimal point when rounding estimates and 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 variable's kurtosis is added to the summary. The kurtosis is retrieved from the describe-function of the **psych**-package and indicated by a lower case Greek omega.
digits.stats	amount of digits for rounding the skewness and kurtosis valuess. 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 openend either in the IDE's viewer pane or the default web browser.
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).
CSS	A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.
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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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.

Details

See 'Details' in sjt.frq.

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

See 'Notes' in sjt.frq.

See Also

- sjPlot manual: sjt-basics
- sjp.stackfrq
- sjp.likert

Examples

```
# ------------------------------
# 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:
sjt.stackfrq(likert_4, value.labels = levels_4, var.labels = items)

# ------------------------------
```

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

sjt.stackfrq(efc[, c(start:end)], altr.row.col = TRUE)

sjt.stackfrq(efc[, c(start:end)], altr.row.col = TRUE,
             show.n = TRUE, show.na = TRUE)

# --------------------------------
# User defined style sheet
# --------------------------------
sjt.stackfrq(efc[, c(start:end)], altr.row.col = 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)
```

`sjt.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

```
sjt.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, statistics = c("auto", "cramer",
  "phi", "spearman", "kendall", "pearson"), 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 = " &#37;",
  hundret = "100.0", CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, remove.spaces = TRUE, ...)
```

Arguments

`var.row`	Variable that should be displayed in the table rows.
`var.col`	Cariable 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	Table caption, as character vector.
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.
statistics	Name of measure of association that should be computed. May be one of "auto", "cramer", "phi", "spearman", "kendall" or "pearson". See 'Details'.
string.total	Character label for the total column / row header
digits	Numeric, amount of digits after decimal point when rounding estimates and values.
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 " %", hence the percentage sign has a non-breaking-space after the percentage value.
hundret	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. For more details, see this package-vignette, or 'Details' in sjt.frq.

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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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().

Details

The p-value for Cramer's V and the Phi coefficient are based on chisq.test(). If any expected value of a table cell is smaller than 5, or smaller than 10 and the df is 1, then fisher.test() is used to compute the p-value. The test statistic is calculated with cramer() resp. phi().

Both test statistic and p-value for Spearman's rho, Kendall's tau and Pearson's r are calculated with cor.test().

When statistics = "auto", only Cramer's V or Phi are calculated, based on the dimension of the table (i.e. if the table has more than two rows or columns, Cramer's V is calculated, else Phi).

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.

See Also

- sjPlot manual: sjt.xtab
- sjp.xtab

Examples

```
# prepare sample data set
data(efc)

# print simple cross table with labels
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep)

# print cross table with manually set
# labels and expected values
sjt.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
sjt.xtab(efc$e16sex, efc$e42dep, show.cell.prc = FALSE, emph.total = TRUE)

# User defined style sheet
sjt.xtab(efc$e16sex, efc$e42dep,
         CSS = list(css.table = "border: 2px solid;",
                    css.tdata = "border: 1px solid;",
                    css.horline = "border-bottom: double blue;"))
## End(Not run)

# ordinal data, use Kendall's tau
sjt.xtab(efc$e42dep, efc$quol_5, statistics = "kendall")

# calculate Spearman's rho, with continuity correction
sjt.xtab(
  efc$e42dep,
  efc$quol_5,
  statistics = "spearman",
  exact = FALSE,
  continuity = TRUE
)
```

---

tab_df *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

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

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.
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. For more details, see this package-vignette, or 'Details' in sjt.frq.
file	Destination file, if the output should be saved as file. If NULL (default), the output will be saved as temporary file and openend 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.

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

```
## 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)
```

---

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

```
view_df(x, weight.by = NULL, altr.row.col = 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,
  hide.progress = FALSE, CSS = NULL, encoding = NULL, file = NULL,
  use.viewer = TRUE, no.output = FALSE, 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	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.
altr.row.col	Logical, if TRUE, alternating rows are highlighted with a light gray background color.
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, 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 strem from weight.by.
show.wtd.prc	Logical, if TRUE, an additional column with weighted percentage of frequencies for each variable is shown. Weights strem 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.
hide.progress	logical, if TRUE, the progress bar that is displayed when creating the output is hidden. Default in FALSE, hence the bar is visible.
CSS	A list with user-defined style-sheet-definitions, according to the official CSS syntax. For more details, see this package-vignette, or 'Details' in sjt.frq.
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 openend 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.
no.output	Logical, if TRUE, the html-output is neither opened in a browser nor shown in the viewer pane and not even saved to file. This option is useful when the html output should be used in knitr documents. The html output can be accessed via the return value.
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