Package 'sjPlot'

September 10, 2014

Type Package

Encoding UTF-8

Title sjPlot - data visualization for statistics in social science

Version 1.5

Date 2014-09-10

Author Daniel Lüdecke <d.luedecke@uke.de>

Maintainer Daniel Luedecke <d.luedecke@uke.de>

Description Collection of several plotting / table output functions for visualizing data and utility functions.

License GPL-3

Imports ggplot2, MASS, car, foreign, plyr, psych, reshape2, scales

Suggests cluster, coin, lsmeans, lmtest, stringdist

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

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

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-10 11:38:24

R topics documented:

`sjPlot-package` *sjPlot - 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 (forest plots), PCA, correlations, cluster analyses, scatter plots etc.

Furthermore, this package contains some tools that are useful when carrying out data analysis or interpreting data (especially intended for people coming from SPSS and who are new to R). These tool functions help with (SPSS) data set import, variable recoding and weighting, statistical tests, determination of cluster groups, interpretation of interaction terms in linear models etc.

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-diagram or print them as html-table

However, meanwhile the amount of functions has increased, hence you'll also find some utility functions beside the plotting functions (see below).

How does this package help me?

Basically, this package either helps those users who...

- have difficulties using and/or understanding all possibilities that ggplot offers to create plots, simply by providing intuitive function parameters, which allow for manipulating the appearance of plots;
- don't want to set up complex ggplot-object each time from the scratch;

- like quick inspections of (basic) statistics via (html-)tables that are shown in the IDE's viewer pane or default browser; or
- want easily create beautiful table outputs that can be imported in office applications.

For advanced users, each functions returns either the prepared ggplot-object (in case of `sjp`-plotting functions) or the HTML-tables (in case of `sjt`-table-output functions), which than can be manipulated even further (for instance, for ggplot-objects, you can specify certain parameters that cannot be modified via the sjPlot package or html-tables could be integrated into knitr-documents).

The functions follow specific naming conventions:

- `sjc` - cluster analysis functions
- `sji` - data import functions
- `sjp` - plotting functions
- `sjt` - (HTML) table output functions
- `sjs` - statistical tests functions
- `sju` - utility and recode functions

Details

Package:	sjPlot
Type:	Package
Version:	1.5
Date:	2014-09-10
License:	GPL-3

Author(s)

Daniel Lüdecke <d.luedecke@uke.de>

References

RPubs: http://rpubs.com/sjPlot
Weblog: http://strengejacke.wordpress.com/sjplot-r-package/
Github: https://github.com/sjPlot/devel
Bug reports: https://github.com/sjPlot/devel/issues

---

efc *Sample dataset from the EUROFAMCARE project*

Description

Sample dataset from the EUROFAMCARE project

References

http://www.uke.de/eurofamcare/

See Also

sji.SPSS
sji.viewSPSS
sjt.df
sji.getValueLabels
sji.getVariableLabels
sji.convertToLabel
sjp.frq

Examples

```
# Attach EFC-data
data(efc)

# Show structure
str(efc)

# show first rows
head(efc)

# show variables
## Not run:
sji.viewSPSS(efc)
## End(Not run)

# show variable labels
sji.getVariableLabels(efc)

# plot efc-data frame summary
## Not run:
sjt.df(efc, alternateRowColor=TRUE)
## End(Not run)
```

---

sjc.cluster *Compute hierarchical or kmeans cluster analysis*

Description

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

Usage

```
sjc.cluster(data, groupcount, method = "h", distance = "euclidean",
  agglomeration = "ward", iter.max = 20, algorithm = "Hartigan-Wong")
```

Arguments

data	The data frame containing all variables that should be used for the cluster analysis.
groupcount	The amount of groups (clusters) that should be retrieved. May also be a set of initial (distinct) cluster centres, in case method is "kmeans" (see kmeans for details on centers parameter). If groupcount indicates a number of clusters, following functions may be helpful for estimating the amount of clusters: • Use sjc.elbow-function to determine the group-count depending on the elbow-criterion. • If using kmeans as method, use sjc.kgap-function to determine the group-count according to the gap-statistic. • If using hierarchical as method (default), use sjc.dend-function to inspect different cluster group solutions. • Use sjc.grpdisc-function to inspect the goodness of grouping (accuracy of classification).
method	Indicates the clustering method. If "hclust" (default), a hierachical clustering using the ward method is computed. Use "kmeans" to compute a k-means clustering. You can specifiy inital letters only.
distance	The distance measure to be used when "method" is "hclust" (for hierarchical clustering). This must be one of "euclidean" (default), "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist.
agglomeration	The agglomeration method to be used when "method" is "hclust" (for hierarchical clustering). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). Note that since R version > 3.0.3, the "ward" option has been replaced by either "ward.D" or "ward.D2". In such case, you may also use these values.
iter.max	the maximum number of iterations allowed. Only applies, if method is "kmeans". See kmeans for details on this parameter.
algorithm	algorithm used for calculating kmeans cluster. Only applies, if method is "kmeans". May be one of "Hartigan-Wong" (default), "Lloyd" (used by SPSS), or "MacQueen". See kmeans for details on this parameter.

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

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 specifying nrow and ncol parameters.
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.

See Also

sjc.qclus
sjc.dend
sjc.grpdisc
sjc.elbow
kmeans
hclust
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 highlighting 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	The data frame containing all 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.

- If using kmeans as method, use sjc.kgap-function to determine the group-count according to the gap-statistic.
- 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 — The distance measure to be used. This must be one of "euclidean" (default), "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist.

agglomeration — The agglomeration method to be used. This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). Note that since R version > 3.0.3, the "ward" option has been replaced by either "ward.D" or "ward.D2". In such case, you may also use these values.

See Also

sjc.cluster
sjc.grpdisc
sjc.elbow

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, showDiff = FALSE)
```

Arguments

data	The data frame containing all variables that should be used for determining the elbow criteria.
steps	The maximum group-count for the k-means cluster analysis for which the elbow-criterion should be displayed. Default is 15.
showDiff	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 parameter is FALSE.

See Also

sjc.kgap
sjc.dend
sjc.cluster
sjc.grpdisc

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, showTotalCorrect = TRUE,
  printPlot = TRUE)
```

Arguments

data	The data frame containing all variables that should be used for the check for goodness of classification of a cluster analysis.
groups	The group classification of the cluster analysis that was returned from the sjc.cluster-function.
groupcount	The amount of groups (clusters) that should be used. Use the sjc.elbow-function to determine the group-count depending on the elbow-criterion.
showTotalCorrect	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.
printPlot	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.

See Also

`sjc.dend`
`sjc.cluster`
`sjc.elbow`

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 (http://cran.r-project.org/web/packages/cluster/index.html) to calculate the data for the plot.

Usage

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

Arguments

`x`	A matrix, where rows are observations and columns are individual dimensions, to compute and plot the gap statistic (according to a uniform reference distribution).
`max`	The 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 — A 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 — 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. (http://cran.r-project.org/web/packages/cluster/index.html)

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 graph.

1. If `method` is `kmeans`, this function first determines the optimal group count via gap statistics (unless parameter `groupcount` is specified), using the `sjc.kgap` function.
2. Than a cluster analysis is performed by running the `sjc.cluster` function to determine the cluster groups.
3. After that, 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 shown in a graph.

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

Usage

```
sjc.qclus(data, groupcount = NULL, groups = NULL, method = "k",
  distance = "euclidean", agglomeration = "ward", iter.max = 20,
  algorithm = "Hartigan-Wong", showAccuracy = FALSE, title = NULL,
  titleSize = 1.3, titleColor = "black", axisLabels.x = NULL,
  axisLabelAngle.x = 0, axisLabelSize = 1.1, axisLabelColor = "gray30",
  axisTitle.x = "Cluster group characteristics",
  axisTitle.y = "Mean of z-scores", axisTitleColor = "black",
  axisTitleSize = 1.3, breakTitleAt = 40, breakLabelsAt = 12,
  breakLegendTitleAt = 20, breakLegendLabelsAt = 20, facetCluster = FALSE,
  barColor = NULL, barAlpha = 1, colorPalette = "GnBu", barWidth = 0.5,
  barSpace = 0.1, barOutline = FALSE, barOutlineSize = 0.2,
  barOutlineColor = "black", theme = NULL, borderColor = NULL,
  axisColor = NULL, hideLegend = FALSE, showTickMarks = TRUE,
  showAxisLabels.x = TRUE, showAxisLabels.y = TRUE, showGroupCount = TRUE,
  showAccuracyLabels = FALSE, legendTitle = NULL, legendLabels = NULL,
  legendPos = "right", legendSize = 1, legendBorderColor = "white",
  legendBackColor = "white", majorGridColor = NULL, minorGridColor = NULL,
  hideGrid.x = FALSE, hideGrid.y = FALSE, flipCoordinates = FALSE,
  reverseAxis.x = FALSE, printPlot = TRUE)
```

Arguments

`data`	The data frame containing all variables that should be used for the cluster analysis.

groupcount — The amount of groups (clusters) that should be retrieved. May also be a set of initial (distinct) cluster centres, in case method is "kmeans" (see kmeans for details on centers parameter). By default (NULL), the optimal amount of clusters is calculated using the gap statistics (see sjc.kgap. However, this works only with kmeans as method. If method is "hclust", you have to specify a group-count. Use the sjc.elbow-function to determine the group-count depending on the elbow-criterion. Use sjc.grpdisc-function to inspect the goodness of grouping.

groups — By default, this parameter is NULL and will be ignored. However, if you just want to plot an already existing cluster solution without computing a new cluster analysis, specifiy groupcount and group. group 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.

method — The 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 — The distance measure to be used when "method" is "hclust" (for hierarchical clustering). This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". See dist. By default, method is "kmeans" and this parameter will be ignored.

agglomeration — The agglomeration method to be used when "method" is "hclust" (for hierarchical clustering). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". Default is "ward" (see hclust). Note that since R version > 3.0.3, the "ward" option has been replaced by either "ward.D" or "ward.D2". In such case, you may also use these values. By default, method is "kmeans" and this parameter will be ignored.

iter.max — the maximum number of iterations allowed. Only applies, if method is "kmeans". See kmeans for details on this parameter.

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

showAccuracy — 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 — Title of diagram as string. Example: title=c("my title")

titleSize — The size of the plot title. Default is 1.3.

titleColor — The color of the plot title. Default is "black".

axisLabels.x — Labels for the x-axis breaks. Example: axisLabels.x=c("Label1", "Label2", "Label3"). Note: If you use the sji.SPSS function and the sji.getValueLabels function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.

axisLabelAngle.x — Angle for axis-labels.

axisLabelSize — The size of axis labels of both x and y axis. Default is 1.1, recommended values range between 0.5 and 3.0.

`axisLabelColor` User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

`axisTitle.x` A label for the x axis. useful when plotting histograms with metric scales where no category labels are assigned to the x axis.

`axisTitle.y` A label for the y axis. useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

`axisTitleColor` The color of the x and y axis labels. Refers to `axisTitle.x` and `axisTitle.y`, not to the tick mark or category labels.

`axisTitleSize` the size of the x and y axis labels. Refers to `axisTitle.x` and `axisTitle.y`, not to the tick mark or category labels. Default is 1.3.

`breakTitleAt` Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

`breakLabelsAt` Determines how many chars of the labels are displayed in one line and when a line break is inserted into the axis labels.

`breakLegendTitleAt`
Determines how many chars of the legend title are displayed in one line and when a line break is inserted into the legend title.

`breakLegendLabelsAt`
Determines how many chars of the legend labels are displayed in one line and when a line break is inserted into the axis labels.

`facetCluster` If `TRUE`, each cluster group will be represented by an own panel. Default is `FALSE`, thus all cluster groups are plotted in a single graph.

`barColor` User defined color for bars.

- If not specified (`NULL`), a default color palette will be used for the bar charts.
- If barColor is `"gs"`, a greyscale will be used.
- If barColor is `"bw"`, a monochrome white filling will be used.
- If barColor is `"brewer"`, use the `colorPalette` parameter to specify a palette of the http://colorbrewer2.org.

Else specify your own color values as vector (e.g. `barColor=c("#f00000", "#00ff00", "#0080ff")`).

`barAlpha` Specify the transparancy (alpha value) of bars.

`colorPalette` If `barColor` is `"brewer"`, specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here.

`barWidth` Width of bars. Recommended values for this parameter are from 0.4 to 1.5

`barSpace` Spacing between bars. Default value is 0.1. If 0 is used, the grouped bars are sticked together and have no space in between. Recommended values for this parameter are from 0 to 0.5

`barOutline` If `TRUE`, each bar gets a colored outline. Default is `FALSE`.

`barOutlineColor`
The color of the bar outline. Only applies, if `barOutline` is set to `TRUE`.

`barOutlineSize` The size of the bar outlines. Only applies if `barOutline` is `TRUE`. Default is 0.2

`theme` Specifies the diagram's background theme. Default (parameter `NULL`) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

borderColor — User defined color of whole diagram border (panel border).

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

hideLegend — Indicates whether legend (guide) should be shown or not.

showTickMarks — Whether tick marks of axes should be shown or not.

showAxisLabels.x — Whether x axis labels (cluster variables) should be shown or not.

showAxisLabels.y — Whether y axis labels (z scores) should be shown or not.

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

showAccuracyLabels — if TRUE, the accuracy-values for each cluster group is added to the legend labels (e.g. "Group 1 (n=87, accuracy=95.3)"). Accuracy is calculated by sjc.grpdisc.

legendTitle — Title of the diagram's legend.

legendLabels — Labels for the guide/legend. Example: See axisLabels.x. If legendLabels is NULL (default), the standard string "Group <nr>" will be used.

legendPos — The position of the legend, if a legend is drawn. 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.

legendSize — The text size of the legend. Default is 1. Relative size, so recommended values are from 0.3 to 2.5

legendBorderColor — Color of the legend's border. Default is "white", so no visible border is drawn.

legendBackColor — Fill color of the legend's background. Default is "white", so no visible background is drawn.

majorGridColor — Specifies the color of the major grid lines of the diagram background.

minorGridColor — Specifies the color of the minor grid lines of the diagram background.

hideGrid.x — If TRUE, the x-axis-gridlines are hidden. Default is FALSE.

hideGrid.y — If TRUE, the y-axis-gridlines are hidden. Default is FALSE.

flipCoordinates — If TRUE, the x and y axis are swapped.

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

printPlot — 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,
- `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

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 specifying `nrow` and `ncol` parameters.
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.

See Also

`sjc.cluster`
`sjc.kgap`
`sjc.elbow`
`sjc.grpdisc`
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, facetCluster=TRUE)
## End(Not run)
```

`sji.convertToLabel` *Replaces variable values with their associated value labels*

Description

This function converts (replaces) variable values (of factors) with their associated value labels. Might be helpful for factor variables. For instance, if you have a Gender variable with 0/1, and associated labels are male/female, this function would convert all 0 to male and all 1 to female in the data frame.

Usage

```
sji.convertToLabel(variable)
```

Arguments

`variable`	A (factor) variable.

Value

A factor variable containing with the replaced value labels.

See Also

sji.convertToValue
sji.getValueLabels
sji.getVariableLabels
sji.SPSS

Examples

```
data(efc)
print(sji.getValueLabels(efc)['c161sex'])
head(efc$c161sex)
head(sji.convertToLabel(efc$c161sex))

print(sji.getValueLabels(efc)['e42dep'])
table(efc$e42dep)
table(sji.convertToLabel(efc$e42dep))
```

sji.convertToValue *Converts factors to numeric variables*

Description

This function converts (replaces) factor values with the related factor level index number, thus the factor is converted to a numeric variable.

Usage

```
sji.convertToValue(fac, startAt = 1)
```

Arguments

fac	A (factor) variable.
startAt	the starting index, i.e. numeric value of the variable.

Value

A numeric variable with values ranging from `startAt` to `startAt` + length of factor levels.

See Also

sji.convertToLabel
sji.getValueLabels
sji.getVariableLabels
sji.SPSS

Examples

```
data(efc)
test <- sji.convertToLabel(efc$e42dep)
table(test)

table(sji.convertToValue(test))
hist(sji.convertToValue(test,0))
```

sji.getValueLabels *Retrieve value labels of an SPSS-imported data frame*

Description

This function retrieves the value labels of an imported SPSS data set and returns the result as list.

Usage

```
sji.getValueLabels(dat)
```

Arguments

dat	a data frame containing imported SPSS data

Value

a list with all value labels from the SPSS dataset

References

http://rpubs.com/sjPlot/datainit

See Also

sji.SPSS
sji.getVariableLabels
sji.convertToLabel
sji.convertToValue
sji.setValueLabels

Examples

```
# import SPSS data set
# mydat <- sji.SPSS("my_spss_data.sav", enc="UTF-8")

# retrieve variable labels
# mydat.var <- sji.getVariableLabels(mydat)

# retrieve value labels
# mydat.val <- sji.getValueLabels(mydat)
```

---

sji.getVariableLabels *Retrieve variable labels of an SPSS-imported data frame*

Description

This function retrieves the variable labels of an imported SPSS data set and returns the result as list.

Usage

```
sji.getVariableLabels(dat)
```

Arguments

dat	A data frame containing imported SPSS data.

Value

A list with all variable labels from the SPSS dataset.

References

http://rpubs.com/sjPlot/datainit

See Also

sji.getValueLabels
sji.setValueLabels
sji.setVariableLabels
sji.SPSS
sji.convertToLabel
sji.convertToValue

Examples

```
# import SPSS data set
# mydat <- sji.SPSS("my_spss_data.sav", enc="UTF-8")

# retrieve variable labels
# mydat.var <- sji.getVariableLabels(mydat)

# retrieve value labels
# mydat.val <- sji.getValueLabels(mydat)
```

---

sji.setValueLabels *Attach value labels to a variable or vector*

Description

This function attaches character labels as "value.labels" attribute to a variable or vector "x", resp. to all variables of a data frame if "x" is a data.frame. These value labels will be accessed by most of this package's functions, in order to automatically set values or legend labels.

Usage

```
sji.setValueLabels(x, labels)
```

Arguments

x	a variable (vector) or a data frame where labels should be attached. Replaces former value labels.
labels	a character vector of labels that will be attached to "x" by setting the "value.labels" attribute. The length of this character vector must equal the value range of "x", i.e. if "x" has values from 1 to 3, "labels" should have a length of 3. If "x" is a data frame, labels must be a list of character vectors.

Value

"x" with attached value labels.

References

http://rpubs.com/sjPlot/datainit

See Also

sji.SPSS
sji.getVariableLabels
sji.convertToLabel
sji.convertToValue
sji.getValueLabels

Examples

```
dummy <- sample(1:4, 40, replace=TRUE)
sjp.frq(dummy)

dummy <- sji.setValueLabels(dummy, c("very low", "low", "mid", "hi"))
sjp.frq(dummy)
```

---

sji.setVariableLabels *Set variable label(s) to a single variable or data frame*

Description

This function sets variable labels to a single variable or to a set of variables in a data frame. To each variable, the attribute `"variable.label"` with the related variable name is attached. Most of this package's functions can automatically retrieve the variable name to use it as axis labels or plot title.

Usage

```
sji.setVariableLabels(x, lab)
```

Arguments

`x`	A single variable (vector) or data frame with variables.
`lab`	If `x` is a vector (single variable), use a single character string with the variable label for `x`. If `x` is a `data.frame`, use a vector with character labels of same length as `ncol(x)`.

Value

`x`, with attached `"variable.label"` attribute(s), which contains the variable name(s).

References

http://rpubs.com/sjPlot/datainit

See Also

sji.getValueLabels
sji.setValueLabels
sji.getVariableLabels
sji.SPSS
sji.convertToLabel
sji.convertToValue

Examples

```
# sample data set, imported from SPSS. Variable labels are attached
# as attribute to the data frame (so variables currently don't have this attribute)
data(efc)
# get variable labels
variable.labels <- sji.getVariableLabels(efc)
# set variable labels as attribute to each single variable of data frame
efc <- sji.setVariableLabels(efc, variable.labels)

## Not run:
sjt.frq(efc$e42dep)
sjt.frq(data.frame(efc$e42dep, efc$e16sex))
## End(Not run)

# ------------------------------------------
# manually set value and variable labels
# ------------------------------------------
dummy <- sample(1:4, 40, replace=TRUE)
dummy <- sji.setValueLabels(dummy, c("very low", "low", "mid", "hi"))
dummy <- sji.setVariableLabels(dummy, "Dummy-variable")
# auto-detection of value labels by default, auto-detection of
# variable labels if parameter "title" set to "auto"
sjp.frq(dummy, title="auto")
```

---

sji.SPSS *Import SPSS dataset as data frame into R*

Description

Import data from SPSS, including NA's, value and variable labels.

Usage

```
sji.SPSS(path, enc = NA, autoAttachVarLabels = FALSE)
```

Arguments

path	The file path to the SPSS dataset.
enc	The file encoding of the SPSS dataset.

autoAttachVarLabels
: if TRUE, variable labels will automatically be attached to each variable as "variable.label" attribute. See sji.setVariableLabels for details.

Value

A data frame containing the SPSS data. retrieve value labels with sji.getValueLabels and variable labels with sji.getVariableLabels.

Note

This is a wrapper function for read.spss of the foreign package, using convenient parameter default settings.

References

http://strengejacke.wordpress.com/sjplot-r-package/

http://strengejacke.wordpress.com/2013/02/24/simplify-your-r-workflow-with-functions-rstats/

See Also

sji.getValueLabels
sji.getVariableLabels
sji.convertToLabel
sji.convertToValue
sji.viewSPSS

Examples

```
# import SPSS data set
# mydat <- sji.SPSS("my_spss_data.sav", enc="UTF-8")

# retrieve variable labels
# mydat.var <- sji.getVariableLabels(mydat)

# retrieve value labels
# mydat.val <- sji.getValueLabels(mydat)
```

---

sji.viewSPSS *View SPSS data set structure*

Description

Save (or show) content of an imported SPSS data file as HTML table. Similar to the SPSS variable view. This quick overview shows variable ID numner, name, label, type and associated value labels. The result can be considered as "codeplan" of the data frame.

Usage

```
sji.viewSPSS(df, file = NULL, alternateRowColors = TRUE, showID = TRUE,
  showType = FALSE, showValues = TRUE, showValueLabels = TRUE,
  showFreq = FALSE, showPerc = FALSE, orderByName = FALSE,
  breakVariableNamesAt = 50, encoding = "UTF-8", hideProgressBar = FALSE,
  CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

df	An imported data frame, imported by sji.SPSS function.
file	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the IDE's viewer pane or in the default web browser.
alternateRowColors	If TRUE, alternating rows are highlighted with a light gray background color.
showID	If TRUE (default), the variable ID is shown in the first column.
showType	If TRUE, the variable type is shown in a separate column. Since SPSS variable types are mostly numeric after import, this column is hidden by default.
showValues	If TRUE (default), the variable values are shown as additional column.
showValueLabels	If TRUE (default), the value labels are shown as additional column.
showFreq	If TRUE, an additional column with frequencies for each variable is shown.
showPerc	If TRUE, an additional column with percentage of frequencies for each variable is shown.
orderByName	If TRUE, rows are ordered according to the variable names. By default, rows (variables) are ordered according to their order in the data frame.
breakVariableNamesAt	Wordwrap for lomg variable names. Determines how many chars of a variable name are displayed in one line and when a line break is inserted. Default value is 50, use NULL to turn off word wrap.
hideProgressBar	If TRUE, the progress bar that is displayed when creating the table is hidden. Default in FALSE, hence the bar is visible.
encoding	The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need: 1. the class-names with "css."-prefix as parameter 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 parameter 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.arc='color:blue;'` for a blue text color each 2nd row.
- `css.summary='+color:blue;'` to add blue font color style to the summary row.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

`useViewer` If `TRUE`, the function tries to show the HTML table in the IDE's viewer pane. If `FALSE` or no viewer available, the HTML table is opened in a web browser.

`no.output` 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.

Value

Invisibly returns a `structure` with

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

for further use.

See Also

`sji.SPSS`
`sjt.df`

Examples

```
# init dataset
data(efc)

# view variables
## Not run:
sji.viewSPSS(efc)
## End(Not run)

# view variables w/o values and value labels
## Not run:
sji.viewSPSS(efc, showValues=FALSE, showValueLabels=FALSE)
## End(Not run)

# view variables including variable typed, orderd by name
## Not run:
sji.viewSPSS(efc, orderByName=TRUE, showType=TRUE)
```

```
## End(Not run)

# ------------------------------------------------------------
# User defined style sheet
# ------------------------------------------------------------
## Not run:
sji.viewSPSS(efc,
             CSS=list(css.table="border: 2px solid;",
                      css.tdata="border: 1px solid;",
                      css.arc="color:blue;"))
## End(Not run)
```

---

sjp.aov1 *Plot One-Way-Anova tables*

Description

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

Usage

```
sjp.aov1(depVar, grpVar, meansums = FALSE, type = "dots",
  hideErrorBars = FALSE, title = NULL, titleSize = 1.3,
  titleColor = "black", axisLabels.y = NULL, reverseOrder = FALSE,
  stringIntercept = "(Intercept)", showAxisLabels.y = TRUE,
  axisLabelSize = 1.1, axisLabelColor = "gray30", axisTitle.x = NULL,
  axisTitleSize = 1.2, axisTitleColor = c("#444444"), axisLimits = NULL,
  valueLabelColor = "grey20", valueLabelColorNS = "grey50",
  valueLabelSize = 4.5, valueLabelAlpha = 0.8, axisLabelAngle.x = 0,
  axisLabelAngle.y = 0, errorBarColor = NULL, errorBarWidth = 0,
  errorBarSize = 0.8, errorBarLineType = 1, pointColor = c("#3366a0",
  "#aa3333"), pointSize = 3, barColor = c("#3366a0", "#aa3333"),
  barWidth = 0.5, barAlpha = 1, barOutline = FALSE,
  barOutlineColor = "black", breakTitleAt = 50, breakLabelsAt = 12,
  gridBreaksAt = NULL, borderColor = NULL, axisColor = NULL,
  theme = NULL, majorGridColor = NULL, minorGridColor = NULL,
  hideGrid.x = FALSE, hideGrid.y = FALSE, expand.grid = FALSE,
  showTickMarks = TRUE, showValueLabels = TRUE, labelDigits = 2,
  showPValueLabels = TRUE, showModelSummary = TRUE, printPlot = TRUE)
```

Arguments

depVar	The dependent variable. Will be used with following formular: `aov(depVar ~ grpVar)`
grpVar	The grouping variable, as unordered factor. Will be used with following formular: `aov(depVar ~ grpVar)`

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

type — Indicates Whether the group means should be plotted as "dots" (aka forest plots, default) or as "bars".

hideErrorBars — If TRUE, the error bars that indicate the confidence intervals of the group means are not shown. Only applies if parameter type is "bars". Default value is FALSE.

title — Diagram's title as string. Example: title=c("my title") Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

titleSize — The size of the plot title. Default is 1.3.

titleColor — The color of the plot title. Default is "black".

axisLabels.y — Value labels of the grouping variable grpVar that are used for labelling the grouping variable axis. Passed as vector of strings. Example: axisLabels.y=c("Label1", "Label2", '
Note: If you use the sji.SPSS function and the sji.getValueLabels function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically. See examples below.
Note: In case type is "bars", the grpVar will be plotted along the x-axis.

reverseOrder — If TRUE, the order of the factor categories (groups) is reversed. Default is FALSE.

stringIntercept — A string that indicates the reference group (intercept), that is appended to the value label of the grouping variable. Default is "(Intercept)".

showAxisLabels.y — Whether y axis text (category value) should be shown (use TRUE) or not. Default is TRUE.

axisLabelSize — The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8.

axisLabelColor — The color of the category labels (predictor labels). Default is a dark grey (grey30).

axisTitle.x — A label for the x axis. Default is NULL, which means no x-axis title. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

axisTitleColor — The color of the x axis label.

axisTitleSize — The size of the x axis label. Default is 1.2.

axisLimits — Defines the range of the axis where the beta coefficients and their confidence intervalls are drawn. By default, the limits range from the lowest confidence interval to the highest one, so the diagram has maximum zoom. Use your own values as 2-value-vector, for instance: limits=c(-0.8,0.8).

valueLabelColor — Colour of the values inside the diagram. Only applies, when parameter showValueLabels is set to TRUE. Use any valid colour value, e.g. valueLabelColor="grey50" or valueLabelColor=c("#cc3366"). Default is "grey20".

valueLabelColorNS
: Colour of the non significant values inside the diagram. Only applies, when parameter showValueLabels is set to TRUE. Use any valid colour value, e.g. valueLabelColor="grey50" or valueLabelColor=c("#cc3366"). Default is "grey50".

valueLabelSize
: Size of the value labels. Drfault is 4.5. Recommended Values range from 2 to 8

valueLabelAlpha
: The alpha level (transparancy) of the value labels. Default is 0.8, use any value from 0 to 1.

axisLabelAngle.x
: Angle for x-axis-labels, passed as numeric value.

axisLabelAngle.y
: Angle for y-axis-labels, passed as numeric value.

errorBarColor
: The color of the error bars that indicate the confidence intervalls of the group means. Default is NULL, which means that if type is "dots", the pointColor value will be used as error bar color. In case type is "bars", "black" will be used as error bar color.

errorBarWidth
: The width of the error bar ends. Default is 0.

errorBarSize
: The size of the error bar. Default is 0.8.

errorBarLineType
: The linetype of error bars. Default is 1 (solid line).

pointColor
: The colors of the points that indicate the mean-value. pointColor is a vector with two values: the first indicating groups with positive means and the second indicating negative means. Default is c("#3366a0", "#aa6633").

pointSize
: The size of the points that indicate the mean-value. Default is 3.

barColor
: The colors of the bars in bar charts. Only applies if parameter type is "bars". barColor is a vector with two values: the first indicating groups with positive means and the second indicating negative means. Default is c("#3366a0", "#aa6633").

barWidth
: The width of the bars in bar charts. Only applies if parameter type is "bars". Default is 0.5

barAlpha
: The alpha value of the bars in bar charts. Only applies if parameter type is "bars". Default is 1

barOutline
: If TRUE, each bar gets a colored outline. Only applies if parameter type is bars. Default is FALSE.

barOutlineColor
: The color of the bar outline. Only applies, if barOutline is set to TRUE. Default is black.

breakTitleAt
: Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title

breakLabelsAt
: Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted

gridBreaksAt
: Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is NULL, so pretty gridbeaks will be used.

borderColor
: User defined color of whole diagram border (panel border).

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

theme — Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

majorGridColor — Specifies the color of the major grid lines of the diagram background.

minorGridColor — Specifies the color of the minor grid lines of the diagram background.

hideGrid.x — If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y — If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

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

showTickMarks — Whether tick marks of axes should be shown or not

showValueLabels — Whether the value labels (mean differences) should be plotted to each dot or not.

labelDigits — The amount of digits for rounding the estimations (see showValueLabels). Default is 2, i.e. estimators have 2 digits after decimal point.

showPValueLabels — Whether the significance levels of each category/group should be appended to values or not.

showModelSummary — If TRUE (default), a summary of the anova model with Sum of Squares between groups (ssb), Sum of Squares within groups (ssw), multiple and adjusted R-square and F-Test is printed to the lower right corner of the diagram. Default is TRUE.

printPlot — 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

sju.aov1.levene

Examples

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


data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
sjp.aov1(efc$c12hour,
         as.factor(efc$e42dep),
         axisLabels.y=efc.val['e42dep'],
         axisTitle.x=efc.var[['c12hour']])

# ------------------------------------------------
# auto-detection of value labels and variable names
# ------------------------------------------------
efc <- sji.setVariableLabels(efc, efc.var)
sjp.aov1(efc$c12hour,
         efc$e42dep,
         title="auto",
         axisTitle.x="auto")

sjp.aov1(efc$c12hour,
         as.factor(efc$c172code),
         axisLabels.y=efc.val['c172code'],
         title=efc.var[['c12hour']],
         type="bars",
         showTickMarks=FALSE,
         showModelSummary=FALSE,
         axisLabelAngle.x=90)
```

---

sjp.chi2 *Plot Pearson's Chi2-Test of multiple contingency tables*

Description

Plot Pearson's Chi2-Test of multiple contingency tables as ellipses or tiles. Requires a data frame with dichotomous (dummy) variables. Calculation of Chi2-matrix taken from following blog-posting: http://talesofr.wordpress.com/2013/05/05/ridiculously-photogenic-factors-heatmap-with-p-valu

Usage

```
sjp.chi2(df, title = "Pearson's Chi2-Test of Independence", titleSize = 1.3,
  titleColor = "black", axisLabels = NULL, valueLabelColor = "black",
  valueLabelSize = 4.5, valueLabelAlpha = 1, outlineColor = "black",
  outlineSize = 0.5, axisColor = NULL, axisLabelSize = 1.1,
  axisLabelColor = "gray30", axisLabelAngle.x = 0, axisLabelAngle.y = 0,
```

```
  breakTitleAt = 50, breakLabelsAt = 12, hideLegend = TRUE,
  legendTitle = NULL, printPlot = TRUE)
```

Arguments

df	a data frame of (dichotomous) factor variables.
title	Title of the diagram, plotted above the whole diagram panel
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
axisLabels	Labels for the x- andy y-axis axisLabels are detected automatically if each variable has a "variable.label" attribute (see sji.setVariableLabels) for details).
valueLabelColor	the color of the value labels (numbers) inside the diagram
valueLabelSize	The size of value labels in the diagram. Default is 4.5, recommended values range between 2 and 8
valueLabelAlpha	specify the transparancy (alpha value) of value labels
outlineColor	defines the outline color of geoms (circles or tiles). Default is black.
outlineSize	defines the outline size of geoms (circles or tiles). Default is 1.
axisColor	user defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border)
axisLabelSize	The size of variable labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0
axisLabelColor	user defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels
axisLabelAngle.x	angle for x-axis-labels
axisLabelAngle.y	angle for y-axis-labels
breakTitleAt	Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title
breakLabelsAt	Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted
hideLegend	show or hide the legend. The legend indicates the strength of correlations by gradient colour fill.
legendTitle	the legend title, provided as string, e.g. legendTitle=c("Strength of correlation"). Default is NULL, hence no legend title is used.
printPlot	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).

References

http://strengejacke.wordpress.com/sjplot-r-package/
http://talesofr.wordpress.com/2013/05/05/ridiculously-photogenic-factors-heatmap-with-p-values/

Examples

```
# create data frame with 5 dichotomous (dummy) variables
df <- 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(df, axisLabels=items)
```

---

sjp.corr *Plot correlation matrix*

Description

Plot correlations as ellipses or tiles. Required parameter is either a data frame or a computed cor-object. 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 darkened.

Usage

```
sjp.corr(data, title = NULL, titleSize = 1.3, titleColor = "black",
  axisLabels = NULL, type = "circle", sortCorrelations = TRUE,
  decimals = 3, missingDeletion = "listwise", corMethod = "spearman",
  geomAlpha = 0.8, valueLabelColor = "black", valueLabelSize = 4.5,
  valueLabelAlpha = 1, circleSize = 15, outlineColor = "black",
  outlineSize = 1, axisColor = NULL, borderColor = NULL,
  axisLabelSize = 1.1, axisLabelColor = "gray30", axisLabelAngle.x = 0,
  axisLabelAngle.y = 0, breakTitleAt = 50, breakLabelsAt = 12,
  hideDiagCircle = TRUE, hideLegend = TRUE, legendTitle = NULL,
  showCorrelationValueLabels = TRUE, showCorrelationPValues = TRUE,
  pvaluesAsNumbers = FALSE, showTickMarks = FALSE, fillColor = NULL,
  majorGridColor = NULL, minorGridColor = NULL, theme = NULL,
  printPlot = TRUE)
```

Arguments

data	A correlation object, built with the R-cor-function, or a data frame which correlations should be calculated.
title	Title of the diagram, plotted above the whole diagram panel.
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
axisLabels	Labels for the x- andy y-axis. axisLabels are detected automatically if data is a data frame where each variable has a "variable.label" attribute (see sji.setVariableLabels) for details).
type	Indicates whether the geoms of correlation values should be plotted as "circle" (default) or as "tile".
sortCorrelations	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 showCorrelationValueLabels is TRUE.
missingDeletion	Indicates how missing values are treated. May be either "listwise" (default) or "pairwise".
corMethod	Indicates the correlation computation method. May be one of "spearman" (default), "pearson" or "kendall".
geomAlpha	Specify the transparancy (alpha value) of geom objects (circles or tiles). Default is 0.8.
valueLabelColor	The color of the value labels (numbers) inside the diagram. Default is "black".
valueLabelSize	The size of value labels in the diagram. Default is 4.5, recommended values range between 2 and 8.
valueLabelAlpha	Specify the transparancy (alpha value) of value labels. Default is 1.
circleSize	Specifies the circle size factor. The circle size depends on the correlation value multiplicated with this factor. Default is 15.
outlineColor	Defines the outline color of geoms (circles or tiles). Default is black.
outlineSize	Defines the outline size of geoms (circles or tiles). Default is 1.
axisColor	User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).
borderColor	User defined color of whole diagram border (panel border).
axisLabelSize	The size of variable labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0.
axisLabelColor	User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

`axisLabelAngle.x`
: Angle for x-axis-labels.

`axisLabelAngle.y`
: Angle for y-axis-labels.

`breakTitleAt`
: Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title. Default is 50.

`breakLabelsAt`
: Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted. Default is 12.

`hideDiagCircle`
: If `TRUE` (default), the geoms of the diagonal correlations (self-correlations with value "1") are not plotted. Only applies if parameter `type` is `"circle"`.

`hideLegend`
: Show or hide the legend. The legend indicates the strength of correlations by gradient colour fill. Default is `TRUE`, hence the legend is hidden.

`legendTitle`
: The legend title, provided as string, e.g. `legendTitle=c("Strength of correlation")`. Default is `NULL`, hence no legend title is used.

`showCorrelationValueLabels`
: Whether correlation values should be plotted to each geom

`showCorrelationPValues`
: Whether significance levels (p-values) of correlations should be plotted to each geom.

`pvaluesAsNumbers`
: If `TRUE`, the significance levels (p-values) are printed as numbers. if `FALSE` (default), asterisks are used.

`showTickMarks`
: Whether tick marks should be plotted or not. Default is `FALSE`.

`fillColor`
: A color palette for fillng the geoms. If not specified, the 5th diverging color palette from the color brewer palettes (RdBu) is used, resulting in red colors for negative and blue colors for positive correlations, that become lighter the weaker the correlations are. Use any color palette that is suitbale for the `scale_fill_gradientn` parameter of ggplot2.

`majorGridColor`
: Specifies the color of the major grid lines of the diagram background.

`minorGridColor`
: Specifies the color of the minor grid lines of the diagram background.

`theme`
: Specifies the diagram's background theme. Default (parameter `NULL`) is a gray background with white grids.

  - Use `"bw"` for a white background with gray grids
  - `"classic"` for a classic theme (black border, no grids)
  - `"minimal"` for a minimalistic theme (no border,gray grids)
  - `"none"` for no borders, grids and ticks or
  - `"themr"` if you are using the `ggthemr` package

  See http://rpubs.com/sjPlot/custplot for details and examples.

`printPlot`
: 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`) and the original correlation matrix (`corr.matrix`).

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2013/04/18/examples-for-sjplotting-functions-including-corre

See Also

sjt.corr

Examples

```
# create data frame with 5 random variables
df <- as.data.frame(cbind(rnorm(10), rnorm(10), rnorm(10), rnorm(10), rnorm(10)))

# plot correlation matrix using circles
sjp.corr(df)

# plot correlation matrix using square tiles without diagram background
sjp.corr(df, type="tile", theme="none")


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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

# we have high correlations here, because all items
# belong to one factor. See example from "sjp.pca".
sjp.corr(df, type="tile", theme="none", outlineColor="white", hideLegend=FALSE)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
sjp.corr(efc[,c(start:end)])
```

`sjp.emm.int` *Plot adjusted (estimated marginal) means of interaction (moderation) in linear models*

Description

Plot estimated marginal means of (significant) interaction terms in linear models (lm). This function may be used to plot differences in interventions between control and treatment groups over multiple time points.

Usage

```
sjp.emm.int(fit, swapPredictors = FALSE, plevel = 0.05, title = NULL,
  titleSize = 1.3, titleColor = "black", lowerBoundColor = "#3366cc",
  upperBoundColor = "#cc3300", colorPalette = "Set2", axisTitle.x = NULL,
  axisTitle.y = NULL, axisLabelColor = "gray30", axisLabelSize = 1.1,
  axisTitleColor = "black", axisTitleSize = 1.3, legendLabels = NULL,
  legendLabelSize = 0.9, legendLabelColor = "black",
  showValueLabels = FALSE, valueLabel.digits = 2, valueLabelSize = 4,
  valueLabelColor = "black", valueLabelAlpha = 0.8, breakTitleAt = 50,
  breakLegendLabelsAt = 20, breakAnnotationLabelsAt = 50,
  axisLimits.y = NULL, gridBreaksAt = NULL, theme = NULL,
  showTickMarks = TRUE, borderColor = NULL, axisColor = NULL,
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, printPlot = TRUE)
```

Arguments

`fit`	the fitted linear model (lm) object, including interaction terms
`swapPredictors`	if `TRUE`, the grouping variable and predictor on the x-axis are swapped.
`plevel`	Indicates at which p-value an interaction term is considered as significant. Default is 0.05 (5 percent).
`title`	a default title used for the plots. Default value is `NULL`, which means that each plot's title includes the dependent variable as well as the names of the interaction terms.
`titleSize`	The size of the plot title. Default is 1.3.
`titleColor`	The color of the plot title. Default is `"black"`.
`lowerBoundColor`	the color of the line indicating the lower bound of the interaction term (moderator value). Default value is `"#3366cc"` (blue-like)
`upperBoundColor`	the color of the line indicating the upper bound of the interaction term (moderator value). Default value is `"#cc3300"` (red-like)

colorPalette If the grouping variable has more than two levels, more than two colors are needed for plotting the lines. In this case, specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here. Alternatively, you can provide a vector of colors, i.e. c("blue", "red", "gren").

axisTitle.x a default title used for the x-axis. Default value is NULL, which means that each plot's x-axis uses the predictor's name as title.

axisTitle.y a default title used for the y-axis. Default value is NULL, which means that each plot's y-axis uses the dependent variable's name as title.

axisLabelColor the color value for the axis labels at the tick marks. Default value is "darkgray".

axisLabelSize The size of axis labels. Default is 1.1, recommended values range between 0.5 and 3.0

axisTitleColor the color value for the axis titles (both x and y). Default value is "black".

axisTitleSize The size of axis titles (both x and y). Default is 1.3, recommended values range between 0.5 and 3.0

legendLabels Labels for the guide/legend. Default is NULL, so the name of the predictor with min/max-effect is used as legend label.

legendLabelSize
The size of legend labels. Default is 0.9, recommended values range between 0.5 and 3.0

legendLabelColor
user defined color for legend labels. If not specified, black will be used for the labels

showValueLabels
if TRUE, value labels are plotted along the lines. Default is FALSE.

valueLabel.digits
the amount of digits of the displayed value labels. Defaults to 2.

valueLabelSize size of the value labels. Default is 4. Recommended Values range from 2 to 8

valueLabelColor
colour of the values inside the diagrams. Only applies, when parameter showValueLabels is set to TRUE. Use any valid colour value, e.g. valueLabelColor="grey50" or valueLabelColor=c("#cc3366"). Default is "black".

valueLabelAlpha
the alpha level (transparancy) of the value labels. Default is 0.8, use any value from 0 to 1.

breakTitleAt Wordwrap for diagram's title. Determines how many chars of the title are displayed in one line and when a line break is inserted. Default is 50.

breakLegendLabelsAt
Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted. Default is 20.

breakAnnotationLabelsAt
Wordwrap for diagram annotation labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted. Default is 50. Only applies if showInterceptLine is TRUE.

axisLimits.y	A vector with two values, defining the lower and upper limit from the y-axis. By default, this value is NULL, i.e. axis limits will be calculated upon the range of y-values.
gridBreaksAt	Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is NULL.
theme	Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids. • Use "bw" for a white background with gray grids • "classic" for a classic theme (black border, no grids) • "minimal" for a minimalistic theme (no border,gray grids) • "none" for no borders, grids and ticks or • "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the lowerBoundColor parameter) See http://rpubs.com/sjPlot/custplot for details and examples.
showTickMarks	Whether tick marks of axes should be shown or not
borderColor	user defined color of whole diagram border (panel border)
axisColor	user defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).
majorGridColor	specifies the color of the major grid lines of the diagram background
minorGridColor	specifies the color of the minor grid lines of the diagram background
hideGrid.x	If TRUE, the x-axis-gridlines are hidden. Default if FALSE.
hideGrid.y	If TRUE, the y-axis-gridlines are hidden. Default if FALSE.
printPlot	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-objects with the complete plot-list (plot.list) as well as the data frame that were used for setting up the ggplot-objects (df.list).

Note

Please note that all interaction terms have to be of type factor! Furthermore, 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

- http://rpubs.com/sjPlot/sjpemmint
- http://strengejacke.wordpress.com/2014/08/19/visualize-pre-post-comparison-of-intervention-rst
- http://www.theanalysisfactor.com/using-adjusted-means-to-interpret-moderators-in-analysis-of-c

See Also

sjp.lm.int
sjp.reglin
sjp.aov1
sjp.lm.ma

Examples

```
## Not run:
# 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.

# prepare data frame
df <- data.frame(mpg=mtcars$mpg,vs=factor(mtcars$vs),am=factor(mtcars$am))
# fit "dummy" model.
fit <- lm(mpg~vs+am+vs:am, data=df)
# show summary to see significant interactions
summary(fit)

# plot marginal means of interaction terms
# note we have to adjust plevel, because no interaction
# is significant
sjp.emm.int(fit, plevel=1)
# plot marginal means of interaction terms, including value labels
sjp.emm.int(fit, plevel=1, showValueLabels=TRUE)


# load sample data set
data(efc)
# create data frame with variables that should be included
# in the model
df <- as.data.frame(cbind(burden=efc$neg_c_7,
                          sex=efc$c161sex,
                          education=efc$c172code))
# convert gender predictor to factor
df$sex <- factor(df$sex)
df$education <- factor(df$education)
# name factor levels and dependent variable
levels(df$sex) <- c("female", "male")
levels(df$education) <- c("low", "mid", "high")
df$burden <- sji.setVariableLabels(df$burden, "care burden")
# fit "dummy" model
fit <- lm(burden ~ .*., data=df, na.action=na.omit)
summary(fit)

# plot marginal means of interactions, no interaction found
sjp.emm.int(fit)
# plot marginal means of interactions, including those with p-value up to 1
sjp.emm.int(fit, plevel=1)
# swap predictors
sjp.emm.int(fit, plevel=1, swapPredictors=TRUE)
## End(Not run)
```

`sjp.frq` *Plot frequencies of (count) variables*

Description

Plot frequencies of a (count) variable as bar graph, histogram, box plot etc. using ggplot.

Usage

```
sjp.frq(varCount, title = NULL, titleSize = 1.3, titleColor = "black",
  weightBy = NULL, weightByTitleString = NULL, interactionVar = NULL,
  maxYlim = FALSE, upperYlim = NULL, order = "none", type = "bars",
  axisLabels.x = NULL, interactionVarLabels = NULL, axisLabelAngle.x = 0,
  axisLabelSize = 1.1, axisLabelColor = "gray30", valueLabelSize = 4,
  valueLabelColor = "black", breakTitleAt = 50, breakLabelsAt = 12,
  gridBreaksAt = NULL, barWidth = 0.6, dotSize = 4, barColor = NULL,
  barAlpha = 1, barOutline = FALSE, barOutlineSize = 0.2,
  innerBoxPlotWidth = 0.15, innerBoxPlotDotSize = 3, borderColor = NULL,
  axisColor = NULL, barOutlineColor = "black", majorGridColor = NULL,
  minorGridColor = NULL, hideGrid.x = FALSE, hideGrid.y = FALSE,
  expand.grid = FALSE, showValueLabels = TRUE, showCountValues = TRUE,
  showPercentageValues = TRUE, showAxisLabels.x = TRUE,
  showAxisLabels.y = TRUE, showTickMarks = TRUE,
  showMeanIntercept = FALSE, showMeanValue = TRUE,
  showStandardDeviation = TRUE, showNormalCurve = FALSE,
  showStandardNormalCurve = FALSE, adjustNormalCurve.x = FALSE,
  meanInterceptLineType = 2, meanInterceptLineSize = 0.5,
  normalCurveColor = "red", normalCurveSize = 0.8, normalCurveAlpha = 0.4,
  axisTitle.x = NULL, axisTitle.y = NULL, axisTitleColor = "black",
  axisTitleSize = 1.3, startAxisAt = "auto", hist.skipZeros = FALSE,
  autoGroupAt = NULL, theme = NULL, flipCoordinates = FALSE,
  labelPos = "outside", na.rm = TRUE, printPlot = TRUE)
```

Arguments

`varCount`	The variable which frequencies should be plotted.
`title`	Title of diagram as string. Example: `title=c("my title")`. Use `"auto"` to automatically detect variable names that will be used as title (see `sji.setVariableLabels`) for details).
`titleSize`	The size of the plot title. Default is 1.3.
`titleColor`	The color of the plot title. Default is `"black"`.
`weightBy`	A weight factor that will be applied to weight all cases from `varCount`. default is `NULL`, so no weights are used.
`weightByTitleString`	If a weight factor is supplied via the parameter `weightBy`, the diagram's title may indicate this with a remark. Default is `NULL`, so the diagram's title will

not be modified when cases are weighted. Use a string as parameter, e.g.: `weightByTitleString=" (weighted)"`.

`interactionVar` An interaction variable which can be used for box plots. Divides the observations in `varCount` into the factors (sub groups) of `interactionVar`. Only applies when parameter `"type"` is `"box"` or `"violin"` (resp. their alternative strings like `"boxplot"`, `"boxplots"` or `"v"`).

`maxYlim` Indicates how to calculate the maximum limit of the y-axis. If `TRUE`, the upper y-limit corresponds to the amount of cases, i.e. y-axis for each plot of a data base are the same. If `FALSE` (default), the maximum y-axis depends on the highest count of a variable's answer category. In this case, the y-axis breaks may change, depending on the variable.

`upperYlim` Uses a pre-defined upper limit for the y-axis. Overrides the `maxYlim` parameter.

`order` Determines whether categories on x-axis should be order according to the frequencies or not. Default is `"none"`, so categories are not ordered by frequency. Use `"asc"` or `"desc"` for sorting categories ascending or descending in relation to the frequencies.

`type` Specifies the type of distribution plot that will be plotted.

- `"bar"`, `"bars"` or `"b"` for simple bars (the default setting).
- `"dots"` or `"dot"` for a dot plot.
- `"h"`, `"hist"` or `"histogram"` for a histogram.
- `"line"`, `"lines"` or `"l"` for a histogram with filled area with line.
- `"dens"`, `"d"` or `"density"` for a density plot.
- `"box"`, `"boxplot"` or `"boxplots"` for box plots.
- `"v"` or `"violin"` for violin plots.

`axisLabels.x` Labels for the x-axis breaks. Example: `axisLabels.x=c("Label1", "Label2", "Label3")`. Note: If you use the `sji.SPSS` function and the `sji.getValueLabels` function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.

`interactionVarLabels`
Labels for the x-axis breaks when having interaction variables included. These labels replace the `axisLabels.x`. Only applies, when using box or violin plots (i.e. `"type"` is `"box"` or `"violin"`) and `interactionVar` is not `NULL`. Example: See `axisLabels.x`.

`axisLabelAngle.x`
Angle for axis-labels.

`axisLabelSize` The size of axis labels of both x and y axis. Default is 1.1, recommended values range between 0.5 and 3.0.

`valueLabelSize` The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8.

`breakTitleAt` Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

`breakLabelsAt` Determines how many chars of the labels are displayed in one line and when a line break is inserted into the axis labels.

gridBreaksAt Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed.

barWidth Width of bars. Default is 0.6, recommended values range from 0.2 to 2.0

dotSize The size of dots in case of dot-plots (type="dots").

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

innerBoxPlotDotSize
Size of mean dot insie a violin plot. Applies only when type is set to "violin" or "box".

barColor User defined color for bars. If not specified, a default blue color palette will be used for the bar charts.

barAlpha Specify the transparancy (alpha value) of bars.

axisLabelColor User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

borderColor User defined color of whole diagram border (panel border).

axisColor User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

barOutline If TRUE, each bar gets a colored outline. Default is FALSE.

barOutlineSize The size of the bar outlines. Only applies if barOutline is TRUE. Default is 0.2

barOutlineColor
The color of the bar outline. Only applies, if barOutline is TRUE.

majorGridColor Specifies the color of the major grid lines of the diagram background.

minorGridColor Specifies the color of the minor grid lines of the diagram background.

hideGrid.x If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

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

showValueLabels
Whether counts and percentage values should be plotted to each bar. Default is TRUE.

showCountValues
If TRUE (default), count values are be plotted to each bar. If FALSE, count values are removed.

showPercentageValues
If TRUE (default), percentage values are be plotted to each bar, if FALSE, percentage-values are removed.

showAxisLabels.x
Whether x axis labels (category names) should be shown or not.

showAxisLabels.y
Whether y axis labels (count values) should be shown or not.

showTickMarks Whether tick marks of axes should be shown or not.

showMeanIntercept
: If TRUE, a vertical line in histograms is drawn to indicate the mean value of the count variables. Only applies to histogram-charts.

showMeanValue
: If TRUE (default value), the mean value is printed to the vertical line that indicates the mean value of the count variables. Only applies to histogram-charts.

showStandardDeviation
: If TRUE, the standard deviation is annotated as shaded rectangle around the mean intercept line. Only applies to histogram-charts.

meanInterceptLineType
: The linetype of the mean intercept line. Only applies to histogram-charts and when showMeanIntercept is TRUE.

meanInterceptLineSize
: The size of the mean intercept line. Only applies to histogram-charts and when showMeanIntercept is TRUE.

showNormalCurve
: 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).

showStandardNormalCurve
: If TRUE, a normal curve, which is not adjusted to the data (thus representing a "true" standard normal curve, which is, however, still just an approximation), is plotted over the histogram or density plot. Default is FALSE. Only applies when histograms or density plots are plotted (see type).

adjustNormalCurve.x
: If TRUE and showStandardNormalCurve is also TRUE, the x-axis-start of the standard normal curve starts with the x-axis limits of the graph. This is only necessary, if minimum value of varCount is larger than 0 or 1.

normalCurveColor
: Specify the color of the normal curve line. Only applies if showNormalCurve is TRUE.

normalCurveSize
: Specifiy the size of the normal curve line. Only applies if showNormalCurve is TRUE.

normalCurveAlpha
: Specify the transparancy (alpha value) of the normal curve. Only applies if showNormalCurve is TRUE.

valueLabelColor
: The color of the value labels (numbers) inside the digram.

axisTitle.x
: A label for the x axis. useful when plotting histograms with metric scales where no category labels are assigned to the x axis. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

axisTitle.y
: A label for the y axis. useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

axisTitleColor
: The color of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

axisTitleSize — the size of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels. Default is 1.3.

hist.skipZeros — If TRUE, zero counts (categories with no answer) in varCout are omitted when drawing histrograms, and the mapping is changed to stat_bin. Only applies to histograms (see type). Use this parameter to get identical results to the default qplot or geom_histogram histogram plots of ggplot. You may need to adjust the barWidth parameter for better visual results (which, by ggplot-default, is 1/30 of the x-axis-range).

startAxisAt — Determines the first value on the x-axis. By default, this value is set to "auto", i.e. the value range on the x axis starts with the lowest value of varCount. If you set startAxisAt to 1, you may have zero counts if the lowest value of varCount is larger than 1 and hence no bars plotted for these values in such cases.

autoGroupAt — A value indicating at which length of unique values of varCount the variable is automatically grouped into smaller units (see sju.groupVar). If varCount has large numbers of unique values, too many bars for the graph have to be plotted. Hence it's recommended to group such variables. For example, if autoGroupAt is 50, i.e. if varCount has 50 and more unique values it will be grouped using sju.groupVar with groupsize="auto" parameter. By default, the maximum group count is 30. However, if autoGroupAt is less than 30, autoGroupAt groups are built. Default value for autoGroupAt is NULL, i.e. auto-grouping is off.

theme — Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates — If TRUE, the x and y axis are swapped. Default is FALSE.

labelPos — If flipCoordinates is TRUE, use this parameter to specify value label position. Can be either "inside" or "outside" (default). You may specify initial letter only. If flipCoordinates is FALSE, this parameter will be ignored.

na.rm — If TRUE, missings are not included in the frequency calculation and diagram plot.

printPlot — 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).

References

- http://rpubs.com/sjPlot/sjpfrq
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjt.frq

Examples

```
# ---------------
# boxplot
# ---------------
sjp.frq(ChickWeight$weight, type="box")

# ---------------
# histogram
# ---------------
sjp.frq(discoveries, type="hist", showMeanIntercept=TRUE)
# histogram with minimal theme and w/0 labels
sjp.frq(discoveries, type="hist", showMeanIntercept=TRUE,
        theme="minimal", minorGridColor="white",
        showTickMarks=FALSE, hideGrid.x=TRUE,
        showValueLabels=FALSE)

# ---------------
# violin plot
# ---------------
sjp.frq(ChickWeight$weight, type="v")

# ---------------
# bar plot
# ---------------
sjp.frq(ChickWeight$Diet)
sjp.frq(ChickWeight$Diet, maxYlim=TRUE)

# ---------------
# bar plot with EUROFAMCARE sample dataset
# dataset was importet from an SPSS-file, using:
# efc <- sji.SPSS("efc.sav", enc="UTF-8")
# ---------------
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
sjp.frq(as.factor(efc$e15relat),
        title=efc.var[['e15relat']],
        axisLabels.x=efc.val['e15relat'],
        axisLabelAngle.x=90)

# bar plot with EUROFAMCARE sample dataset
# grouped variable
ageGrp <- sju.groupVar(efc$e17age)
```

```
ageGrpLab <- sju.groupVarLabels(efc$e17age)
sjp.frq(ageGrp,
        title=efc.var[['e17age']],
        axisLabels.x=ageGrpLab)
# minimal theme
sjp.frq(ageGrp,
        title=efc.var[['e17age']],
        axisLabels.x=ageGrpLab,
        theme="minimal",
        minorGridColor="white",
        showTickMarks=FALSE,
        hideGrid.x=TRUE)

# ---------------
# box plots with interaction variable
# the following example is equal to the function call
# sjp.grpfrq(efc$e17age, efc$e16sex, type="box")
# ---------------
sjp.frq(efc$e17age,
        title=paste(efc.var[['e17age']], "by", efc.var[['e16sex']]),
        interactionVar=efc$e16sex,
        interactionVarLabels=efc.val['e16sex'],
        type="box")

# ------------------------------------------------
# auto-detection of value labels and variable names
# ------------------------------------------------
efc <- sji.setVariableLabels(efc, sji.getVariableLabels(efc))

# negative impact scale, ranging from 7-28, assuming that
# variable scale (lowest value) starts with 1
sjp.frq(efc$neg_c_7, startAxisAt=1, title="auto")

# negative impact scale, ranging from 7-28, using
# automatic detection of start index of x-axis
sjp.frq(efc$neg_c_7, axisTitle.x="auto")

# ------------------------------------------------
# Simulate ggplot-default histogram, using "hist.skipZeros"
# and adjusted "barWidth".
# ------------------------------------------------
sjp.frq(efc$c160age, type="h", hist.skipZeros=TRUE, barWidth=1)
```

`sjp.glm` *Plot odds ratios (forest plots)*

Description

Plot odds ratios (exponentiated coefficients) with confidence intervalls as bar chart or dot plot

Usage

```
sjp.glm(fit, sortOdds = TRUE, title = NULL, titleSize = 1.3,
  titleColor = "black", axisLabels.y = NULL, axisLabelSize = 1.1,
  axisLabelAngle.x = 0, axisLabelAngle.y = 0, axisLabelColor = "gray30",
  axisTitle.x = "Odds Ratios", axisTitleSize = 1.2,
  axisTitleColor = c("#444444"), axisLimits = NULL, breakTitleAt = 50,
  breakLabelsAt = 12, gridBreaksAt = 0.5, transformTicks = FALSE,
  type = "dots", hideErrorBars = FALSE, errorBarWidth = 0,
  errorBarSize = 0.8, errorBarLineType = 1, pointSize = 3,
  colorPalette = "Paired", barColor = NULL, barWidth = 0.3,
  barAlpha = 1, valueLabelColor = "black", valueLabelSize = 4.5,
  valueLabelAlpha = 1, axisColor = NULL, borderColor = NULL,
  barOutline = FALSE, barOutlineColor = "black", interceptLineType = 2,
  interceptLineColor = "grey70", majorGridColor = NULL,
  minorGridColor = NULL, hideGrid.x = FALSE, hideGrid.y = FALSE,
  theme = NULL, flipCoordinates = TRUE, showIntercept = FALSE,
  showAxisLabels.y = TRUE, showTickMarks = TRUE, showValueLabels = TRUE,
  labelDigits = 2, showPValueLabels = TRUE, showModelSummary = TRUE,
  printPlot = TRUE)
```

Arguments

fit	The fitted model of a logistic regression (or any other glm-object).
sortOdds	If TRUE (default), the odds ratios are ordered according their OR value from highest first to lowest last. Use FALSE if you don't want to change the order of the predictors.
title	Diagram's title as string. Example: title=c("my title")
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
axisLabels.y	Labels of the predictor variables (independent vars, odds) that are used for labelling the axis. Passed as vector of strings. Example: axisLabels.y=c("Label1", "Label2", "Label Note: If you use the sji.SPSS function and the sji.getValueLabels function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.
axisLabelSize	The size of value labels in the diagram. Default is 1.1, recommended values range between 0.7 and 3.0
showAxisLabels.y	Whether odds names (predictor labels) should be shown or not.
showTickMarks	Whether tick marks of axes should be shown or not.
axisTitle.x	A label ("title") for the x axis.
axisTitleColor	The color of the x axis label.
axisTitleSize	The size of the x axis label.
axisLimits	Defines the range of the axis where the beta coefficients and their confidence intervalls are drawn. By default, the limits range from the lowest confidence interval to the highest one, so the diagram has maximum zoom. Use your own values as 2-value-vector, for instance: limits=c(-0.8,0.8).

axisLabelAngle.x
: Angle for axis-labels where the odds ratios are printed. Note that due to the coordinate flip, the acutal y-axis with odds ratio labels are appearing on the x-axis.

axisLabelAngle.y
: Angle for axis-labels where the predictor labels (axisLabels.y) are printed. Note that due to the coordinate flip, the acutal x-axis with predictor labels are appearing on the y-axis.

breakTitleAt
: Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title

breakLabelsAt
: Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted

gridBreaksAt
: Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is 0.5

transformTicks
: if TRUE, the grid bars have exponential distances, i.e. they visually have the same distance from one grid bar to the next. Default is FALSE which means that grids are plotted on every gridBreaksAt's position, thus the grid bars become narrower with higher odds ratio values.

type
: Indicates Whether the Odds Ratios should be plotted as "dots" (aka forest plots, default) or as "bars".

hideErrorBars
: If TRUE, the error bars that indicate the confidence intervals of the odds ratios are not shown. Only applies if parameter type is bars. Default value is FALSE.

pointSize
: The size of the points that indicate the beta-value. Default is 3.

barColor
: A vector with colors for representing the odds values (i.e. points and error bars in case the parameter type is "dots" or the bar charts in case of "bars"). The first color value indicates odds ratio values larger than 1, the second color value indicates odds ratio values lower or equal to 1. Default colors is a blue/red-scheme. You can also use:

  - "bw" or "black" for only one colouring in almost black
  - "gray", "grey" or "gs" for a grayscale
  - "brewer" for colours from the color brewer palette.

  If barColors is "brewer", use the colorPalette parameter to specify a palette of the http://colorbrewer2.org Else specify your own color values as vector (e.g. barColors=c("#f00000", "#00ff00")).

colorPalette
: If parameter barColor is brewer, specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here.

barWidth
: The width of the bars in bar charts. only applies if parameter type is bars. Default is 0.5

barAlpha
: The alpha value of the bars in bar charts. only applies if parameter type is bars. Default is 1

axisLabelColor
: Colour of the tick labels at the axis (variable names, odds names).

valueLabelColor
: The colour of the odds values. These values are printed above the plots respectively beside the bar charts. default color is "black".

valueLabelSize Size of the value labels. Drfault is 4.5. Recommended Values range from 2 to 8

valueLabelAlpha
The alpha level (transparancy) of the value labels. Default is 1, use any value from 0 to 1.

axisColor User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

borderColor User defined color of whole diagram border (panel border).

barOutline If TRUE, each bar gets a colored outline. only applies if parameter type is bars. Default is FALSE.

barOutlineColor
The color of the bar outline. Only applies, if barOutline is set to TRUE. Default is black.

interceptLineType
The linetype of the intercept line (zero point). Default is 2 (dashed line).

interceptLineColor
The color of the intercept line. Default value is "grey70".

errorBarWidth The width of the error bar ends. Default is 0

errorBarSize The size (thickness) of the error bars. Default is 0.8

errorBarLineType
The linetype of error bars. Default is 1 (solid line).

majorGridColor Specifies the color of the major grid lines of the diagram background.

minorGridColor Specifies the color of the minor grid lines of the diagram background.

hideGrid.x If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

theme Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates
If TRUE (default), predictors are plotted on the left y-axis and estimate values are plotted on the x-axis.

showIntercept If TRUE, the intercept of the fitted model is also plotted. Default is FALSE. Please note that due to exp-transformation of estimates, the intercept in some cases can not be calculated, thus the function call is interrupted and no plot printed.

showValueLabels
Whether the beta and standardized beta values should be plotted to each dot or not.

labelDigits — The amount of digits for rounding the estimations (see showValueLabels). Default is 2, i.e. estimators have 2 digits after decimal point.

showPValueLabels — Whether the significance levels of each coefficient should be appended to values or not.

showModelSummary — If TRUE (default), a summary of the regression model with Intercept, R-square, F-Test and AIC-value is printed to the lower right corner of the diagram.

printPlot — 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

Based on the script from surefoss: http://www.surefoss.org/dataanalysis/plotting-odds-ratios-aka-a-forrestp

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2013/03/22/plotting-lm-and-glm-models-with-ggplot-rstats/
- http://www.surefoss.org/dataanalysis/plotting-odds-ratios-aka-a-forrestplot-with-ggplot2/

See Also

sjp.glm.ma

Examples

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

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

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

# print Odds Ratios as bars
sjp.glm(fitOR, type="bars")


# ------------------------------
# Predictors for negative impact
# of care. Data from the EUROFAMCARE
# sample dataset
```

```
# ------------------------------
data(efc)
# retrieve predictor variable labels
labs <- sji.getVariableLabels(efc)
predlab <- c(labs[['c161sex']],
             labs[['e42dep']],
             paste0(labs[['c172code']], " (mid)"),
             paste0(labs[['c172code']], " (high)"))
# create binary response
y <- ifelse(efc$neg_c_7<median(na.omit(efc$neg_c_7)), 0, 1)
# create dummy variables for educational status
edu.mid <- ifelse(efc$c172code==2, 1, 0)
edu.high <- ifelse(efc$c172code==3, 1, 0)
# create data frame for fitted model
df <- na.omit(as.data.frame(cbind(y,
                as.factor(efc$c161sex),
                as.factor(efc$e42dep),
                as.factor(edu.mid),
                as.factor(edu.high))))
# fit model
fit <- glm(y ~., data=df, family=binomial(link="logit"))
# plot odds
sjp.glm(fit, title=labs[['neg_c_7']], axisLabels.y=predlab)
```

`sjp.glm.ma` *Plot model assumptions of glm's*

Description

Plots model assumptions of generalized linear models to verify if generalized linear regression is applicable

Usage

```
sjp.glm.ma(logreg, showOriginalModelOnly = TRUE)
```

Arguments

`logreg`	a fitted `glm`-model
`showOriginalModelOnly`	if `TRUE` (default), only the model assumptions of the fitted model `logreg` are plotted. if `FALSE`, the model assumptions of an updated model where outliers are automatically excluded are also plotted.

Value

an updated fitted generalized linear model where outliers are dropped out.

See Also

sjp.glm

Examples

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

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

# plot model assumptions
sjp.glm.ma(fitOR)
```

---

sjp.glmm *Plot odds ratios (forest plots) of multiple fitted glm's*

Description

Plot odds ratios (forest plots) of multiple fitted glm's with confidence intervalls in one plot.

Usage

```
sjp.glmm(..., title = NULL, titleSize = 1.3, titleColor = "black",
  labelDependentVariables = NULL, legendDepVarTitle = "Dependent Variables",
  legendPValTitle = "p-level", stringModel = "Model", axisLabels.y = NULL,
  axisLabelSize = 1.1, axisLabelAngle.x = 0, axisLabelAngle.y = 0,
  axisLabelColor = "gray30", axisTitle.x = "Odds Ratios",
  axisTitleSize = 1.2, axisTitleColor = c("#444444"), axisLimits = NULL,
  breakTitleAt = 50, breakLabelsAt = 12, breakLegendAt = 20,
  gridBreaksAt = 0.5, transformTicks = FALSE, errorBarWidth = 0,
  errorBarSize = 0.5, errorBarLineType = 1, pointSize = 3,
  modelPlotSpace = 0.4, colorPalette = "Paired", modelColors = NULL,
  valueLabelColor = "black", valueLabelSize = 4, valueLabelAlpha = 1,
  nsAlpha = 1, usePShapes = FALSE, axisColor = NULL, borderColor = NULL,
  interceptLineType = 2, interceptLineColor = "grey70",
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, theme = NULL, flipCoordinates = TRUE,
  legendPos = "right", legendSize = 1, legendBorderColor = "white",
  legendBackColor = "white", showIntercept = FALSE,
  showAxisLabels.y = TRUE, showTickMarks = TRUE, showValueLabels = TRUE,
  labelDigits = 2, showPValueLabels = TRUE, printPlot = TRUE)
```

Arguments

...	One or more fitted glm-objects.
title	Diagram's title as string. Example: title=c("my title")
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
labelDependentVariables	Labels of the dependent variables of all fitted models which have been used as first parameter(s), provided as char vector.
legendDepVarTitle	A character vector used for the title of the dependent variable's legend. Default is "Dependent Variables".
legendPValTitle	A character vector used for the title of the significance level's legend. Default is "p-level". Only applies if usePShapes is TRUE.
stringModel	String constant used as legend text for the model names in case no labels for the dependent variables are provided (see labelDependentVariables). Default is "Model".
axisLabels.y	Labels of the predictor variables (independent vars, odds) that are used for labelling the axis. Passed as vector of strings. Example: axisLabels.y=c("Label1", "Label2", "Label Note: If you use the sji.SPSS function and the sji.getValueLabels function, you receive a list object with label strings. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.
axisLabelSize	The size of value labels in the diagram. Default is 1.1, recommended values range between 0.7 and 3.0
showAxisLabels.y	Whether odds names (predictor labels) should be shown or not.
showTickMarks	Whether tick marks of axes should be shown or not.
axisTitle.x	A label ("title") for the x axis.
axisTitleColor	The color of the x axis label.
axisTitleSize	The size of the x axis label.
axisLimits	Defines the range of the axis where the beta coefficients and their confidence intervalls are drawn. By default, the limits range from the lowest confidence interval to the highest one, so the diagram has maximum zoom. Use your own values as 2-value-vector, for instance: limits=c(-0.8,0.8).
axisLabelAngle.x	Angle for axis-labels where the odds ratios are printed. Note that due to the coordinate flip, the acutal y-axis with odds ratios are appearing on the x-axis.
axisLabelAngle.y	Angle for axis-labels where the predictor labels (axisLabels.y) are printed. Note that due to the coordinate flip, the acutal x-axis with predictor labels are appearing on the y-axis.
breakTitleAt	Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title

breakLabelsAt — Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted

breakLegendAt — Wordwrap for legend, i.e. names of the dependent variables of each fitted model. See parameter labelDependentVariables. Determines how many chars of each dependent variable name is displayed in one line in the legend and when a line break is inserted

gridBreaksAt — Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is 0.5

transformTicks — if TRUE, the grid bars have exponential distances, i.e. they visually have the same distance from one grid bar to the next. Default is FALSE which means that grids are plotted on every gridBreaksAt's position, thus the grid bars become narrower with higher odds ratio values.

pointSize — The size of the points that indicate the beta-value. Default is 3.

modelPlotSpace — Defines the space between the dots and error bars of the plotted fitted models. Default is 0.3.

modelColors — A vector with colors for representing the odds values (i.e. points and error bars) of the different fitted models. Thus, the length of this vector must be equal to the length of supplied fitted models, so each model is represented by its own color. You can use:

- "bw" or "black" for only one colouring in almost black
- "gray", "grey" or "gs" for a grayscale
- "brewer" for colours from the color brewer palette.

If modelColors is "brewer", use the colorPalette parameter to specify a palette of the http://colorbrewer2.org Else specify your own color values as vector (e.g. modelColors=c("#f00000", "#00ff00")).

colorPalette — If parameter modelColors is brewer, specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here.

axisLabelColor — Colour of the tick labels at the axis (variable names, odds names).

valueLabelColor — The colour of the odds values. These values are printed above the plots respectively beside the bar charts. default color is "black".

valueLabelSize — Size of the value labels. Drfault is 4. Recommended Values range from 2 to 8

valueLabelAlpha — The alpha level (transparancy) of the value labels. Default is 1, use any value from 0 to 1.

nsAlpha — The alpha level (transparancy) of non significant predicors. Points and error bars are affected by this value and plotted with a slight transparancy. Default is 1.

usePShapes — If TRUE, significant levels are distinguished by different point shapes and a related legend is plotted. Default is FALSE.

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

borderColor — User defined color of whole diagram border (panel border).

`interceptLineType`
: The linetype of the intercept line (zero point). Default is `2` (dashed line).

`interceptLineColor`
: The color of the intercept line. Default value is `"grey70"`.

`errorBarWidth`
: The width of the error bar ends. Default is `0`

`errorBarSize`
: The size (thickness) of the error bars. Default is `0.5`

`errorBarLineType`
: The linetype of error bars. Default is `1` (solid line).

`majorGridColor`
: Specifies the color of the major grid lines of the diagram background.

`minorGridColor`
: Specifies the color of the minor grid lines of the diagram background.

`hideGrid.x`
: If `TRUE`, the x-axis-gridlines are hidden. Default if `FALSE`.

`hideGrid.y`
: If `TRUE`, the y-axis-gridlines are hidden. Default if `FALSE`.

`theme`
: Specifies the diagram's background theme. Default (parameter `NULL`) is a gray background with white grids.
  - Use `"bw"` for a white background with gray grids
  - `"classic"` for a classic theme (black border, no grids)
  - `"minimal"` for a minimalistic theme (no border,gray grids)
  - `"none"` for no borders, grids and ticks or
  - `"themr"` if you are using the `ggthemr` package

  See http://rpubs.com/sjPlot/custplot for details and examples.

`flipCoordinates`
: If `TRUE` (default), predictors are plotted on the left y-axis and estimate values are plotted on the x-axis.

`legendPos`
: The position of the legend, if a legend is drawn. 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.

`legendSize`
: The text size of the legend. Default is 1. Relative size, so recommended values are from 0.3 to 2.5

`legendBorderColor`
: Color of the legend's border. Default is `"white"`, so no visible border is drawn.

`legendBackColor`
: Fill color of the legend's background. Default is `"white"`, so no visible background is drawn.

`showIntercept`
: If `TRUE`, the intercept of the fitted model is also plotted. Default is `FALSE`. Please note that due to exp-transformation of estimates, the intercept in some cases can not be calculated, thus the function call is interrupted and no plot printed.

`showValueLabels`
: Whether the beta and standardized beta values should be plotted to each dot or not.

`labelDigits`
: The amount of digits for rounding the estimations (see `showValueLabels`). Default is 2, i.e. estimators have 2 digits after decimal point.

`showPValueLabels`
: Whether the significance levels of each coefficient should be appended to values or not.

printPlot — 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).

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2014/01/29/comparing-multiple-glm-in-one-graph-rstats/

See Also

sjp.glm
sjt.glm
sjp.glm.ma
sjp.lmm

Examples

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

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

# plot multiple models
sjp.glmm(fitOR1, fitOR2, fitOR3)

# plot multiple models with legend labels and point shapes instead of value  labels
sjp.glmm(fitOR1, fitOR2, fitOR3,
         labelDependentVariables=c("Fertility", "Infant Mortality", "Agriculture"),
         showValueLabels=FALSE,
         showPValueLabels=FALSE,
         usePShapes=TRUE,
         nsAlpha=0.2)
```

`sjp.grpfrq` *Plot grouped or stacked frequencies*

Description

Plot grouped or stacked frequencies of variables as bar/dor graphs, box or violin plots, histograms etc. using ggplot.

Usage

```
sjp.grpfrq(varCount, varGroup, weightBy = NULL, weightByTitleString = NULL,
  interactionVar = NULL, type = "bars", dotSize = 4, hideLegend = FALSE,
  maxYlim = FALSE, upperYlim = NULL, useFacetGrid = FALSE, title = NULL,
  titleSize = 1.3, titleColor = "black", legendTitle = NULL,
  axisLabels.x = NULL, axisLabelSize = 1.1, axisLabelColor = "gray30",
  axisLabelAngle.x = 0, interactionVarLabels = NULL, legendLabels = NULL,
  valueLabelSize = 4, valueLabelColor = "black", breakTitleAt = 50,
  breakLabelsAt = 12, breakLegendTitleAt = 20, breakLegendLabelsAt = 20,
  gridBreaksAt = NULL, barPosition = "dodge", barWidth = 0.6,
  barSpace = 0.1, barColor = NULL, barAlpha = 1,
  innerBoxPlotWidth = 0.15, innerBoxPlotDotSize = 3,
  colorPalette = "GnBu", lineType = 1, lineSize = 1, lineAlpha = 1,
  smoothLines = FALSE, borderColor = NULL, axisColor = NULL,
  barOutline = FALSE, barOutlineSize = 0.2, barOutlineColor = "black",
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, expand.grid = FALSE, showValueLabels = TRUE,
  showCountValues = TRUE, showPercentageValues = TRUE,
  showAxisLabels.x = TRUE, showAxisLabels.y = TRUE, showTickMarks = TRUE,
  showPlotAnnotation = TRUE, showMeanIntercept = FALSE,
  showMeanValue = TRUE, showStandardDeviation = FALSE,
  showTableSummary = TRUE, summaryLabelColor = "black",
  showGroupCount = FALSE, tableSummaryPos = "r",
  meanInterceptLineType = 2, meanInterceptLineSize = 0.5,
  axisTitle.x = NULL, axisTitle.y = NULL, axisTitleColor = "black",
  axisTitleSize = 1.3, autoGroupAt = NULL, startAxisAt = "auto",
  theme = NULL, legendPos = "right", legendSize = 1,
  legendBorderColor = "white", legendBackColor = "white",
  flipCoordinates = FALSE, labelPos = "outside", na.rm = TRUE,
  printPlot = TRUE)
```

Arguments

`varCount`	The variable which frequencies should be plotted. The counts of this variable are along the y-axis, the variable's categories on the x-axis.
`varGroup`	the grouping variable, where each value represents a single bar chart within each category of the `varCount` variable.

`weightBy` A weight factor that will be applied to weight all cases from `varCount`.

`weightByTitleString`
If a weight factor is supplied via the parameter `weightBy`, the diagram's title may indicate this with a remark. Default is `NULL`, so the diagram's title will not be modified when cases are weighted. Use a string as parameter, e.g.: `weightByTitleString=" (weighted)"`.

`interactionVar` An interaction variable which can be used for box plots. Divides each category indicated by `varGroup` into the factors of `interactionVar`, so that each category of `varGroup` is subgrouped into `interactionVar`'s categories. Only applies when parameter `type` is `box` or `violin` (resp. their alternative strings like `"boxplot"`, `"boxplots"` or `"v"`).

`barPosition` Indicates whether bars should be positioned side-by-side (default, or use `"dodge"` as parameter) or stacked (use `"stack"` as parameter). If `type` is `"histogram"`, you can use either `"dodge"` (default value), which displays the bars side-by-side, or `"identity"`, which results in overlaying bars. In the latter case, it's recommended to adjust the `barAlpha` value.

`type` The plot type. May be one of the following:

- `"b"`, `"bar"`, `"bars"` (default) for bar charts
- `"l"`, `"line"`, `"lines"` for line diagram
- `"d"`, `"dot"`, `"dots"` for dot plots
- `"h"`, `"hist"`, `"histogram"` for grouped histograms
- `"box"`, `"boxplot"`, `"boxplots"` for box plots
- `"v"`, `"violin"` for violin box plots

`dotSize` Size of dots. Applies only when `type` is set to `"dots"`.

`hideLegend` Indicates whether legend (guide) should be shown or not.

`maxYlim` Indicates how to calculate the maximum limit of the y-axis. If `TRUE`, the upper y-limit corresponds to the amount of cases, i.e. y-axis for each plot of a data base are the same. If `FALSE` (default), the maximum y-axis depends on the highest count of a variable's answer category. In this case, the y-axis breaks may change, depending on the variable.

`upperYlim` Uses a pre-defined upper limit for the y-axis. Overrides the `maxYlim` parameter.

`useFacetGrid` `TRUE` when bar charts should be plotted as facet grids instead of integrated single bar charts. Ideal for larger amount of groups. This parameter wraps a single panel into `varGroup` amount of panels, i.e. each group is represented within a new panel.

`title` Title of the diagram, plotted above the whole diagram panel. Use `"auto"` to automatically detect variable names that will be used as title (see `sji.setVariableLabels`) for details).

`titleSize` The size of the plot title. Default is 1.3.

`titleColor` The color of the plot title. Default is `"black"`.

`legendTitle` Title of the diagram's legend.

`axisLabels.x` Labels for the x-axis breaks. Passed as vector of strings. *Note:* This parameter is not necessary when data was either imported with `sji.SPSS` or has

named factor levels (see examples below). Else, specifiy parameter like this: `axisLabels.x=c("Label1", "Label2", "Label3")`. Note: If you use the `sji.SPSS` function and the `sji.getValueLabels` function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.

`interactionVarLabels`
: Labels for the x-axis breaks when having interaction variables included. These labels replace the `axisLabels.x`. Only applies, when using box or violin plots (i.e. `"type"` is `"box"` or `"violin"`) and `interactionVar` is not `NULL`. Example: See `axisLabels.x`.

`legendLabels`
: Labels for the guide/legend. Example: See `axisLabels.x`.

`axisLabelSize`
: The size of axis labels of both x and y axis. Default is 1.1, recommended values range between 0.5 and 3.0

`valueLabelSize`
: The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8

`axisLabelAngle.x`
: Angle for axis-labels.

`breakTitleAt`
: Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

`breakLabelsAt`
: Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted.

`breakLegendTitleAt`
: Wordwrap for diagram legend title. Determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

`breakLegendLabelsAt`
: Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

`gridBreaksAt`
: Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed.

`barWidth`
: Width of bars. Recommended values for this parameter are from 0.4 to 1.5

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

`innerBoxPlotDotSize`
: Size of mean dot insie a violin plot. Applies only when `type` is set to `"violin"`.

`barSpace`
: Spacing between bars. Default value is 0.1. If 0 is used, the grouped bars are sticked together and have no space in between. Recommended values for this parameter are from 0 to 0.5

`barColor`
: User defined color for bars.

  - If not specified (`NULL`), a default red-green-yellow color palette will be used for the bar charts.
  - If barColor is `"gs"`, a greyscale will be used.
  - If barColor is `"bw"`, a monochrome white filling will be used.

- If barColor is "brewer", use the colorPalette parameter to specify a palette of the http://colorbrewer2.org.

Else specify your own color values as vector (e.g. barColor=c("#f00000", "#00ff00", "#0080ff")).

colorPalette — If barColor is "brewer", specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here.

barAlpha — Specify the transparancy (alpha value) of bars.

lineType — The linetype when using line diagrams. Only applies, when parameter type is set to "lines".

lineSize — The size of lines in a line diagram. Only applies, when parameter type is set to "lines".

lineAlpha — The alpha value of lines in a line diagram. Only applies, when parameter type is set to "lines".

smoothLines — Prints a smooth line curve. Only applies, when parameter type is set to "lines".

axisLabelColor — User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

borderColor — User defined color of whole diagram border (panel border).

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

barOutline — If TRUE, each bar gets a colored outline. Default is FALSE.

barOutlineSize — The size of the bar outlines. Only applies if barOutline is TRUE. Default is 0.2

barOutlineColor — The color of the bar outline. Only applies, if barOutline is set to TRUE.

majorGridColor — Specifies the color of the major grid lines of the diagram background.

minorGridColor — Specifies the color of the minor grid lines of the diagram background.

hideGrid.x — If TRUE, the x-axis-gridlines are hidden. Default is FALSE.

hideGrid.y — If TRUE, the y-axis-gridlines are hidden. Default is FALSE.

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

showValueLabels — Whether counts and percentage values should be plotted to each bar. Default is TRUE.

showCountValues — If TRUE (default), count values are be plotted to each bar. If FALSE, count values are removed.

showPercentageValues — If TRUE (default), percentage values are be plotted to each bar, if FALSE, percentage-values are removed.

showAxisLabels.x — Whether x axis labels (category names) should be shown or not.

showAxisLabels.y — Whether y axis labels (count values) should be shown or not.

showTickMarks — Whether tick marks of axes should be shown or not.

showPlotAnnotation
: If TRUE, the groups of dots in a dot-plot are highlighted with a shaded rectangle.

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

showMeanValue
: If TRUE (default value), the mean value is printed to the vertical line that indicates the mean value of the count variables. Only applies to histogram-charts.

showStandardDeviation
: If TRUE, the standard deviation is annotated as shaded rectangle around the mean intercept line. Only applies to histogram-charts. The shaded rectangles have borders in the group colors, so it's easier to see which shaded area belongs to which mean value resp. group

showTableSummary
: If TRUE (default), a summary of the cross tabulation with N, Chi-square (see chisq.test), df, Cramer's V or Phi-value and p-value is printed to the upper right corner of the diagram. 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-square test. If the table's matrix is larger than 2x2, Fisher's excact test with Monte Carlo simulation is computed. Only applies to bar-charts or dot-plots, i.e. when parameter type is either "bars" or "dots".

summaryLabelColor
: The color of the table summary labels.

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

tableSummaryPos
: Position of the model summary which is printed when showTableSummary is TRUE. Default is "r", i.e. it's printed to the upper right corner. Use "l" for upper left corner.

meanInterceptLineType
: The linetype of the mean intercept line. Only applies to histogram-charts and when showMeanIntercept is TRUE.

meanInterceptLineSize
: The size of the mean intercept line. Only applies to histogram-charts and when showMeanIntercept is TRUE.

valueLabelColor
: The color of the value labels (numbers) inside the diagram.

axisTitle.x
: A label for the x axis. Useful when plotting histograms with metric scales where no category labels are assigned to the x axis. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

axisTitle.y
: A label for the y axis. Useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

axisTitleColor
: The color of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

axisTitleSize
: The size of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

autoGroupAt
: A value indicating at which length of unique values of varCount the variable is automatically grouped into smaller units (see sju.groupVar). If varCount has large numbers of unique values, too many bars for the graph have to be plotted. Hence it's recommended to group such variables. For example, if autoGroupAt is 50, i.e. if varCount has 50 and more unique values it will be grouped using sju.groupVar with groupsize="auto" parameter. By default, the maximum group count is 30. However, if autoGroupAt is less than 30, autoGroupAt groups are built. Default value for autoGroupAt is NULL, i.e. auto-grouping is off.

startAxisAt
: Determines the first value on the x-axis. By default, this value is set to "auto", i.e. the value range on the x axis starts with the lowest value of varCount. If you set startAxisAt to 1, you may have zero counts if the lowest value of varCount is larger than 1 and hence no bars plotted for these values in such cases.

theme
: Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.
  - Use "bw" for a white background with gray grids
  - "classic" for a classic theme (black border, no grids)
  - "minimal" for a minimalistic theme (no border,gray grids)
  - "none" for no borders, grids and ticks or
  - "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

  See http://rpubs.com/sjPlot/custplot for details and examples.

legendPos
: The position of the legend, if a legend is drawn. 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.

legendSize
: The text size of the legend. Default is 1. Relative size, so recommended values are from 0.3 to 2.5

legendBorderColor
: Color of the legend's border. Default is "white", so no visible border is drawn.

legendBackColor
: Fill color of the legend's background. Default is "white", so no visible background is drawn.

flipCoordinates
: If TRUE, the x and y axis are swapped.

labelPos
: If flipCoordinates is TRUE, use this parameter to specify value label position. Can be either "inside" or "outside" (default). You may specify initial letter only. If flipCoordinates is FALSE, this parameter will be ignored.

na.rm
: If TRUE, missings are not included in the frequency calculation and diagram plot.

printPlot
: 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).

References

- http://rpubs.com/sjPlot/sjpgrpfrq
- http://strengejacke.wordpress.com/sjplot-r-package/

Examples

```
# histrogram with EUROFAMCARE sample dataset
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
sjp.grpfrq(efc$e17age,
           efc$e16sex,
           title=efc.var['e17age'],
           legendTitle=efc.var['e16sex'],
           type="hist",
           showValueLabels=FALSE,
           showMeanIntercept=TRUE)

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

# ------------------------------------------------
# auto-detection of value labels and variable names
# ------------------------------------------------
efc.var <- sji.getVariableLabels(efc)
efc <- sji.setVariableLabels(efc, efc.var)
# grouped bars using necessary y-limit
sjp.grpfrq(efc$e42dep, efc$e16sex, title="auto")

# grouped bars using the maximum y-limit
sjp.grpfrq(efc$e42dep,
           efc$e16sex,
           title=efc.var['e42dep'],
           axisLabels.x=efc.val[['e42dep']], # not needed for SPSS-data sets
           legendTitle=efc.var['e16sex'],
           legendLabels=efc.val[['e16sex']], # not needed for SPSS-data sets
           maxYlim=TRUE)

# box plots with interaction variable
sjp.grpfrq(efc$e17age,
           efc$e42dep,
           interactionVar=efc$e16sex,
         title=paste(efc.var['e17age'], "by", efc.var['e42dep'], "and", efc.var['e16sex']),
           axisLabels.x=efc.val[['e17age']],
           interactionVarLabels=efc.val[['e16sex']],
           legendTitle=efc.var['e42dep'],
           legendLabels=efc.val[['e42dep']],
           type="box")

# Grouped bar plot ranging from 1 to 28 (though scale starts with 7)
sjp.grpfrq(efc$neg_c_7, efc$e42dep, showValueLabels=FALSE, startAxisAt=1)
# Same grouped bar plot ranging from 7 to 28
```

```
sjp.grpfrq(efc$neg_c_7, efc$e42dep, showValueLabels=FALSE)
```

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

Description

Plot likert scales as centered stacked bars. "Neutral" categories (odd-numbered categories) will be removed from the plot.

Usage

```
sjp.likert(items, legendLabels = NULL, orderBy = NULL,
  reverseOrder = FALSE, dropLevels = NULL, weightBy = NULL,
  weightByTitleString = NULL, hideLegend = FALSE, title = NULL,
  titleSize = 1.3, titleColor = "black", legendTitle = NULL,
  includeN = TRUE, axisLabels.y = NULL, axisLabelSize = 1.1,
  axisLabelAngle.x = 0, axisLabelColor = "gray30", valueLabelSize = 4,
  valueLabelColor = "black", breakTitleAt = 50, breakLabelsAt = 30,
  breakLegendTitleAt = 30, breakLegendLabelsAt = 28, gridRange = 1,
  gridBreaksAt = 0.2, expand.grid = TRUE, barWidth = 0.5,
  barColor = NULL, colorPalette = "GnBu", barAlpha = 1,
  borderColor = NULL, axisColor = NULL, barOutline = FALSE,
  barOutlineColor = "black", majorGridColor = NULL, minorGridColor = NULL,
  hideGrid.x = FALSE, hideGrid.y = FALSE, axisTitle.x = NULL,
  axisTitle.y = NULL, axisTitleColor = "black", axisTitleSize = 1.3,
  theme = NULL, showTickMarks = FALSE, showValueLabels = TRUE,
  jitterValueLabels = FALSE, showItemLabels = TRUE,
  showSeparatorLine = FALSE, separatorLineColor = "grey80",
  separatorLineSize = 0.3, legendPos = "right", legendSize = 1,
  legendBorderColor = "white", legendBackColor = "white",
  flipCoordinates = TRUE, printPlot = TRUE)
```

Arguments

`items`	A data frame with each column representing one likert-item.
`legendLabels`	A list or vector of strings that indicate the likert-scale-categories and which appear as legend text.
`orderBy`	Indicates whether the `items` should be ordered by total sum of positive or negative answers. Use `"pos"` to order descending by sum of positive answers, `"neg"` for sorting descending negative answers or `NULL` (default) for no sorting.
`reverseOrder`	If `TRUE`, the item order (positive/negative) are reversed. Default is `FALSE`.
`dropLevels`	Indicates specific factor levels that should be dropped from the items before the likert scale is plotted. Default is `NULL`, hence all factor levels are included. Exampe to drop first factor level: `dropLevels=c(1)`.
`weightBy`	A weight factor that will be applied to weight all cases from `items`.

weightByTitleString
: If a weight factor is supplied via the parameter weightBy, the diagram's title may indicate this with a remark. Default is NULL, so the diagram's title will not be modified when cases are weighted. Use a string as parameter, e.g.: weightByTitleString=" (weighted)"

hideLegend
: Indicates whether legend (guide) should be shown or not.

title
: Title of the diagram, plotted above the whole diagram panel.

titleSize
: The size of the plot title. Default is 1.3.

titleColor
: The color of the plot title. Default is "black".

legendTitle
: Title of the diagram's legend.

includeN
: If TRUE (default), the N of each item is included into axis labels.

axisLabels.y
: Labels for the y-axis (the labels of the items). These parameters must be passed as list! Example: axisLabels.y=list(c("Q1", "Q2", "Q3")) Axis labels will automatically be detected, when they have a "variable.lable" attribute (see sji.setVariableLabels) for details).

axisLabelSize
: The size of category labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0

axisLabelAngle.x
: Angle for axis-labels.

axisLabelColor
: User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

valueLabelSize
: The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8

valueLabelColor
: The color of value labels in the diagram. Default is black.

breakTitleAt
: Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

breakLabelsAt
: Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted.

breakLegendTitleAt
: Wordwrap for diagram legend title. Determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

breakLegendLabelsAt
: Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

gridRange
: Sets the limit of the x-axis-range. Default is 1, so the x-scale ranges from zero to 100 percent on both sides from the center. Valid values range from 0 (0 percent) to 1 (100 percent).

gridBreaksAt
: Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Valid values range from 0 to 1.

expand.grid
: If TRUE (default), the diagram has margins, i.e. the y-axis is not exceeded to the diagram's boundaries.

barWidth
: Width of bars. Recommended values for this parameter are from 0.4 to 1.5

barColor — User defined color for bars. If not specified (NULL), a default red-green color palette for four(!) categories will be used for the bar charts. You can use predefined color-sets that are independent from the amount of categories:

- If barColor is "brown", a brown-marine-palette will be used.
- If barColor is "violet", a violet-green palette will be used.
- If barColor is "pink", a pink-green palette will be used.
- If barColor is "brewer", use the colorPalette parameter to specify a palette of the color brewer.

Else specify your own color values as vector (e.g. barColor=c("darkred", "red", "green", "darkg

colorPalette — If barColor is "brewer", specify a color palette from the color brewer here. All color brewer palettes supported by ggplot are accepted here.

barAlpha — Specify the transparancy (alpha value) of bars.

borderColor — User defined color of whole diagram border (panel border).

barOutline — If TRUE, each bar gets a colored outline. Default is FALSE.

barOutlineColor — The color of the bar outline. Only applies, if barOutline is set to TRUE.

majorGridColor — Specifies the color of the major grid lines of the diagram background.

minorGridColor — Specifies the color of the minor grid lines of the diagram background.

hideGrid.x — If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y — If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

axisTitle.x — A label for the x axis. Useful when plotting histograms with metric scales where no category labels are assigned to the x axis.

axisTitle.y — A label for the y axis. Useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

axisTitleColor — The color of the x and y axis labels. refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

axisTitleSize — The size of the x and y axis labels. refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

showValueLabels — Whether counts and percentage values should be plotted to each bar

jitterValueLabels — If TRUE, the value labels on the bars will be "jittered", i.e. they have alternating vertical positions to avoid overlapping of labels in case bars are very short. Default is FALSE.

showItemLabels — Whether x axis text (category names) should be shown or not

showTickMarks — Whether tick marks of axes should be shown or not

showSeparatorLine — If TRUE, a line is drawn to visually "separate" each bar in the diagram.

separatorLineColor — The color of the separator line. Only applies, if showSeparatorLine is TRUE

separatorLineSize
: The size of the separator line. only applies, if showSeparatorLine is TRUE

legendPos
: The position of the legend. Default is "right". Use one of the following values: "right", "left", "bottom", "top".

legendSize
: The size of the legend.

legendBorderColor
: The border color of the legend.

legendBackColor
: The background color of the legend.

theme
: Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.
  - Use "bw" for a white background with gray grids
  - "classic" for a classic theme (black border, no grids)
  - "minimal" for a minimalistic theme (no border,gray grids)
  - "none" for no borders, grids and ticks or
  - "themr" if you are using the ggthemr package

  See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates
: If TRUE, the x and y axis are swapped.

printPlot
: 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

Since package version 1.3, the parameter legendLabels, which represent the value labels, are retrieved automatically if a) the variables in items come from a data frame that was imported with the sji.SPSS function (because then value labels are attached as attributes to the data) or b) when the variables are factors with named factor levels (e.g., see column group in dataset PlantGrowth). However, you still can use own parameters as axis- and legendlabels.

Transformation of data and ggplot-code taken from http://statisfactions.com/2012/improved-net-stacked-distri

References

http://strengejacke.wordpress.com/sjplot-r-package/

http://strengejacke.wordpress.com/2013/07/17/plotting-likert-scales-net-stacked-distributions-with-

See Also

sjp.stackfrq

Examples

```
# prepare data for dichotomous likert scale, 5 items
likert_2 <- data.frame(as.factor(sample(1:2, 500, replace=TRUE, prob=c(0.3,0.7))),
                       as.factor(sample(1:2, 500, replace=TRUE, prob=c(0.6,0.4))),
                       as.factor(sample(1:2, 500, replace=TRUE, prob=c(0.25,0.75))),
                       as.factor(sample(1:2, 500, replace=TRUE, prob=c(0.9,0.1))),
                       as.factor(sample(1:2, 500, replace=TRUE, prob=c(0.35,0.65))))
# create labels
levels_2 <- list(c("Disagree", "Agree"))

# 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 <- list(c("Strongly disagree", "Disagree", "Agree", "Strongly Agree"))

# prepare data for 6-category likert scale, 5 items
likert_6 <- data.frame(
             as.factor(sample(1:6, 500, replace=TRUE, prob=c(0.2,0.1,0.1,0.3,0.2,0.1))),
            as.factor(sample(1:6, 500, replace=TRUE, prob=c(0.15,0.15,0.3,0.1,0.1,0.2))),
            as.factor(sample(1:6, 500, replace=TRUE, prob=c(0.2,0.25,0.05,0.2,0.2,0.2))),
             as.factor(sample(1:6, 500, replace=TRUE, prob=c(0.2,0.1,0.1,0.4,0.1,0.1))),
            as.factor(sample(1:6, 500, replace=TRUE, prob=c(0.1,0.4,0.1,0.3,0.05,0.15))))
# create labels
levels_6 <- list(c("Very strongly disagree", "Strongly disagree", "Disagree",
                   "Agree", "Strongly Agree", "Very strongly agree"))

# create item labels
items <- list(c("Q1", "Q2", "Q3", "Q4", "Q5"))

# plot dichotomous likert scale, ordered by "negative" values
sjp.likert(likert_2, legendLabels=levels_2, axisLabels.y=items, orderBy="neg")

# plot 4-category-likert-scale, no order
sjp.likert(likert_4, legendLabels=levels_4, axisLabels.y=items)

# plot 4-category-likert-scale, ordered by positive values,
# in brown color scale and with jittered value labels
sjp.likert(likert_6, legendLabels=levels_6, barColor="brown",
           axisLabels.y=items, orderBy="pos", jitterValueLabels=TRUE)
```

`sjp.lm` *Plot beta coefficients of lm*

Description

Plot beta coefficients of linear regressions with confidence intervalls as dot plot (forest plot). Additionally, the standardized beta values are plotted as red dots.

Usage

```
sjp.lm(fit, sort = "beta", title = NULL, titleSize = 1.3,
  titleColor = "black", axisLabels.y = NULL, showAxisLabels.y = TRUE,
  axisLabelSize = 1.1, axisLabelColor = "gray30",
  axisTitle.x = "Estimates", axisTitleSize = 1.4,
  axisTitleColor = c("#444444"), axisLimits = NULL,
  valueLabelColor = "grey20", valueLabelColorNS = "grey50",
  valueLabelSize = 4.5, valueLabelAlpha = 0.8, axisLabelAngle.x = 0,
  axisLabelAngle.y = 0, errorBarColor = "#3366a0", errorBarWidth = 0,
  errorBarSize = 0.8, pointColor = "#3366a0", pointSize = 3,
  pointColorStdBeta = "#cc5533", pointSizeStdBeta = 3,
  stdBetaLineType = 2, stdBetaLineAlpha = 0.3, interceptLineType = 2,
  interceptLineColor = "grey70", breakTitleAt = 50, breakLabelsAt = 12,
  gridBreaksAt = NULL, borderColor = NULL, axisColor = NULL,
  theme = NULL, flipCoordinates = TRUE, majorGridColor = NULL,
  minorGridColor = NULL, hideGrid.x = FALSE, hideGrid.y = FALSE,
  showTickMarks = TRUE, showValueLabels = TRUE, labelDigits = 2,
  showPValueLabels = TRUE, showModelSummary = TRUE,
  showStandardBeta = FALSE, showStandardBetaLine = FALSE,
  printPlot = TRUE)
```

Arguments

`fit`	The model of the linear regression (lm-Object).
`title`	Diagram's title as string. Example: `title=c("my title")`
`titleSize`	The size of the plot title. Default is 1.3.
`titleColor`	The color of the plot title. Default is `"black"`.
`sort`	Determines whether the predictors are sorted by beta-values (default, or use `"beta"` as parameter) or by standardized beta values (use `"std"`).
`axisLabels.y`	Labels of the predictor variables (independent vars) that are used for labelling the axis. Passed as vector of strings. Example: `axisLabels.y=c("Label1", "Label2", "Label3")`. Note: If you use the `sji.SPSS` function and the `sji.getValueLabels` function, you receive a list object with label string. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.
`showAxisLabels.y`	Whether x axis text (category names, predictor labels) should be shown (use `TRUE`) or not. Default is `TRUE`
`axisLabelSize`	The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8.
`axisLabelColor`	The color of the category labels (predictor labels). Default is a dark grey (grey30).
`axisTitle.x`	A label for the x axis. Default is `"Estimates"`.
`axisTitleColor`	The color of the x axis label. Default is a dark grey.
`axisTitleSize`	The size of the x axis label. Default is 1.4.

`axisLimits` Defines the range of the axis where the beta coefficients and their confidence intervalls are drawn. By default, the limits range from the lowest confidence interval to the highest one, so the diagram has maximum zoom. Use your own values as 2-value-vector, for instance: `limits=c(-0.8,0.8)`.

`valueLabelColor`
Colour of the values (significant beta coefficients) inside the diagrams. Only applies, when parameter `showValueLabels` is set to `TRUE`. Use any valid colour value, e.g. `valueLabelColor="grey50"` or `valueLabelColor=c("#cc3366")`.

`valueLabelColorNS`
Colour of the non significant values (non significant beta coefficients) inside the diagrams. Only applies, when parameter `showValueLabels` is set to `TRUE`. Use any valid colour value, e.g. `valueLabelColor="grey50"` or `valueLabelColor=c("#cc3366")`.

`valueLabelSize` Size of the value labels. Default is 4.5. Recommended Values range from 2 to 8

`valueLabelAlpha`
The alpha level (transparancy) of the value labels. Default is 0.8, use any value from 0 to 1.

`axisLabelAngle.x`
Angle for axis-labels where the estimates are printed. Note that due to the coordinate flip, the acutal y-axis with estimates labels are appearing on the x-axis.

`axisLabelAngle.y`
Angle for axis-labels, passed as numeric value.

`errorBarColor` The color of the error bars that indicate the confidence intervalls of the beta-coefficients

`errorBarWidth` The width of the error bar ends. Default is 0

`errorBarSize` The size of the error bar. Default is 0.8

`pointColor` The colour of the points that indicate the beta-value.

`pointSize` The size of the points that indicate the beta-value. Default is 3.

`pointColorStdBeta`
The colour of the points that indicate the standardized beta-value.

`pointSizeStdBeta`
The size of the points that indicate the standardized beta-value. Default is 3.

`stdBetaLineType`
The standardized beta-value dots are connected by a thin line for a better overview. With this parameter you can specify the line type.

`stdBetaLineAlpha`
The alpha-value for the line that connects the standardized beta-value dots.

`interceptLineType`
The linetype of the intercept line (zero point). Default is 2 (dashed line).

`interceptLineColor`
The color of the intercept line. Default value is `"grey70"`.

`breakTitleAt` Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title

`breakLabelsAt` Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted

gridBreaksAt Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is NULL, so pretty gridbeaks will be used.

borderColor User defined color of whole diagram border (panel border).

axisColor User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

theme Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the pointColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates
If TRUE (default), predictors are plotted on the left y-axis and estimate values are plotted on the x-axis.

majorGridColor Specifies the color of the major grid lines of the diagram background.

minorGridColor Specifies the color of the minor grid lines of the diagram background.

hideGrid.x If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

showTickMarks Whether tick marks of axes should be shown or not

showValueLabels
Whether the beta and standardized beta values should be plotted to each dot or not.

labelDigits The amount of digits for rounding the estimations (see showValueLabels). Default is 2, i.e. estimators have 2 digits after decimal point.

showPValueLabels
Whether the significance levels of each coefficient should be appended to values or not

showModelSummary
If TRUE (default), a summary of the regression model with Intercept, R-square, F-Test and AIC-value is printed to the lower right corner of the diagram.

showStandardBeta
Whether or not the dots for the standardized beta values should be plotted to the diagram.

showStandardBetaLine
Whether or not the connecting line for the standardized beta values should be plotted to the diagram. Default is FALSE.

printPlot 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

Based on an an idea from surefoss: http://www.surefoss.org/dataanalysis/plotting-odds-ratios-aka-a-forrest

References

- http://rpubs.com/sjPlot/sjplm
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.lm.ma
sjp.reglin
sjp.lm.int
sjp.scatter
sjs.betaCoef

Examples

```
# fit linear model
fit <- lm(airquality$Ozone ~ airquality$Wind + airquality$Temp + airquality$Solar.R)

# plot estimates with CI and standardized beta-values
sjp.lm(fit, gridBreaksAt=2)

# plot estimates with CI without standardized beta-values
# and with narrower tick marks (because "gridBreaksAt" was not specified)
sjp.lm(fit, showStandardBeta=FALSE)
```

`sjp.lm.int` *Plot interaction terms (moderation) of linear models*

Description

Plot regression curves of significant interaction terms (moderation) in linear models (lm). Note that beside interaction terms, also the single predictors of each interaction 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.lm.int(fit, smooth = "none", diff = FALSE, moderatorValues = "minmax",
  swapPredictors = FALSE, plevel = 0.05, title = NULL, titleSize = 1.3,
  titleColor = "black", fillColor = "grey", fillAlpha = 0.4,
  lowerBoundColor = "#3366cc", upperBoundColor = "#cc3300",
  meanColor = "#00cc33", lineColor = "#33cc66", axisTitle.x = NULL,
  axisTitle.y = NULL, axisLabelColor = "gray30", axisLabelSize = 1.1,
  axisTitleColor = "black", axisTitleSize = 1.3, legendLabels = NULL,
  legendLabelSize = 0.9, legendLabelColor = "black",
  showValueLabels = FALSE, valueLabelSize = 4, valueLabelColor = "black",
  valueLabelAlpha = 0.8, breakTitleAt = 50, breakLegendLabelsAt = 20,
  breakAnnotationLabelsAt = 50, axisLimits.y = NULL, gridBreaksAt = NULL,
  theme = NULL, showTickMarks = TRUE, showInterceptLines = FALSE,
  showInterceptLabels = TRUE, interceptLineColor = "#3366cc",
  estLineColor = "#cc3300", lineLabelSize = 3.7, lineLabelColor = "black",
  lineLabelString = "(no interaction)", borderColor = NULL,
  axisColor = NULL, majorGridColor = NULL, minorGridColor = NULL,
  hideGrid.x = FALSE, hideGrid.y = FALSE, printPlot = TRUE)
```

Arguments

fit	the fitted linear model (lm) object, including interaction terms
smooth	smoothes the regression line in case it is not linear. Default is "none", so no smoothing is applied. Use "loess" for loess-smoothing or "lm" to force linear regression lines.
diff	if FALSE (default), the minimum and maximum interaction effects of predictor 2 on predictor 1 are shown (one line each). if TRUE, only the difference between minimum and maximum interaction effect is shown (single line)
moderatorValues	indicates which values of the moderator variable should be used when plotting the effects of the independent variable on the dependent variable. By default, "minmax" is used, i.e. the minimum and maximum values (lower and upper bounds) of the moderator are used to plot the interaction between independent variable and moderator. Use "meansd" to use 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 http://www.theanalysisfactor.com/3-tips-interpreting-moderation/).
swapPredictors	if TRUE, the predictor with less unique values is printed along the x-axis. Default is FALSE, so the predictor with more unique values is printed along the x-axis.
plevel	Indicates at which p-value an interaction term is considered as significant. Default is 0.05 (5 percent).
title	a default title used for the plots. Default value is NULL, which means that each plot's title includes the dependent variable as well as the names of the interaction terms.

titleSize The size of the plot title. Default is 1.3.

titleColor The color of the plot title. Default is "black".

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

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

lowerBoundColor
the color of the line indicating the lower bound of the interaction term (moderator value). Default value is "#3366cc" (blue-like)

upperBoundColor
the color of the line indicating the upper bound of the interaction term (moderator value). Default value is "#cc3300" (red-like)

meanColor the color of the line indicating the mean value of the interaction term (moderator value). Only applies, when moderatorValues is "meansd".

lineColor the color of the line indicating the upper difference between lower and upper bound of interaction terms. Only applies if diff is TRUE. Default value is "#33cc66" (green-like)

axisTitle.x a default title used for the x-axis. Default value is NULL, which means that each plot's x-axis uses the predictor's name as title.

axisTitle.y a default title used for the y-axis. Default value is NULL, which means that each plot's y-axis uses the dependent variable's name as title.

axisLabelColor the color value for the axis labels at the tick marks. Default value is "darkgray".

axisLabelSize The size of axis labels. Default is 1.1, recommended values range between 0.5 and 3.0

axisTitleColor the color value for the axis titles (both x and y). Default value is "black".

axisTitleSize The size of axis titles (both x and y). Default is 1.3, recommended values range between 0.5 and 3.0

legendLabels Labels for the guide/legend. Default is NULL, so the name of the predictor with min/max-effect is used as legend label.

legendLabelSize
The size of legend labels. Default is 0.9, recommended values range between 0.5 and 3.0

legendLabelColor
user defined color for legend labels. If not specified, black will be used for the labels

showValueLabels
if TRUE, value labels are plotted along the lines. Default is FALSE.

valueLabelSize size of the value labels. Default is 4. Recommended Values range from 2 to 8

valueLabelColor
colour of the values inside the diagrams. Only applies, when parameter showValueLabels is set to TRUE. Use any valid colour value, e.g. valueLabelColor="grey50" or valueLabelColor=c("#cc3366"). Default is "black".

valueLabelAlpha
: the alpha level (transparancy) of the value labels. Default is 0.8, use any value from 0 to 1.

breakTitleAt
: Wordwrap for diagram's title. Determines how many chars of the title are displayed in one line and when a line break is inserted. Default is 50.

breakLegendLabelsAt
: Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted. Default is 20.

breakAnnotationLabelsAt
: Wordwrap for diagram annotation labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted. Default is 50. Only applies if showInterceptLine is TRUE.

axisLimits.y
: A vector with two values, defining the lower and upper limit from the y-axis. By default, this value is NULL, i.e. axis limits will be calculated upon the range of y-values.

gridBreaksAt
: Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is NULL.

theme
: Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

  - Use "bw" for a white background with gray grids
  - "classic" for a classic theme (black border, no grids)
  - "minimal" for a minimalistic theme (no border,gray grids)
  - "none" for no borders, grids and ticks or
  - "themr" if you are using the ggthemr package

  See http://rpubs.com/sjPlot/custplot for details and examples.

showTickMarks
: Whether tick marks of axes should be shown or not

showInterceptLines
: If TRUE, the intercept and the estimate of the predictor (reference category of predictor in case interaction is not present) are plotted.

showInterceptLabels
: If TRUE (default), the intercept lines are labelled. Only applies if showInterceptLines is TRUE.

interceptLineColor
: The line color of the model's intercept line. Only applies, if showInterceptLines is TRUE.

estLineColor
: The line color of the model's predictor's estimate line. Only applies, if showInterceptLines is TRUE.

lineLabelSize
: The size of the intercept line annotations inside the plot. Only applies if showInterceptLines is TRUE. Default is 3.7.

lineLabelColor
: The color of the intercept line annotations inside the plot. Only applies if showInterceptLines is TRUE. Default is "black".

lineLabelString
: Default string for the intercept lines that is appended to the predictor variable name. By default, this string is "(no interaction)".

borderColor	user defined color of whole diagram border (panel border)
axisColor	user defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).
majorGridColor	specifies the color of the major grid lines of the diagram background
minorGridColor	specifies the color of the minor grid lines of the diagram background
hideGrid.x	If TRUE, the x-axis-gridlines are hidden. Default if FALSE.
hideGrid.y	If TRUE, the y-axis-gridlines are hidden. Default if FALSE.
printPlot	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-objects with the complete plot-list (plot.list) as well as the data frame that were used for setting up the ggplot-objects (df.list).

Note

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

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2013/10/31/visual-interpretation-of-interaction-terms-in-lin
- http://www.theanalysisfactor.com/interpreting-interactions-in-regression/
- http://www.theanalysisfactor.com/clarifications-on-interpreting-interactions-in-regression/
- http://www.theanalysisfactor.com/3-tips-interpreting-moderation/
- Aiken and West (1991). Multiple Regression: Testing and Interpreting Interactions.

See Also

sjp.lm
sjp.emm.int
sjp.reglin
sjp.lm.ma

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.
fit <- lm(weight ~ Time * Diet, data=ChickWeight, x=TRUE)

# show summary to see significant interactions
summary(fit)
```

```
# plot regression line of interaction terms
sjp.lm.int(fit)
# plot regression line of interaction terms, including value labels
sjp.lm.int(fit, showValueLabels=TRUE)


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

# plot interactions
sjp.lm.int(fit)
# plot interactions, including those with p-value up to 0.1
sjp.lm.int(fit, plevel=0.1, showInterceptLines=TRUE)
```

`sjp.lm.ma` *Plot model assumptions of lm's*

Description

Plots model assumptions of linear models to verify if linear regression is applicable

Usage

```
sjp.lm.ma(linreg, showOriginalModelOnly = TRUE, completeDiagnostic = FALSE)
```

Arguments

`linreg`	a fitted lm-model
`showOriginalModelOnly`	if `TRUE` (default), only the model assumptions of the fitted model `linreg` are plotted. if `FALSE`, the model assumptions of an updated model where outliers are automatically excluded are also plotted.
`completeDiagnostic`	if `TRUE`, additional tests are performed. Default is `FALSE`

Value

an updated fitted linear model where outliers are dropped out.

References

http://rpubs.com/sjPlot/sjplm

See Also

sjp.lm
sjp.reglin
sjp.lm.int

Examples

```
## Not run:
# fit linear model
fit <- lm(airquality$Ozone ~ airquality$Wind + airquality$Temp + airquality$Solar.R)
fit.updated <- sjp.lm.ma(fit)
## End(Not run)
```

---

sjp.lm1 *Plot regression line of fitted lm*

Description

Plot a regression line with confidence interval for a fitted model with only one predictor (i.e. `lm(y~x)`). This function may plot two lines: The resulting linear regression line including confidence interval (in blue) by default, and a loess-smoothed line without confidence interval (in red) if parameter `showLoess` is `TRUE`. The better the linear relationship of predictor and response is, the more both lines should overlap (i.e. the red loess-smoothed line is almost linear).

Furthermore, a scatter plot of response and predictor values is plotted.

Usage

```
sjp.lm1(fit, data, title = NULL, titleSize = 1.3, titleColor = "black",
  breakTitleAt = 50, axisLabel.x = NULL, axisLabel.y = NULL,
  breakLabelsAt = 12, axisLabelSize = 1.1, axisLabelColor = "gray30",
  axisTitleSize = 1.4, axisTitleColor = c("#444444"), lineColor = "blue",
  modsumLabelColor = "grey20", modsumLabelSize = 4.5, showCI = TRUE,
  ciLevel = 0.95, pointAlpha = 0.2, pointColor = "black",
  showScatterPlot = TRUE, showLoess = FALSE, loessLineColor = "red",
  showLoessCI = FALSE, loessCiLevel = 0.95, showModelSummary = TRUE,
  borderColor = NULL, axisColor = NULL, theme = NULL,
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, showTickMarks = TRUE, printPlot = TRUE)
```

Arguments

fit	The model of the linear regression (lm-Object).
data	The data/dataset/dataframe used to fit the model
title	Diagram's title as string. Example: title=c("my title")
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
breakTitleAt	Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title
axisLabel.x	Labels of the predictor (independent variable) that is used for labelling the axis. Passed as string. Example: axisLabel.x=c("My Predictor Var"). Note: If you use the sji.SPSS function and the sji.getVariableLabels function, you receive a character vector with variable label strings. You can use it like so: axisLabel.x=sji.getVariableLabels(efc)['quol_5']
axisLabel.y	Labels of the response (dependent variable) that is used for labelling the axis. Passed as string. Example: axisLabel.y=c("My Depdendent Var"). Note: If you use the sji.SPSS function and the sji.getVariableLabels function, you receive a character vector with variable label strings. You can use it like so: axisLabel.y=sji.getVariableLabels(efc)['neg_c_7']
breakLabelsAt	Wordwrap for axis labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted
axisLabelSize	The size of axis tick marks Default is 1.1.
axisLabelColor	The color of the axis tick marks. Default is a dark grey (grey30).
axisTitleColor	The color of the axis labels (response and predictor label). Default is a dark grey.
axisTitleSize	The size of the axis label (response and predictor label). Default is 1.4.
lineColor	The color of the regression line. Default is "blue".
modsumLabelColor	Colour of the model summary inside the diagrams. Only applies, when parameter showModelSummary is set to TRUE. Use any valid colour value, e.g. modsumLabelColor="grey50" or valueLabelColor=c("#cc3366").
modsumLabelSize	Size of the model summary text. Default is 4.5. Recommended Values range from 2 to 8
showCI	If TRUE (default), a confidence region for the regression line will be plotted. Use ciLevel to specifiy the confidence level.
ciLevel	The confidence level of the confidence region. Only applies when showCI is TRUE. Default is 0.95.
pointAlpha	The alpha values of the scatter plot's point-geoms. Default is 0.2.
pointColor	The color of the scatter plot's point-geoms. Only applies when showScatterPlot is TRUE. Default is "black".
showScatterPlot	If TRUE (default), a scatter plot of response and predictor values for each predictor of the fitted model fit is plotted.

showLoess — If TRUE, an additional loess-smoothed line is plotted.

loessLineColor — The color of the loess-smoothed line. Default is "red". Only applies, if showLoess is TRUE.

showLoessCI — If TRUE, a confidence region for the loess-smoothed line will be plotted. Default is FALSE. Use loessCiLevel to specifiy the confidence level. Only applies, if showLoess is TRUE.

loessCiLevel — The confidence level of the loess-line's confidence region. Only applies, if showLoessCI is TRUE. Default is 0.95.

showModelSummary — If TRUE (default), a summary of the regression model with Intercept, R-square, F-Test and AIC-value is printed to the lower right corner of the diagram.

borderColor — User defined color of whole diagram border (panel border).

axisColor — User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

theme — Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package

See http://rpubs.com/sjPlot/custplot for details and examples.

majorGridColor — Specifies the color of the major grid lines of the diagram background.

minorGridColor — Specifies the color of the minor grid lines of the diagram background.

hideGrid.x — If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y — If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

showTickMarks — Whether tick marks of axes should be shown or not

printPlot — 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

sjp.lm
sjp.reglin
sjp.scatter

Examples

```
# load sample data
data(efc)
# fit model
fit <- lm(neg_c_7 ~ quol_5, data=efc, na.action=na.omit)
# plot regression line
sjp.lm1(fit, efc)
# plot regression line with label strings
sjp.lm1(fit,
        efc,
        axisLabel.x=sji.getVariableLabels(efc)['quol_5'],
        axisLabel.y=sji.getVariableLabels(efc)['neg_c_7'],
        showLoess=TRUE)
```

---

sjp.lmm *Plot beta coefficients of multiple fitted lm's*

Description

Plot beta coefficients (estimates) with confidence intervalls of multiple fitted linear models in one plot.

Usage

```
sjp.lmm(..., title = NULL, titleSize = 1.3, titleColor = "black",
  labelDependentVariables = NULL, legendDepVarTitle = "Dependent Variables",
  legendPValTitle = "p-level", stringModel = "Model", axisLabels.y = NULL,
  axisLabelSize = 1.1, axisLabelAngle.x = 0, axisLabelAngle.y = 0,
  axisLabelColor = "gray30", axisTitle.x = "Estimates",
  axisTitleSize = 1.2, axisTitleColor = c("#444444"), axisLimits = NULL,
  breakTitleAt = 50, breakLabelsAt = 12, breakLegendAt = 20,
  gridBreaksAt = NULL, errorBarWidth = 0, errorBarSize = 0.5,
  errorBarLineType = 1, pointSize = 3, modelPlotSpace = 0.4,
  colorPalette = "Paired", modelColors = NULL, valueLabelColor = "black",
  valueLabelSize = 4, valueLabelAlpha = 1, nsAlpha = 1,
  usePShapes = FALSE, axisColor = NULL, borderColor = NULL,
  interceptLineType = 2, interceptLineColor = "grey70",
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, theme = NULL, flipCoordinates = TRUE,
  legendPos = "right", legendSize = 1, legendBorderColor = "white",
  legendBackColor = "white", showIntercept = FALSE,
  showAxisLabels.y = TRUE, showTickMarks = TRUE, showValueLabels = TRUE,
  labelDigits = 2, showPValueLabels = TRUE, useFacetGrid = FALSE,
  printPlot = TRUE)
```

Arguments

...	One or more fitted lm-objects.
title	Diagram's title as string. Example: title=c("my title")
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
labelDependentVariables	Labels of the dependent variables of all fitted models which have been used as first parameter(s), provided as char vector.
legendDepVarTitle	A character vector used for the title of the dependent variable's legend. Default is "Dependent Variables".
legendPValTitle	A character vector used for the title of the significance level's legend. Default is "p-level". Only applies if usePShapes is TRUE.
stringModel	String constant used as legend text for the model names in case no labels for the dependent variables are provided (see labelDependentVariables). Default is "Model".
axisLabels.y	Labels of the predictor variables (independent vars, betas) that are used for labelling the axis. Passed as vector of strings. Example: axisLabels.y=c("Label1", "Label2", "Label Note: If you use the sji.SPSS function and the sji.getValueLabels function, you receive a list object with label strings. The labels may also be passed as list object. They will be unlisted and converted to character vector automatically.
axisLabelSize	The size of value labels in the diagram. Default is 1.1, recommended values range between 0.7 and 3.0
showAxisLabels.y	Whether beta names (predictor labels) should be shown or not.
showTickMarks	Whether tick marks of axes should be shown or not.
axisTitle.x	A label ("title") for the x axis.
axisTitleColor	The color of the x axis label.
axisTitleSize	The size of the x axis label.
axisLimits	Defines the range of the axis where the beta coefficients and their confidence intervalls are drawn. By default, the limits range from the lowest confidence interval to the highest one, so the diagram has maximum zoom. Use your own values as 2-value-vector, for instance: limits=c(-0.8,0.8).
axisLabelAngle.x	Angle for axis-labels where the estimates labels are printed. Note that due to the coordinate flip, the acutal y-axis with estimates are appearing on the x-axis.
axisLabelAngle.y	Angle for axis-labels where the predictor labels (axisLabels.y) are printed. Note that due to the coordinate flip, the acutal x-axis with predictor labels are appearing on the y-axis.
breakTitleAt	Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title

`breakLabelsAt` Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted

`breakLegendAt` Wordwrap for legend, i.e. names of the dependent variables of each fitted model. See parameter `labelDependentVariables`. Determines how many chars of each dependent variable name is displayed in one line in the legend and when a line break is inserted

`gridBreaksAt` Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Default is `NULL`, so `pretty` gridbeaks will be used.

`pointSize` The size of the points that indicate the beta-value. Default is 3.

`modelPlotSpace` Defines the space between the dots and error bars of the plotted fitted models. Default is 0.3.

`modelColors` A vector with colors for representing the beta values (i.e. points and error bars) of the different fitted models. Thus, the length of this vector must be equal to the length of supplied fitted models, so each model is represented by its own color. You can use:

- `"bw"` or `"black"` for only one colouring in almost black
- `"gray"`, `"grey"` or `"gs"` for a grayscale
- `"brewer"` for colours from the color brewer palette.

If `modelColors` is `"brewer"`, use the `colorPalette` parameter to specify a palette of the `http://colorbrewer2.org`. Else specify your own color values as vector (e.g. `modelColors=c("#f00000", "#00ff00")`).

`colorPalette` If parameter `modelColors` is `brewer`, specify a color palette from the `http://colorbrewer2.org` here. All color brewer palettes supported by ggplot are accepted here.

`axisLabelColor` Colour of the tick labels at the axis (variable names, beta names).

`valueLabelColor`
The colour of the beta values. These values are printed above the plots respectively beside the bar charts. default color is `"black"`.

`valueLabelSize` Size of the value labels. Drfault is 4. Recommended Values range from 2 to 8

`valueLabelAlpha`
The alpha level (transparancy) of the value labels. Default is 1, use any value from 0 to 1.

`nsAlpha` The alpha level (transparancy) of non significant predicors. Points and error bars are affected by this value and plotted with a slight transparancy. Default is 1.

`usePShapes` If `TRUE`, significant levels are distinguished by different point shapes and a related legend is plotted. Default is `FALSE`.

`axisColor` User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

`borderColor` User defined color of whole diagram border (panel border).

`interceptLineType`
The linetype of the intercept line (zero point). Default is `2` (dashed line).

`interceptLineColor`
The color of the intercept line. Default value is `"grey70"`.

errorBarWidth The width of the error bar ends. Default is 0

errorBarSize The size (thickness) of the error bars. Default is 0.5

errorBarLineType
The linetype of error bars. Default is 1 (solid line).

majorGridColor Specifies the color of the major grid lines of the diagram background.

minorGridColor Specifies the color of the minor grid lines of the diagram background.

hideGrid.x If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

theme Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates
If TRUE (default), predictors are plotted on the left y-axis and estimate values are plotted on the x-axis.

legendPos The position of the legend, if a legend is drawn. 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.

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

legendBorderColor
Color of the legend's border. Default is "white", so no visible border is drawn.

legendBackColor
Fill color of the legend's background. Default is "white", so no visible background is drawn.

showIntercept If TRUE, the intercept of the fitted model is also plotted. Default is FALSE.

showValueLabels
Whether the beta value labels should be plotted.

labelDigits The amount of digits for rounding the estimations (see showValueLabels). Default is 2, i.e. estimators have 2 digits after decimal point.

showPValueLabels
Whether the significance levels of each coefficient should be appended to values or not.

useFacetGrid TRUE when each model should be plotted as single facet instead of an integrated single graph.

printPlot 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`).

References

http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.lm
sjt.lm
sjp.lm.ma
sjp.glmm

Examples

```
# prepare dummy variables for binary logistic regression
# Now fit the models. Note that all models share the same predictors
# and only differ in their dependent variable
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)
# fit third model
fit3 <- lm(tot_sc_e ~ c160age + c12hour + c161sex + c172code, data=efc)

# plot multiple models
sjp.lmm(fit1, fit2, fit3, useFacetGrid=TRUE)

# plot multiple models with legend labels and point shapes instead of value  labels
sjp.lmm(fit1, fit2, fit3,
         axisLabels.y=c("Carer's Age", "Hours of Care",
                        "Carer's Sex", "Educational Status"),
         labelDependentVariables=c("Barthel Index", "Negative Impact", "Services used"),
         showValueLabels=FALSE,
         showPValueLabels=FALSE,
         usePShapes=TRUE,
         nsAlpha=0.3)
```

---

sjp.pca *Plot PCA results*

Description

Performes a principle component analysis on a data frame or matrix and plots the factor solution as ellipses or tiles.

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

```
sjp.pca(data, numberOfFactors = NULL, factorLoadingTolerance = 0.1,
  plotEigenvalues = FALSE, digits = 2, title = NULL, titleSize = 1.3,
  titleColor = "black", axisLabels.y = NULL, type = "tile",
  geomAlpha = 0.8, valueLabelColor = "black", valueLabelSize = 4.5,
  valueLabelAlpha = 1, circleSize = 10, outlineColor = "black",
  outlineSize = 0.2, axisColor = NULL, borderColor = NULL,
  axisLabelSize = 1.1, axisLabelColor = "gray30", axisLabelAngle.x = 0,
  axisLabelAngle.y = 0, breakTitleAt = 50, breakLabelsAt = 20,
  hideLegend = TRUE, legendTitle = NULL, showValueLabels = TRUE,
  showTickMarks = FALSE, showCronbachsAlpha = TRUE, fillColor = NULL,
  majorGridColor = NULL, minorGridColor = NULL, theme = NULL,
  printPlot = TRUE)
```

Arguments

`data`	A data frame with factors (each columns one variable) that should be used to compute a PCA, or a `prcomp` object.
`numberOfFactors`	A predefined number of factors to use for the calculating the varimax rotation. By default, this value is `NULL` and the amount of factors is calculated according to the Kaiser-criteria. See paramater `plotEigenvalues`.
`factorLoadingTolerance`	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
`plotEigenvalues`	If `TRUE`, a plot showing the Eigenvalues according to the Kaiser criteria is plotted to determine the number of factors.
`digits`	The amount of decimals used. Default is 2.
`title`	Title of the diagram, plotted above the whole diagram panel.
`titleSize`	The size of the plot title. Default is 1.3.
`titleColor`	The color of the plot title. Default is `"black"`.

`axisLabels.y` The item labels that are printed on the y-axis. If no item labels are provided (default), the data frame's column names are used. Item labels must be a string vector, e.g.: `axisLabels.y=c("Var 1", "Var 2", "Var 3")`.

`type` Indicates whether `"circle"` (default) or `"tile"` geoms should be used for plotting.

`geomAlpha` Specify the transparancy (alpha value) of geom objects (circles or tiles). Default is 0.8.

`valueLabelColor`
The color of the value labels (numbers) inside the diagram. Default is `"black"`.

`valueLabelSize` The size of value labels in the diagram. Default is 4.5, recommended values range between 2 and 8.

`valueLabelAlpha`
Specify the transparancy (alpha value) of value labels. Default is 0.8

`circleSize` Specifies the circle size factor. The circle size depends on the correlation value multiplicated with this factor. Default is 10.

`outlineColor` Defines the outline color of geoms (circles or tiles). Default is `"black"`.

`outlineSize` Defines the outline size of geoms (circles or tiles). Default is 1.

`axisColor` User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

`borderColor` User defined color of whole diagram border (panel border).

`axisLabelSize` The size of variable labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0.

`axisLabelColor` User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

`axisLabelAngle.x`
Angle for x-axis-labels.

`axisLabelAngle.y`
Angle for y-axis-labels.

`breakTitleAt` Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title. Default is 50.

`breakLabelsAt` Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted. Default is 12.

`hideLegend` Show or hide the legend. The legend indicates the strength of correlations by gradient colour fill. Default is `TRUE`, hence the legend is hidden.

`legendTitle` The legend title, provided as string, e.g. `legendTitle=c("Factor loading")`. Default is `NULL`, hence no legend title is used.

`showValueLabels`
Whether factor loading values should be plotted to each geom. Default is `TRUE`.

`showTickMarks` Whether tick marks should be plotted or not. Default is `FALSE`.

`showCronbachsAlpha`
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 and no `prcomp` object.

fillColor	A color palette for fillng the geoms. If not specified, the 5th diverging color palette from the color brewer palettes (RdBu) is used, resulting in red colors for negative and blue colors for positive factor loadings, that become lighter the weaker the loadings are. Use any color palette that is suitbale for the scale_fill_gradientn parameter of ggplot2.
majorGridColor	Specifies the color of the major grid lines of the diagram background.
minorGridColor	Specifies the color of the minor grid lines of the diagram background.
theme	Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids. • Use "bw" for a white background with gray grids • "classic" for a classic theme (black border, no grids) • "minimal" for a minimalistic theme (no border,gray grids) • "none" for no borders, grids and ticks or • "themr" if you are using the ggthemr package See http://rpubs.com/sjPlot/custplot for details and examples.
printPlot	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 varimax-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 parameter factorLoadingTolerance for more details) (removed.df)
- the ggplot-object (plot),
- the data frame that was used for setting up the ggplot-object (df).

Note

This PCA uses the prcomp function and the varimax rotation.

References

http://strengejacke.wordpress.com/sjplot-r-package/

http://strengejacke.wordpress.com/2013/07/08/plotting-principal-component-analysis-with-ggplot-rsta

See Also

sjt.pca
sjs.reliability
sjt.itemanalysis
sjs.cronbach

Examples

```
# randomly create data frame with 7 items, each consisting of 4 categories
likert_4 <- data.frame(sample(1:4, 500, replace=TRUE, prob=c(0.2,0.3,0.1,0.4)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.5,0.25,0.15,0.1)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.4,0.15,0.25,0.2)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.25,0.1,0.4,0.25)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.1,0.4,0.4,0.1)),
                       sample(1:4, 500, replace=TRUE),
                       sample(1:4, 500, replace=TRUE, prob=c(0.35,0.25,0.15,0.25)))

# Create variable labels
colnames(likert_4) <- c("V1", "V2", "V3", "V4", "V5", "V6", "V7")

# plot results from PCA as square-tiled "heatmap"
sjp.pca(likert_4)

# manually compute PCA
pca <- prcomp(na.omit(likert_4), 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, plotEigenvalues=TRUE, type="circle")


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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

sjp.pca(df)

# --------------------------------
# auto-detection of labels
# --------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
sjp.pca(efc[,c(start:end)])
```

`sjp.reglin` *Plot regression lines for each predictor*

Description

Plot regression lines with confidence intervals for each single predictor of a fitted model. This method extracts all predictors of a fitted model and fits each of them against the response variable.

This function plots two lines: The resulting linear regression line including confidence interval (in blue) and a loess-smoothed line without confidence interval (in red). The better the linear relationship of predictor and response is, the more both lines should overlap (i.e. the red loess-smoothed line is almost linear).

Furthermore, a scatter plot of response and predictor values is plotted.

Usage

```
sjp.reglin(fit, data, lineColor = "blue", showCI = TRUE, ciLevel = 0.95,
  pointAlpha = 0.2, pointColor = "black", showScatterPlot = TRUE,
  showLoess = TRUE, loessLineColor = "red", showLoessCI = FALSE,
  loessCiLevel = 0.95, printPlot = TRUE)
```

Arguments

`fit`	The model of the linear regression (lm-Object).
`data`	The data/dataset/dataframe used in the fitted model.
`lineColor`	The color of the regression line. Default is `"blue"`.
`showCI`	If `TRUE` (default), a confidence region for the regression line will be plotted. Use `ciLevel` to specifiy the confidence level.
`ciLevel`	The confidence level of the confidence region. Only applies when `showCI` is `TRUE`. Default is 0.95.
`pointAlpha`	The alpha values of the scatter plot's point-geoms. Default is 0.2.
`pointColor`	The color of the scatter plot's point-geoms. Only applies when `showScatterPlot` is `TRUE`. Default is `"black"`.
`showScatterPlot`	If `TRUE` (default), a scatter plot of response and predictor values for each predictor of the fitted model `fit` is plotted.
`showLoess`	If `TRUE` (default), an additional loess-smoothed line is plotted.
`loessLineColor`	The color of the loess-smoothed line. Default is `"red"`. Only applies, if `showLoess` is `TRUE`.
`showLoessCI`	If `TRUE`, a confidence region for the loess-smoothed line will be plotted. Default is `FALSE`. Use `loessCiLevel` to specifiy the confidence level. Only applies, if `showLoess` is `TRUE`.
`loessCiLevel`	The confidence level of the loess-line's confidence region. Only applies, if `showLoessCI` is `TRUE`. Default is 0.95.
`printPlot`	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-objects with the complete plot-list (`plot.list`) as well as the data frame that were used for setting up the ggplot-objects (`df.list`).

References

http://rpubs.com/sjPlot/sjplm

See Also

sjp.lm
sjp.scatter
sjp.lm.ma
sjp.lm.int

Examples

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

# reression line and scatter plot
sjp.reglin(fit, efc)

# reression line w/o scatter plot
sjp.reglin(fit, efc, showScatterPlot=FALSE)

# reression line w/o CI
sjp.reglin(fit, efc, showCI=FALSE)
```

---

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, y, grp = NULL, title = NULL, titleSize = 1.3,
  titleColor = "black", legendTitle = NULL, legendLabels = NULL,
  axisTitle.x = NULL, axisTitle.y = NULL, axisTitleColor = "black",
  axisTitleSize = 1.3, axisTickMarkSize = 1, axisTickMarkColor = "grey50",
  breakTitleAt = 50, breakLegendTitleAt = 20, breakLegendLabelsAt = 20,
  pointAlpha = 0.5, pointSize = 3, pointColors = NULL,
  legendPos = "right", legendSize = 1, legendBorderColor = "white",
  legendBackColor = "white", showTickMarkLabels.x = TRUE,
  showTickMarkLabels.y = TRUE, showTickMarks = TRUE,
```

```
  majorGridColor = NULL, minorGridColor = NULL, hideGrid.x = FALSE,
  hideGrid.y = FALSE, borderColor = NULL, axisColor = NULL,
  showGroupFitLine = FALSE, showTotalFitLine = FALSE, showSE = FALSE,
  fitmethod = "lm", useJitter = FALSE, autojitter = TRUE,
  jitterRatio = 0.15, showRug = FALSE, hideLegend = FALSE, theme = NULL,
  useFacetGrid = FALSE, printPlot = TRUE)
```

Arguments

x	A vector (variable) indicating the x positions.
y	A vector (variable) indicating the y positions.
grp	A grouping variable. If not NULL, the scatter plot will be grouped. See examples below. Default is NULL, i.e. not grouping is done.
title	Title of the diagram, plotted above the whole diagram panel. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
legendTitle	Title of the diagram's legend.
legendLabels	Labels for the guide/legend.
axisTitle.x	A label (title) for the x axis. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).
axisTitle.y	A label (title) for the y axis. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).
axisTitleColor	The color of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.
axisTitleSize	The size of the x and y axis labels. Refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.
axisTickMarkSize	The size of tick mark values of both x and y axis. Default is 1, recommended values range between 0.5 and 3.0
axisTickMarkColor	User defined color for tick mark values. If not specified, a default mid gray color will be used for the labels.
breakTitleAt	Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.
breakLegendTitleAt	Wordwrap for diagram legend title. Determines how many chars of the legend's title are displayed in one line and when a line break is inserted.
breakLegendLabelsAt	Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted.
pointAlpha	The alpha values of scattered points. Useful to better cope with overplotting. Default is 0.5

pointSize
: The size of scattered points.

pointColors
: The color(s) of scattered points. If grp is not NULL, groups are indicated by different colors, thus a vector with multiple color values has to be supplied. By default, the Set1 palette of diverging palette type is chosen (see http://colorbrewer2.org).

legendPos
: The position of the legend, if a legend is drawn. 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.

legendSize
: The text size of the legend. Default is 1. Relative size, so recommended values are from 0.3 to 2.5

legendBorderColor
: Color of the legend's border. Default is "white", so no visible border is drawn.

legendBackColor
: Fill color of the legend's background. Default is "white", so no visible background is drawn.

showTickMarkLabels.x
: Whether x axis tick mark labels should be shown or not.

showTickMarkLabels.y
: Whether y axis tick mark labels should be shown or not.

showTickMarks
: Whether tick marks of axes should be shown or not.

showGroupFitLine
: If TRUE, a fitted line for each group is drawn. See fitmethod to change the fit method of the fitted lines.

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

showSE
: If TRUE, a shaded region indicating the standard error of the fitted lines will be added.

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

  - "lm"
  - "glm"
  - "loess"
  - "auto"

  (see http://docs.ggplot2.org/current/stat_smooth.html for more details).

useJitter
: If TRUE, points will be jittered (to avoid overplotting).

autojitter
: 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 jitterRatio) of the approximated amount of unique point coordinates seem to overlap, they are automatically jittered.

jitterRatio
: The ratio of tolerated overlapping (see autojitter). If approximated amount of overlapping points exceed this ration, they are automatically jittered. Default is 0.15. Valid values range between 0 and 1.

showRug	If TRUE, a marginal rug plot is displayed in the graph (see http://docs.ggplot2.org/current/geom_rug.html for more details).)
hideLegend	Indicates whether legend (guide) should be shown or not.
borderColor	User defined color of whole diagram border (panel border).
axisColor	User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).
majorGridColor	Specifies the color of the major grid lines of the diagram background.
minorGridColor	Specifies the color of the minor grid lines of the diagram background.
hideGrid.x	If TRUE, the x-axis-gridlines are hidden. Default is FALSE.
hideGrid.y	If TRUE, the y-axis-gridlines are hidden. Default is FALSE.
theme	Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids. • Use "bw" for a white background with gray grids • "classic" for a classic theme (black border, no grids) • "minimal" for a minimalistic theme (no border,gray grids) • "none" for no borders, grids and ticks or • "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the pointColors parameter) See http://rpubs.com/sjPlot/custplot for details and examples.
useFacetGrid	TRUE when each scatter plot group should be plotted as single facet instead of an integrated single graph. Only applies if grp is not NULL. Each category of grp will be plotted in an own facet.
printPlot	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).

References

- http://rpubs.com/sjPlot/sjpscatter
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.lm1
sjp.reglin

Examples

```
# load sample date
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, showRug=TRUE)

# grouped and labelled scatter plot
sjp.scatter(efc$c160age,efc$e17age, efc$e42dep, title="Scatter Plot",
            legendTitle=sji.getVariableLabels(efc)['e42dep'],
            legendLabels=sji.getValueLabels(efc)[['e42dep']],
            axisTitle.x=sji.getVariableLabels(efc)['c160age'],
            axisTitle.y=sji.getVariableLabels(efc)['e17age'],
            showGroupFitLine=TRUE)

# grouped and labelled scatter plot as facets
sjp.scatter(efc$c160age,efc$e17age, efc$e42dep, title="Scatter Plot",
            legendTitle=sji.getVariableLabels(efc)['e42dep'],
            legendLabels=sji.getValueLabels(efc)[['e42dep']],
            axisTitle.x=sji.getVariableLabels(efc)['c160age'],
            axisTitle.y=sji.getVariableLabels(efc)['e17age'],
            showGroupFitLine=TRUE, useFacetGrid=TRUE, showSE=TRUE)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, sji.getVariableLabels(efc))
sjp.scatter(efc$c160age,efc$e17age, efc$e42dep,
            title="auto", axisTitle.x="auto", axisTitle.y="auto")
```

---

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, legendLabels = NULL, orderBy = NULL, weightBy = NULL,
  weightByTitleString = NULL, hideLegend = FALSE, reverseOrder = TRUE,
  title = NULL, titleSize = 1.3, titleColor = "black",
  legendTitle = NULL, includeN = TRUE, axisLabels.y = NULL,
  axisLabelSize = 1.1, axisLabelAngle.x = 0, axisLabelColor = "gray30",
  valueLabelSize = 4, valueLabelColor = "black", breakTitleAt = 50,
  breakLabelsAt = 30, breakLegendTitleAt = 30, breakLegendLabelsAt = 28,
  gridBreaksAt = 0.2, expand.grid = FALSE, barWidth = 0.5,
  barColor = NULL, colorPalette = "GnBu", barAlpha = 1,
  borderColor = NULL, axisColor = NULL, barOutline = FALSE,
  barOutlineSize = 0.2, barOutlineColor = "black", majorGridColor = NULL,
  minorGridColor = NULL, hideGrid.x = FALSE, hideGrid.y = FALSE,
  axisTitle.x = NULL, axisTitle.y = NULL, axisTitleColor = "black",
  axisTitleSize = 1.3, theme = NULL, showTickMarks = FALSE,
  showValueLabels = TRUE, showPercentageAxis = TRUE,
  jitterValueLabels = FALSE, showItemLabels = TRUE,
  showSeparatorLine = FALSE, separatorLineColor = "grey80",
  separatorLineSize = 0.3, legendPos = "right", legendSize = 1,
  legendBorderColor = "white", legendBackColor = "white",
  flipCoordinates = TRUE, printPlot = TRUE)
```

Arguments

items	A data frame with each column representing one likert-item.
legendLabels	A list or vector of strings that indicate the likert-scale-categories and which appear as legend text.
orderBy	Indicates whether the items should be ordered by highest count of first or last category of items. Use "first" to order ascending by lowest count of first category, "last" to order ascending by lowest count of last category or NULL (default) for no sorting. In case you want to revers order (descending from highest count), use reverseOrder parameter.
weightBy	A weight factor that will be applied to weight all cases from items.
weightByTitleString	If a weight factor is supplied via the parameter weightBy, the diagram's title may indicate this with a remark. Default is NULL, so the diagram's title will not be modified when cases are weighted. Use a string as parameter, e.g.: weightByTitleString=" (weighted)".
hideLegend	Indicates whether legend (guide) should be shown or not.
reverseOrder	If TRUE, the item order on the x-axis is reversed.
title	Title of the diagram, plotted above the whole diagram panel.
titleSize	The size of the plot title. Default is 1.3.
titleColor	The color of the plot title. Default is "black".
legendTitle	Title of the diagram's legend.
includeN	If TRUE (default), the N of each item is included into axis labels.

`axisLabels.y` Labels for the y-axis (the labels of the `items`). These parameters must be passed as list! Example: `axisLabels.y=list(c("Q1", "Q2", "Q3"))` Axis labels will automatically be detected, when they have a `"variable.lable"` attribute (see `sji.setVariableLabels`) for details).

`axisLabelSize` The size of category labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0

`axisLabelAngle.x`
Angle for axis-labels.

`axisLabelColor` User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

`valueLabelSize` The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8

`valueLabelColor`
The color of value labels in the diagram. Default is black.

`breakTitleAt` Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

`breakLabelsAt` Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted.

`breakLegendTitleAt`
Wordwrap for diagram legend title. Determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

`breakLegendLabelsAt`
Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

`gridBreaksAt` Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Valid values range from 0 to 1.

`expand.grid` If `TRUE` (default), the diagram has margins, i.e. the y-axis is not exceeded to the diagram's boundaries.

`barWidth` Width of bars. Recommended values for this parameter are from 0.4 to 1.5

`barColor` User defined color for bars. If not specified (`NULL`), a default blue color palette will be used for the bar charts. You can use pre-defined color-sets that are independent from the amount of categories: If barColor is `"brewer"`, use the `colorPalette` parameter to specify a palette of the color brewer Else specify your own color values as vector (e.g. `barColor=c("darkred", "red", "green", "darkgreen")`)

`colorPalette` If `barColor` is `"brewer"`, specify a color palette from the color brewer here. All color brewer palettes supported by ggplot are accepted here.

`barAlpha` Specify the transparancy (alpha value) of bars.

`borderColor` User defined color of whole diagram border (panel border).

`barOutline` If `TRUE`, each bar gets a colored outline. Default is `FALSE`.

`barOutlineColor`
The color of the bar outline. Only applies, if `barOutline` is set to `TRUE`.

`barOutlineSize` The size of the bar outlines. Only applies if `barOutline` is `TRUE`. Default is 0.2

`majorGridColor` Specifies the color of the major grid lines of the diagram background.

`minorGridColor` Specifies the color of the minor grid lines of the diagram background.

`hideGrid.x` If `TRUE`, the x-axis-gridlines are hidden. Default if `FALSE`.

`hideGrid.y` If `TRUE`, the y-axis-gridlines are hidden. Default if `FALSE`.

`axisColor` User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

`axisTitle.x` A label for the x axis. Useful when plotting histograms with metric scales where no category labels are assigned to the x axis.

`axisTitle.y` A label for the y axis. Useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

`axisTitleColor` The color of the x and y axis labels. Refers to `axisTitle.x` and `axisTitle.y`, not to the tick mark or category labels.

`axisTitleSize` The size of the x and y axis labels. Refers to `axisTitle.x` and `axisTitle.y`, not to the tick mark or category labels.

`showValueLabels`
Whether counts and percentage values should be plotted to each bar.

`showPercentageAxis`
If `TRUE` (default), the percentage values at the x-axis are shown.

`jitterValueLabels`
If `TRUE`, the value labels on the bars will be "jittered", i.e. they have alternating vertical positions to avoid overlapping of labels in case bars are very short. Default is `FALSE`.

`showItemLabels` Whether x axis text (category names) should be shown or not.

`showTickMarks` Whether tick marks of axes should be shown or not.

`showSeparatorLine`
If `TRUE`, a line is drawn to visually "separate" each bar in the diagram.

`separatorLineColor`
The color of the separator line. only applies, if `showSeparatorLine` is `TRUE`.

`separatorLineSize`
The size of the separator line. only applies, if `showSeparatorLine` is `TRUE`.

`legendPos` The position of the legend. Default is `"right"`. Use one of the following values: `"right"`, `"left"`, `"bottom"`, `"top"`.

`legendSize` The size of the legend.

`legendBorderColor`
The border color of the legend.

`legendBackColor`
The background color of the legend.

`theme` Specifies the diagram's background theme. Default (parameter `NULL`) is a gray background with white grids.

- Use `"bw"` for a white background with gray grids
- `"classic"` for a classic theme (black border, no grids)
- `"minimal"` for a minimalistic theme (no border,gray grids)
- `"none"` for no borders, grids and ticks or

- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates
: If TRUE, the x and y axis are swapped.

printPlot
: 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 (http://www.clas.ufl.edu/users/forrest/) for bug fixes

References

- http://rpubs.com/sjPlot/sjpstackfrq
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.likert
sjt.stackfrq

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 <- list(c("Independent", "Slightly dependent", "Dependent", "Severely dependent"))

# create item labels
items <- list(c("Q1", "Q2", "Q3", "Q4", "Q5"))

# plot stacked frequencies of 5 (ordered) item-scales
sjp.stackfrq(likert_4, legendLabels=levels_4, axisLabels.y=items)


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

```
# Data from the EUROFAMCARE sample dataset
# ------------------------------
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")

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(efc)
vallabs <- sji.getValueLabels(efc)

# create value labels. We need just one variable of
# the COPE-index scale because they have all the same
# level / categorie / value labels
levels <- vallabs['c82cop1']

# create item labels
items <- list(varlabs[c(start:end)])

sjp.stackfrq(efc[,c(start:end)], legendLabels=levels,
             axisLabels.y=items, jitterValueLabels=TRUE)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
sjp.stackfrq(efc[,c(start:end)])
```

`sjp.vif` *Plot Variance Inflation Factors of linear models*

Description

Plots the Variance Inflation Factors (check for multicollinearity) of (generalized) linear models. Values below 5 are good and indicating no multicollinearity, values between 5 and 10 may be tolerable. Values greater than 10 are not acceptable and indicate multicollinearity between model's predictors.

Usage

```
sjp.vif(fit)
```

Arguments

`fit`	The fitted (generalized) linear model which should be checked for multicollinearity.

Value

(invisibly) returns the VIF values.

Examples

```
# fit linear model
fit <- lm(airquality$Ozone ~ airquality$Wind + airquality$Temp + airquality$Solar.R)
# plot VIF values
sjp.vif(fit)
```

---

sjp.xtab *Plot contingency tables*

Description

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

Usage

```
sjp.xtab(y, x, title = NULL, titleSize = 1.3, titleColor = "black",
  legendTitle = NULL, weightBy = NULL, weightByTitleString = NULL,
  type = "bars", tableIndex = "col", reverseOrder = FALSE,
  maxYlim = TRUE, upperYlim = NULL, axisLabels.x = NULL,
  axisLabelColor.x = "darkgray", axisLabelAngle.x = 0,
  axisLabelSize.x = 1.1, legendLabels = NULL, valueLabelSize = 4,
  valueLabelColor = "black", valueLabelPosOnTop = TRUE,
  stringTotal = "Total", breakTitleAt = 50, breakLabelsAt = 12,
  breakLegendTitleAt = 20, breakLegendLabelsAt = 20, gridBreaksAt = 0.2,
  barWidth = 0.7, barSpace = 0.1, barPosition = "dodge",
  barColor = NULL, colorPalette = "GnBu", barAlpha = 1, lineType = 1,
  lineSize = 1, lineAlpha = 1, lineDotSize = 3, smoothLines = FALSE,
  borderColor = NULL, axisColor = NULL, barOutline = FALSE,
  barOutlineSize = 0.2, barOutlineColor = "black", majorGridColor = NULL,
  minorGridColor = NULL, hideGrid.x = FALSE, hideGrid.y = FALSE,
  expand.grid = FALSE, showValueLabels = TRUE, jitterValueLabels = FALSE,
  showCategoryLabels = TRUE, showTickMarks = TRUE,
  showTableSummary = TRUE, tableSummaryPos = "r", showTotalColumn = TRUE,
  hideLegend = FALSE, axisTitle.x = NULL, axisTitle.y = NULL,
  axisTitleColor = "black", axisTitleSize = 1.3, theme = NULL,
  flipCoordinates = FALSE, printPlot = TRUE)
```

Arguments

y	The variable which proportions (percentage values) should be plotted. The percentage proportion (within table row, table column or complete table, see parameter tableIndex of this variable are along the y-axis, the variable's categories on the x-axis.

x — The grouping variable, where each value represents a single bar chart within each category of the y variable.

weightBy — A weight factor that will be applied to weight all cases from y.

weightByTitleString — If a weight factor is supplied via the parameter weightBy, the diagram's title may indicate this with a remark. Default is NULL, so the diagram's title will not be modified when cases are weighted. Use a string as parameter, e.g.: weightByTitleString=" (weighted)".

type — The plot type. may be either "b", "bar", "bars" (default) for bar charts, or "l", "line", "lines" for line diagram.

tableIndex — Indicates which data from the proportional table should be plotted. Use "row" for calculating row percentages, "col" for column percentages and "cell" for cell percentages. Only when tableIndex is "col", an additional bar chart with the total sum of each column (i.e. of each category on the x-axis) can be added with the parameter showTotalColumn.

barPosition — Indicates whether bars should be positioned side-by-side (default, or use "dodge" as parameter) or stacked (use "stack" as parameter).

hideLegend — Indicates whether legend (guide) should be shown or not. Default is FALSE, thus the legend is shown.

reverseOrder — Whether the categories along the x-axis should apper in reversed order or not.

maxYlim — Indicates how to calculate the maximum limit of the y-axis. If TRUE, the y-axes ranges from 0 to 100 depends on the highest percentage value of a variable's answer category. In this case, the y-axis breaks may change, depending on the variable.

upperYlim — Uses a pre-defined upper limit for the y-axis. Overrides the maxYlim parameter.

title — Title of the diagram, plotted above the whole diagram panel. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

titleSize — The size of the plot title. Default is 1.3.

titleColor — The color of the plot title. Default is "black".

legendTitle — Title of the diagram's legend.

axisLabels.x — Labels for the x-axis breaks.

legendLabels — Labels for the guide/legend.

axisLabelSize.x — The size of category labels at the axes. Default is 1.1, recommended values range between 0.5 and 3.0.

valueLabelSize — The size of value labels in the diagram. Default is 4, recommended values range between 2 and 8.

axisLabelAngle.x — Angle for axis-labels.

breakTitleAt — Wordwrap for diagram title. Determines how many chars of the title are displayed in one line and when a line break is inserted into the title.

breakLabelsAt Wordwrap for diagram labels. Determines how many chars of the category labels are displayed in one line and when a line break is inserted.

breakLegendTitleAt
Wordwrap for diagram legend title. Determines how many chars of the legend's title are displayed in one line and when a line break is inserted.

breakLegendLabelsAt
Wordwrap for diagram legend labels. Determines how many chars of the legend labels are displayed in one line and when a line break is inserted.

gridBreaksAt Sets the breaks on the y axis, i.e. at every n'th position a major grid is being printed. Valid values range from 0 to 1.

barWidth Width of bars. Recommended values for this parameter are from 0.4 to 1.5.

barSpace Spacing between bars. If unchanges, the grouped bars are sticked together and have no space in between. Recommended values for this parameter are from 0 to 0.5.

barColor User defined color for bars.

- If not specified (NULL), the "Set1" color palette from http://colorbrewer2.org will be used for the bar charts.
- If barColor is "gs", a greyscale will be used.
- If barColor is "bw", a monochrome white filling will be used.
- If barColor is "brewer", use the colorPalette parameter to specify a palette of the http://colorbrewer2.org.

Else specify your own color values as vector (e.g. barColor=c("#f00000", "#00ff00", "#0080ff")).

colorPalette If barColor is "brewer", specify a color palette from the http://colorbrewer2.org here. All color brewer palettes supported by ggplot are accepted here.

barAlpha Specify the transparancy (alpha value) of bars.

lineType The linetype when using line diagrams. Only applies, when parameter type is set to "lines".

lineSize The size of lines in a line diagram. Only applies, when parameter type is set to "lines".

lineAlpha The alpha value of lines in a line diagram. Only applies, when parameter type is set to "lines".

lineDotSize Size of dots. Only applies, when parameter type is set to "lines".

smoothLines Prints a smooth line curve. Only applies, when parameter type is set to "lines".

axisLabelColor.x
User defined color for axis labels. If not specified, a default dark gray color palette will be used for the labels.

borderColor User defined color of whole diagram border (panel border).

axisColor User defined color of axis border (y- and x-axis, in case the axes should have different colors than the diagram border).

barOutline If TRUE, each bar gets a colored outline. Default is FALSE.

barOutlineColor
The color of the bar outline. Only applies, if barOutline is set to TRUE

barOutlineSize The size of the bar outlines. Only applies if barOutline is TRUE. Default is 0.2

majorGridColor specifies the color of the major grid lines of the diagram background

minorGridColor specifies the color of the minor grid lines of the diagram background

hideGrid.x If TRUE, the x-axis-gridlines are hidden. Default if FALSE.

hideGrid.y If TRUE, the y-axis-gridlines are hidden. Default if FALSE.

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

showValueLabels
Whether counts and percentage values should be plotted to each bar

jitterValueLabels
If TRUE, the value labels on the bars will be "jittered", i.e. they have alternating vertical positions to avoid overlapping of labels in case bars are very short. Default is FALSE.

valueLabelPosOnTop
Whether value labels should be displayed on top of dodged bars or inside the bars. Default is TRUE, i.e. the value labels are displayed on top of the bars. Only applies if parameter barPosition is dodge (default).

stringTotal The string for the legend label when a total-column is added. Only applies if showTotalColumn is TRUE. Default is "Total".

showCategoryLabels
Whether x axis text (category names) should be shown or not.

showTickMarks Whether tick marks of axes should be shown or not.

showTableSummary
If TRUE (default), a summary of the cross tabulation with N, Chi-square (see chisq.test), df, Cramer's V or Phi-value and p-value is printed to the upper right corner of the diagram. 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-square test. If the table's matrix is larger than 2x2, Fisher's excact test with Monte Carlo simulation is computed. Only applies to bar-charts or dot-plots, i.e. when parameter type is either "bars" or "dots".

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

showTotalColumn
if tableIndex is "col", an additional bar chart with the sum within each category and it's percentages will be added to each category.

valueLabelColor
The color of the value labels (numbers) inside the diagram.

axisTitle.x A label for the x axis. useful when plotting histograms with metric scales where no category labels are assigned to the x axis. Use "auto" to automatically detect variable names that will be used as title (see sji.setVariableLabels) for details).

axisTitle.y A label for the y axis. useful when plotting histograms with metric scales where no category labels are assigned to the y axis.

axisTitleColor The color of the x and y axis labels. refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

axisTitleSize The size of the x and y axis labels. refers to axisTitle.x and axisTitle.y, not to the tick mark or category labels.

theme Specifies the diagram's background theme. Default (parameter NULL) is a gray background with white grids.

- Use "bw" for a white background with gray grids
- "classic" for a classic theme (black border, no grids)
- "minimal" for a minimalistic theme (no border,gray grids)
- "none" for no borders, grids and ticks or
- "themr" if you are using the ggthemr package (in such cases, you may use the ggthemr::swatch function to retrieve theme-colors for the barColor parameter)

See http://rpubs.com/sjPlot/custplot for details and examples.

flipCoordinates

If TRUE, the x and y axis are swapped.

printPlot 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

Since package version 1.3, the parameters axisLabels.x and legendLabels, which represent the value labels, are retrieved automatically if a) the variables x and y come from a data frame that was imported with the sji.SPSS function (because then value labels are attached as attributes to the data) or b) when the variables are factors with named factor levels (e.g., see column group in dataset PlantGrowth). However, you still can use own parameters as axis- and legendlabels.

References

- http://rpubs.com/sjPlot/sjpxtab
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjt.xtab

Examples

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

```
# plot "cross tablulation" of x and y
sjp.xtab(y,x)

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

# plot "cross tablulation" of x and y as stacked proportional bars
sjp.xtab(y,x, tableIndex="row", barPosition="stack", flipCoordinates=TRUE)

# grouped bars with EUROFAMCARE sample dataset
# dataset was importet from an SPSS-file, using:
# efc <- sji.SPSS("efc.sav", enc="UTF-8")
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)

sjp.xtab(efc$e42dep,
         efc$e16sex,
         title=efc.var['e42dep'],
         axisLabels.x=efc.val[['e42dep']],
         legendTitle=efc.var['e16sex'],
         legendLabels=efc.val[['e16sex']])

sjp.xtab(efc$e16sex,
         efc$e42dep,
         title=efc.var['e16sex'],
         axisLabels.x=efc.val[['e16sex']],
         legendTitle=efc.var['e42dep'],
         legendLabels=efc.val[['e42dep']])

sjp.xtab(efc$e16sex,
         efc$e42dep,
         title=efc.var['e16sex'],
         axisLabels.x=efc.val[['e16sex']],
         legendTitle=efc.var['e42dep'],
         legendLabels=efc.val[['e42dep']],
         tableIndex="row",
         barPosition="stack",
         flipCoordinates=TRUE,
         jitterValueLabels=TRUE)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, efc.var)
sjp.xtab(efc$e16sex, efc$e42dep, title="auto", axisTitle.x="auto")
```

---

sjs.betaCoef *Retrieve std. beta coefficients of lm*

Description

Returns the standardized beta coefficients of a fitted linear model.

Usage

```
sjs.betaCoef(fit)
```

Arguments

fit	A fitted linear model.

Value

The standardiized beta coefficients of the fitted linear model.

Note

"Standardized coefficients refer to how many standard deviations a dependent variable will change, per standard deviation increase in the predictor variable. Standardization of the coefficient is usually done to answer the question of which of the independent variables have a greater effect on the dependent variable in a multiple regression analysis, when the variables are measured in different units of measurement (for example, income measured in dollars and family size measured in number of individuals)." (Source: Wikipedia)

References

http://en.wikipedia.org/wiki/Standardized_coefficient

See Also

sjp.lm
sjt.lm

Examples

```
# fit linear model
fit <- lm(airquality$Ozone ~ airquality$Wind + airquality$Temp + airquality$Solar.R)
# print std. beta coefficients
sjs.betaCoef(fit)
```

`sjs.chi2.gof` *Performs a Chi-square goodness-of-fit-test*

Description

Performs a Chi-square goodness-of-fit-test

Usage

```
sjs.chi2.gof(var, prob, weights = NULL)
```

Arguments

`var`	a numeric vector / variable.
`prob`	a vector of probabilities (indicating the population probabilities) of the same length as `var`'s amount of categories / factor levels. Use `nrow(table(var))` to determine the amount of necessary values for `prob`.
`weights`	a vector with weights, used to weight `var`.

Value

(insisibly) returns the object of the computed `chisq.test`.

Note

This function is a convenient function for `chisq.test`, performing goodness-of-fit test.

See Also

`sjs.mwu`, `sju.aov1.levene` and `wilcox.test`, `ks.test`, `kruskal.test`, `t.test`, `chisq.test`, `fisher.test`, `ks.test`

Examples

```
data(efc)
# differing from population
sjs.chi2.gof(efc$e42dep, c(0.3,0.2,0.22,0.28))
# equal to population
sjs.chi2.gof(efc$e42dep, prop.table(table(efc$e42dep)))
```

sjs.cramer *Cramer's V for a contingency table*

Description

Compute Cramer's V for a table with more than 2x2 fields.

Usage

```
sjs.cramer(tab)
```

Arguments

tab	A simple table or ftable. Tables of class xtabs and other will be coerced to ftable objects.

Value

The table's Cramer's V.

See Also

sjs.table.values
sjs.phi

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:3, 30, TRUE))
sjs.cramer(tab)
```

sjs.cronbach *Calculates Cronbach's Alpha for a matrix*

Description

This function calculates the Cronbach's alpha value for each column of a data frame or matrix.

Usage

```
sjs.cronbach(df)
```

Arguments

df	A data frame or matrix with more than 2 columns.

Value

The Cronbach's alpha value for df.

Note

For use case, see sjp.pca and sjt.pca.

See Also

sjs.reliability
sjt.itemanalysis
sjp.pca
sjt.pca

---

sjs.mic *Computes a mean inter-item-correlation.*

Description

This function calculates a mean inter-item-correlation, i.e. a correlation matrix of data will be computed (unless data is already a cor-object) and the mean of all added item's correlation values is returned. Requires either a data frame or a computed cor-object.

Usage

```
sjs.mic(data, corMethod = "pearson")
```

Arguments

data	A correlation object, built with the R-cor-function, or a data frame which correlations should be calculated.
corMethod	Indicates the correlation computation method. May be one of "spearman" (default), "pearson" or "kendall".

Value

The value of the computed mean inter-item-correlation.

See Also

sjs.cronbach
sjt.itemanalysis
sjs.reliability
sjp.pca
sjt.pca

Examples

```
# ------------------------------
# Data from the EUROFAMCARE sample dataset
# ------------------------------
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")
# create data frame with COPE-index scale
df <- as.data.frame(efc[,c(start:end)])

sjs.mic(df)
```

---

sjs.mwu *Performs a Mann-Whitney-U-Test*

Description

This function performs a Mann-Whitney-U-Test (or `Wilcoxon rank sum test`, see `wilcox.test` and `wilcox_test`) for the variable `var`, which is divided into groups indicated by `grp` (so the formula `var ~ grp` is used). If `grp` has more than two categories, a comparison between each two groups is performed.

The function reports U, p and Z-values as well as effect size r and group-rank-means.

Usage

```
sjs.mwu(var, grp, distribution = "asymptotic", weights = NULL)
```

Arguments

`var`	A numeric vector / variable, where the Mann-Whitney-U-Test should be applied to.
`grp`	The grouping variable indicating the groups that should be used for comparison.
`distribution`	indicates how the null distribution of the test statistic should be computed. Mey be one of `exact`, `approximate` or `asymptotic` (default). See `wilcox_test` for details.
`weights`	defining integer valued weights for the observations. By default, this is `NULL`.

Value

(Invisibly) returns a data frame with U, p and Z-values for each group-comparison as well as effect-size r.

Note

This function calls the wilcox_test (from the coin package) with formula. If grp has more than two groups, additionally a Kruskal-Wallis-Test (see kruskal.test) is performed.

Interpretation of effect sizes:

- small effect >= 0.1
- medium effect >= 0.3
- large effect >= 0.5

See Also

sjs.chi2.gof, sju.aov1.levene and wilcox.test, ks.test, kruskal.test, t.test, chisq.test, fisher.test

Examples

```
## Not run:
data(efc)
# Mann-Whitney-U-Tests for elder's age by elder's dependency.
sjs.mwu(efc$e17age, efc$e42dep)
## End(Not run)
```

---

sjs.phi *Phi value for a contingency table*

Description

Compute Phi value for a contingency table.

Usage

```
sjs.phi(tab)
```

Arguments

tab	A simple table or ftable. Tables of class xtabs and other will be coerced to ftable objects.

Value

The table's Phi value.

See Also

sjs.table.values
sjs.cramer

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:2, 30, TRUE))
sjs.phi(tab)
```

---

sjs.reliability *Performs a reliability test on an item scale.*

Description

This function calculates the item discriminations (corrected item-total correlations for each item of `df` with the remaining items) and the Cronbach's alpha for each item, if it was deleted from the scale.

Usage

```
sjs.reliability(df, scaleItems = FALSE, digits = 3)
```

Arguments

`df`	A data frame with items (from a scale)
`scaleItems`	If `TRUE`, the data frame's vectors will be scaled. Recommended, when the variables have different measures / scales.
`digits`	Amount of digits for Cronbach's Alpha and correlation values in returned data frame.

Value

A data frame with the corrected item-total correlations (item discrimination) and Cronbach's alpha (if item deleted) for each item of the scale, or `NULL` if data frame had too less columns.

Note

This function is similar to a basic reliability test in SPSS. The correlations in the Item-Total-Statistic are a computed correlation of each item against the sum of the remaining items (which are thus treated as one item).

See Also

`sjs.cronbach`
`sjt.itemanalysis`
`sjs.mic`
`sjp.pca`
`sjt.pca`
`sjt.df`

Examples

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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

## Not run:
sjt.df(sjs.reliability(df),
       describe=FALSE,
       showCommentRow=TRUE,
       commentString=sprintf("Cronbach's &alpha;=%.2f", sjs.cronbach(df)))
## End(Not run)

# ---------------------------------------
# Compute PCA on Cope-Index, and perform a
# reliability check on each extracted factor.
# ---------------------------------------
## Not run:
factors <- sjt.pca(df)$factor.index
findex <- sort(unique(factors))
for (i in 1:length(findex)) {
 rel.df <- subset(df, select=which(factors==findex[i]))
 if (ncol(rel.df)>=3) {
   sjt.df(sjs.reliability(rel.df),
          describe=FALSE,
          showCommentRow=TRUE,
          useViewer=FALSE,
          title="Item-Total-Statistic",
          commentString=sprintf("Scale's overall Cronbach's &alpha;=%.2f",
                                sjs.cronbach(rel.df)))
   }
 }
## End(Not run)
```

`sjs.table.values` *Compute table's values*

Description

This function calculates a table's cell, row and column percentages as well as expected values and returns all results as lists of tables.

Usage

```
sjs.table.values(tab, digits = 2)
```

Arguments

tab	A simple table or ftable of which cell, row and column percentages as well as expected values are calculated. Tables of class xtabs and other will be coerced to ftable objects.
digits	The amount of digits for the table percentage values.

Value

(invisibly) returns a list with four tables:

1. cell a table with cell percentages of tab
2. row a table with row percentages of tab
3. col a table with column percentages of tab
4. expected a table with expected values of tab

See Also

sjs.phi
sjs.cramer

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:3, 30, TRUE))
# show expected values
sjs.table.values(tab)$expected
# show cell percentages
sjs.table.values(tab)$cell
```

---

sjt.corr *Show correlations as HTML table*

Description

Shows the results of a computed correlation as HTML table. Requires either a data frame or a computed cor-object.

Usage

```
sjt.corr(data, missingDeletion = "pairwise", corMethod = "spearman",
  title = NULL, showPValues = TRUE, pvaluesAsNumbers = FALSE,
  fadeNS = TRUE, file = NULL, varlabels = NULL, breakLabelsAt = 40,
  digits = 3, triangle = "both", val.rm = NULL, stringDiagonal = NULL,
  encoding = "UTF-8", CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

data	A correlation object, built with the R-cor-function, or a data frame which correlations should be calculated.
missingDeletion	Indicates how missing values are treated. May be either "listwise" or "pairwise" (default).
corMethod	Indicates the correlation computation method. May be one of "spearman" (default), "pearson" or "kendall".
title	A table caption. By default, title is NULL, hence no title will be used.
showPValues	Whether significance levels (p-values) of correlations should be printed or not.
pvaluesAsNumbers	If TRUE, the significance levels (p-values) are printed as numbers. if FALSE (default), asterisks are used.
fadeNS	If TRUE (default), non-significant correlation-values appear faded (by using a lighter grey text color).
file	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.
varlabels	The item labels that are printed along the first column/row. If no item labels are provided (default), the data frame's column names are used. Item labels must be a string vector, e.g.: varlabels=c("Var 1", "Var 2", "Var 3"). varlabels are detected automatically if data is a data frame where each variable has a "variable.label" attribute (see sji.setVariableLabels) for details).
breakLabelsAt	Wordwrap for diagram labels. Determines how many chars of the variable labels are displayed in one line and when a line break is inserted. Default is 40.
digits	The amount of digits used the values inside table cells. Default is 2.
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.
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 that "val.rm=.4" and thus not be omitted. By default, this parameter 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 parameter ("css.valueremove") to change color and appearance of those correlation value that are smaller than the limit specified by val.rm.

stringDiagonal — 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 parameter is NULL and the diagnal cells are empty.

encoding — The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.summary='+color:blue;' adds blue font color to summary row.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (output.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 (specify parameter 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).

References

- http://rpubs.com/sjPlot/sjtcorr
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.corr

Examples

```
# create data frame with 5 random variables
df <- as.data.frame(cbind(rnorm(10), rnorm(10), rnorm(10), rnorm(10), rnorm(10)))

# plot correlation matrix using circles
## Not run:
sjt.corr(df)
## End(Not run)

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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

# we have high correlations here, because all items
# belong to one factor. See example from "sjp.pca".
## Not run:
sjt.corr(df, pvaluesAsNumbers=TRUE)
## End(Not run)

# ------------------------------
# auto-detection of labels, only lower triangle
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.corr(efc[,c(start:end)], triangle="lower")
## End(Not run)

# ------------------------------
# auto-detection of labels, only lower triangle,
# all correlation values smaller than 0.3 are not
```

```
# shown in the table
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.corr(efc[,c(start:end)], triangle="lower", val.rm=0.3)
## End(Not run)

# ------------------------------
# auto-detection of labels, only lower triangle,
# all correlation values smaller than 0.3 are printed
# in blue
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.corr(efc[,c(start:end)], triangle="lower",
         val.rm=0.3, CSS=list(css.valueremove='color:blue;'))
## End(Not run)
```

`sjt.df` *Show (description of) data frame as HTML table*

Description

Shows description or the content of data frame (rows and columns) as HTML table, or saves it as file. Helpful if you want a quick overview of a data frame's content. See parameter `describe` for details. By default, `describe` is `TRUE` and a description of the data frame is given, using the `describe` function of the `psych` package.

Usage

```
sjt.df(df, describe = TRUE, file = NULL, alternateRowColors = FALSE,
  orderColumn = NULL, orderAscending = TRUE, title = NULL,
  repeatHeader = FALSE, stringVariable = "Variable", showType = FALSE,
  showRowNames = TRUE, showCommentRow = FALSE,
  commentString = "No comment...", hideProgressBar = FALSE,
  encoding = "UTF-8", CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

`df`	A data frame that should be printed.
`describe`	If `TRUE` (default), a description of the data frame's variables is given. The description is retrieved from the `describe` function of the `psych` package. If this parameter is `FALSE`, the data frame's content (values) is shown.
`file`	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.
`alternateRowColors`	If `TRUE`, alternating rows are highlighted with a light gray background color.

orderColumn — Indicates a column, either by column name or by column index number, that should be orderd. Default order is ascending, which can be changed with orderAscending parameter. Default is NULL, hence the data frame is printed with no specific order. See examples for further details.

orderAscending — If TRUE (default) and orderColumn is not NULL, data frame is ordered according to the specified column in an ascending order. Use FALSE to apply descending order. See examples for further details.

title — A table caption. By default, title is NULL, hence no title will be used.

stringVariable — A string used for the first column name. Default is "Variable".

repeatHeader — If TRUE, the header row will also be added at the bottom at the table. This might be helpful, if you have longer tables and want to see the column names at the end of the table as well.

showType — If TRUE, the variable type is shown in a separate row below the column names.

showRowNames — If TRUE and describe is false, first table column contains row names of data frame. Use showRowNames=FALSE to omit first table column with row names.

showCommentRow — If TRUE, an optional comment line can be added to the end / below the table. Use commentString to specify the comment.

commentString — A string that will be added to the end / below the table. Only applies, if showCommentRow is TRUE.

hideProgressBar — If TRUE, the progress bar that is displayed when creating the table is hidden. Default in FALSE, hence the bar is visible.

encoding — The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.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 below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

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

for further use.

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2014/03/04/beautiful-table-outputs-in-r-part-2-rstats-sjplot

See Also

sji.viewSPSS

Examples

```
# init dataset
data(efc)

# plot efc-data frame summary
## Not run:
sjt.df(efc, alternateRowColors=TRUE)
## End(Not run)

# plot content, first 50 rows of first 5 columns of example data set
## Not run:
sjt.df(efc[1:50,1:5], describe=FALSE, stringVariable="Observation")
## End(Not run)

# plot efc-data frame summary, ordered descending by mean-column
## Not run:
sjt.df(efc, orderColumn="mean", orderAscending=FALSE)
## End(Not run)

# plot first 20 rows of first 5 columns of example data set,
# ordered by column "e42dep" with alternating row colors
## Not run:
sjt.df(efc[1:20,1:5], alternateRowColors=TRUE,
```

```
       orderColumn="e42dep", describe=FALSE)
## End(Not run)

# plot first 20 rows of first 5 columns of example data set,
# ordered by 4th column in descending order.
## Not run:
sjt.df(efc[1:20,1:5], orderColumn=4, orderAscending=FALSE, describe=FALSE)
## End(Not run)

# ------------------------------------------------------------
# User defined style sheet
# ------------------------------------------------------------
## Not run:
sjt.df(efc,
       alternateRowColor=TRUE,
       CSS=list(css.table="border: 2px solid #999999;",
                css.tdata="border-top: 1px solid;",
                css.arc="color:blue;"))
## End(Not run)
```

---

sjt.frq *Show frequencies as HTML table*

Description

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

Usage

```
sjt.frq(data, file = NULL, weightBy = NULL, variableLabels = NULL,
  valueLabels = NULL, autoGroupAt = NULL, alternateRowColors = FALSE,
  stringValue = "value", stringCount = "N", stringPerc = "raw %",
  stringValidPerc = "valid %", stringCumPerc = "cumulative %",
  stringMissingValue = "missings", highlightMedian = FALSE,
  highlightQuartiles = FALSE, skipZeroRows = FALSE, showSummary = TRUE,
  showSkew = FALSE, showKurtosis = FALSE, skewString = "&gamma;",
  kurtosisString = "&omega;", removeStringVectors = TRUE,
  autoGroupStrings = TRUE, maxStringDist = 3, encoding = "UTF-8",
  CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

data	The variables which frequencies should be printed as table. Either use a single variable (vector) or a data frame where each column represents a variable (see examples).
file	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.

weightBy A weight factor that will be applied to weight all cases from data. default is NULL, so no weights are used.

variableLabels A single character vector or a list of character vectors that indicate the variable names of those variables from data and will be used as variable labels in the output. Note that if multiple variables are supplied (as data frame), the variable labels must be supplied as list object (see examples).

valueLabels A list of character vectors that indicate the value labels of those variables from data. Note that if multiple variables are supplied (as data frame), the value labels must be supplied as nested list object (see examples).

autoGroupAt A value indicating at which length of unique values a variable from data is automatically grouped into smaller units (see sju.groupVar). Variables with large numbers of unique values may be too time consuming when a HTML table is created and R would not respond any longer. Hence it's recommended to group such variables. Default value is 50, i.e. variables with 50 and more unique values will be grouped using sju.groupVar with groupsize="auto" parameter. By default, the maximum group count is 30. However, if autoGroupAt is less than 30, autoGroupAt groups are built. Default value is NULL, i.e. auto-grouping is turned off.

alternateRowColors If TRUE, alternating rows are highlighted with a light gray background color.

stringValue String label for the very first table column containing the values (see valueLabels).

stringCount String label for the first table data column containing the counts. Default is "N".

stringPerc String label for the second table data column containing the percentages, where the count percentages include missing values.

stringValidPerc String label for the third data table column containing the valid percentages, i.e. the count percentage value exluding possible missing values.

stringCumPerc String label for the last table data column containing the cumulative percentages.

stringMissingValue String label for the last table data row containing missing values.

highlightMedian If TRUE, the table row indicating the median value will be highlighted.

highlightQuartiles If TRUE, the table row indicating the lower and upper quartiles will be highlighted.

skipZeroRows If TRUE, rows with only zero-values are not printed. Default is FALSE.

showSummary If TRUE (default), a summary row with total and valid N as well as mean and standard deviation is shown.

showSkew If TRUE, the variable's skewness is added to the summary. The skewness is retrieved from the describe function of the psych package.

showKurtosis If TRUE, the variable's kurtosis is added to the summary. The kurtosis is retrieved from the describe function of the psych package.

skewString A character string, which is used as header for the skew column (see showSkew)). Default is lower case Greek gamma.

kurtosisString A character string, which is used as header for the kurtosis column (see showKurtosis)). Default is lower case Greek omega.

removeStringVectors
If TRUE (default), character vectors / string variables will be removed from data before frequency tables are computed.

autoGroupStrings
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 maxStringDist parameter. This parameter only applies if removeStringVectors is FALSE.

maxStringDist 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.

encoding The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.caption='+color:red;' to add red font-color to the default table caption style.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- each frequency table as web page content (page.content.list),

- the complete html-output (`output.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 (specify parameter `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`).

References

- http://rpubs.com/sjPlot/sjtfrq
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.frq
sjt.xtab

Examples

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

# retrieve value and variable labels
variables <- sji.getVariableLabels(efc)
values <- sji.getValueLabels(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,
        variableLabels=variables['e42dep'],
        valueLabels=values[['e42dep']])

# plot frequencies of e42dep, e16sex and c172code in one HTML file
# and show table in RStudio Viewer Pane or default web browser
sjt.frq(as.data.frame(cbind(efc$e42dep, efc$e16sex, efc$c172code)),
       variableLabels=list(variables['e42dep'], variables['e16sex'], variables['c172code']),
        valueLabels=list(values[['e42dep']], values[['e16sex']], values[['c172code']]))

# plot larger scale including zero-counts
# indicating median and quartiles
sjt.frq(efc$neg_c_7,
        variableLabels=variables['neg_c_7'],
```

```
        valueLabels=values[['neg_c_7']],
        highlightMedian=TRUE,
        highlightQuartiles=TRUE)

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

# --------------------------------
# User defined style sheet
# --------------------------------
sjt.frq(efc$e42dep,
        variableLabels=variables['e42dep'],
        valueLabels=values[['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` *Show (and compare) generalized linear models as HTML table*

Description

Shows (and compares multiple) generalized linear models (Odds Ratios) as HTML table, or saves them as file. The fitted glm's should have the same predictor variables and either

- differ only in their response (dependent variable), to see the effect of a specific set of predictors on different responses, or
- all have the same reponse variables, but differ in their `family` objects and link function in order to see which model fits best to the data.

See parameter `showFamily` for details and section `examples`.

Usage

```
sjt.glm(..., file = NULL, labelPredictors = NULL,
  labelDependentVariables = NULL, stringPredictors = "Predictors",
  stringDependentVariables = "Dependent Variables",
  showHeaderStrings = FALSE, stringModel = "Model",
  stringIntercept = "(Intercept)", stringObservations = "Observations",
  stringOR = "OR", stringCI = "CI", stringSE = "std. Error",
  stringP = "p", digits.est = 2, digits.p = 3, digits.ci = 2,
  digits.se = 2, digits.summary = 3, exp.coef = TRUE,
  pvaluesAsNumbers = FALSE, boldpvalues = TRUE, showConfInt = TRUE,
  showStdError = FALSE, separateConfColumn = FALSE, newLineConf = TRUE,
```

```
  showAbbrHeadline = TRUE, showPseudoR = TRUE, showLogLik = FALSE,
  showAIC = FALSE, showChi2 = FALSE, showFamily = FALSE,
  cellSpacing = 0.2, encoding = "UTF-8", CSS = NULL, useViewer = TRUE,
  no.output = FALSE)
```

Arguments

...	One or more fitted glm-objects.
file	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.
labelPredictors	Labels of the predictor variables, provided as char vector.
labelDependentVariables	Labels of the dependent variables of all fitted models which have been used as first parameter(s), provided as char vector.
stringPredictors	String constant used as headline for the predictor column. Default is "Predictors".
stringDependentVariables	String constant used as headline for the dependent variable columns. Default is "Dependent Variables".
showHeaderStrings	If TRUE, the header strings stringPredictors and stringDependentVariables are shown. By default, they're hidden.
stringModel	String constant used as headline for the model names in case no labels for the dependent variables are provided (see labelDependentVariables). Default is "Model".
stringIntercept	String constant used as headline for the Intercept row default is "Intercept".
stringObservations	String constant used in the summary row for the count of observation (cases). Default is "Observations".
stringOR	String used for the column heading of odds ratio values. Default is "OR".
stringCI	String used for the column heading of confidence interval values. Default is "CI".
stringSE	String used for the column heading of standard error values. Default is "std. Error".
stringP	String used for the column heading of p values. Default is "p".
digits.est	Amount of decimals for estimators.
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.
exp.coef	If TRUE (default), regression coefficients and confidence intervals are exponentiated (exp(coef(fit)). Use FALSE if you want the non-exponentiated coefficients as they are provided by the summary function.

pvaluesAsNumbers
: If TRUE, p-values are shown as numbers. If FALSE (default), p-values are indicated by asterisks.

boldpvalues
: If TRUE (default), significant p-values are shown bold faced.

showConfInt
: If TRUE (default), the confidence intervall is also printed to the table. Use FALSE to omit the CI in the table.

showStdError
: If TRUE, the standard errors are also printed. Default is FALSE.

separateConfColumn
: if TRUE, the CI values are shown in a separate table column. Default is FALSE.

newLineConf
: If TRUE and separateConfColumn is FALSE, inserts a line break between OR and CI values. If FALSE, CI values are printed in the same line with OR values.

showAbbrHeadline
: If TRUE (default), the table data columns have a headline with abbreviations for odds ratios, confidence interval and p-values.

showPseudoR
: If TRUE (default), the pseudo R2 values for each model are printed in the model summary. R2cs is the Cox-Snell-pseudo R-square value, R2n is Nagelkerke's pseudo R-square value.

showLogLik
: If TRUE, the Log-Likelihood for each model is printed in the model summary. Default is FALSE.

showAIC
: If TRUE, the AIC value for each model is printed in the model summary. Default is FALSE.

showChi2
: If TRUE, the chi-square value for each model is printed in the model summary. Default is FALSE.

showFamily
: 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 showAIC) to get a decision help for which model to choose.

cellSpacing
: The inner padding of table cells. By default, this value is 0.2 (measure 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 parameter for the CSS parameter for changing cell spacing, which would be: CSS=list(css.thead="padding:0.2cm;", css.tzdata="padding:0.2cm;").

encoding
: The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

  1. the class-names with "css."-prefix as parameter 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 parameter 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.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

`useViewer` If `TRUE`, the function tries to show the HTML table in the IDE's viewer pane. If `FALSE` or no viewer available, the HTML table is opened in a web browser.

`no.output` 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.

Value

Invisibly returns a `structure` with

- the web page style sheet (`page.style`),
- the web page content (`page.content`),
- the complete html-output (`output.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 (specify parameter `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`).

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2013/08/20/print-glm-output-to-html-table-rstats/

See Also

`sjt.lm`
`sjp.glm`

Examples

```
# prepare dummy variables for binary logistic regression
y1 <- ifelse(swiss$Fertility<median(swiss$Fertility), 0, 1)
y2 <- ifelse(swiss$Infant.Mortality<median(swiss$Infant.Mortality), 0, 1)
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 ~ swiss$Education+swiss$Examination+swiss$Catholic,
              family=binomial(link="logit"))
fitOR2 <- glm(y2 ~ swiss$Education+swiss$Examination+swiss$Catholic,
              family=binomial(link="logit"))
fitOR3 <- glm(y3 ~ swiss$Education+swiss$Examination+swiss$Catholic,
              family=binomial(link="logit"))

# open HTML-table in RStudio Viewer Pane or web browser
## Not run:
sjt.glm(fitOR1, fitOR2, labelDependentVariables=c("Fertility", "Infant Mortality"),
        labelPredictors=c("Education", "Examination", "Catholic"))
## End(Not run)

# open HTML-table in RStudio Viewer Pane or web browser,
# table indicating p-values as numbers
## Not run:
sjt.glm(fitOR1, fitOR2, labelDependentVariables=c("Fertility", "Infant Mortality"),
        labelPredictors=c("Education", "Examination", "Catholic"),
        pvaluesAsNumbers=TRUE)
## End(Not run)

# open HTML-table in RStudio Viewer Pane or web browser,
# printing CI in a separate column
## Not run:
sjt.glm(fitOR1, fitOR2, fitOR3,
        labelDependentVariables=c("Fertility", "Infant Mortality", "Agriculture"),
        labelPredictors=c("Education", "Examination", "Catholic"),
        separateConfColumn=TRUE)
## End(Not run)

# open HTML-table in RStudio Viewer Pane or web browser,
# indicating p-values as numbers and printing CI in a separate column
## Not run:
sjt.glm(fitOR1, fitOR2, fitOR3,
        labelDependentVariables=c("Fertility", "Infant Mortality", "Agriculture"),
        labelPredictors=c("Education", "Examination", "Catholic"),
        pvaluesAsNumbers=TRUE, separateConfColumn=TRUE)
## End(Not run)

# ------------------------------------------------------------------
# User defined style sheet
# ------------------------------------------------------------------
## Not run:
sjt.glm(fitOR1, fitOR2, fitOR3,
```

```
        labelDependentVariables=c("Fertility", "Infant Mortality", "Agriculture"),
        labelPredictors=c("Education", "Examination", "Catholic"),
        CSS=list(css.table="border: 2px solid;",
                 css.tdata="border: 1px solid;",
                 css.depvarhead="color:#003399;"))
## End(Not run)

# ------------------------------------------------------------
# Compare models with different link functions, but same
# predictors and response
# ------------------------------------------------------------
# load efc sample data
data(efc)
# dichtomozize service usage by "service usage yes/no"
efc$services <- sju.dicho(efc$tot_sc_e, "v", 0)
# 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
## Not run:
sjt.glm(fit1, fit2, fit3, showAIC=TRUE, showFamily=TRUE, showPseudoR=FALSE)
## End(Not run)
```

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

Description

This function performs an item analysis with certain statistics that are useful for scale / index development. The resulting tables are shown in the viewer pane / 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",
  scaleItems = FALSE, minValidRowMeanValue = 2, alternateRowColors = TRUE,
  orderColumn = NULL, orderAscending = TRUE, showShapiro = FALSE,
  showKurtosis = FALSE, showCompCorrMat = TRUE, file = NULL,
  encoding = "UTF-8", CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

df	A data frame with items (from a scale)
factor.groups	If not NULL, the data frame 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 for more details.
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.
scaleItems	If TRUE, the data frame's vectors will be scaled when calculating the Cronbach's Alpha value (see sjs.reliability). Recommended, when the variables have different measures / scales.
minValidRowMeanValue	the 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 minValidRowMeanValue per case (observation, or technically, row). See sju.mean.n for details.
alternateRowColors	If TRUE, alternating rows are highlighted with a light gray background color.
orderColumn	Indicates a column, either by column name or by column index number, that should be orderd. Default order is ascending, which can be changed with orderAscending parameter. Default is NULL, hence the results of the item analysis are printed with no specific order. See examples in sjt.df for further details.
orderAscending	If TRUE (default) and orderColumn is not NULL, data frame is ordered according to the specified column in an ascending order. Use FALSE to apply descending order. See examples in sjt.df for further details.
showShapiro	If TRUE, a Shapiro-Wilk normality test is computed for each item. See shapiro.test for details.
showKurtosis	If TRUE, the kurtosis for each item will also be shown (see kurtosi and describe in the psych-package for more details.
showCompCorrMat	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.

`file` The destination file, which will be in html-format. If no filepath is specified (default), the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.

`encoding` The charset encoding used for variable and value labels. Default is `"UTF-8"`. 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 (see http://www.w3.org/Style/CSS/). See return value `page.style` for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with `"css."`-prefix as parameter 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 parameter 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.arc='color:blue;'` for a blue text color each 2nd row.
- `css.arc='+font-style:italic;'` to add italic formatting to each 2nd row.

See further examples at http://rpubs.com/sjPlot/sjtbasics.

`useViewer` If `TRUE`, the function tries to show the HTML table in the IDE's viewer pane. If `FALSE` or no viewer available, the HTML table is opened in a web browser.

`no.output` 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.

Value

(Invisibly) returns a structure with following elements:

- `df.list`: List of data frames with the item analysis for each sub.group (or complete, if `factor.groups` was `NULL`)
- `index.scores`: List of standardized scale / index scores of each case (mean value of all scale items for each case) for each sub-group. Note that NA's are removed from this list. Use `df.index.scores` if you want to append the cases' related index scores to the original data frame.
- `df.index.scores`: A data frame with all `index.scores` as column variables. While `index.scores` don't have NA's included, this data frame's row-length equals to the originals data frame's row-length (and thus can be appended)

- `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` was `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 `sjs.reliability` 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.

References

- Jorion N, Self B, James K, Schroeder L, DiBello L, Pellegrino J (2013) Classical Test Theory Analysis of the Dynamics Concept Inventory. (https://www.academia.edu/4104752/Classical_Test_Theory_Analysis_of_the_Dynamics_Concept_Inventory)
- 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 (http://onlinelibrary.wiley.com/doi/10.1111/j.1467-6494.1986.tb00391.x/abstract)
- 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. http://www.socialresearchmethods.net/kb/reltypes.php

See Also

`sjs.cronbach`
`sjs.reliability`
`sjs.mic`
`sjp.pca`
`sjt.pca`
`sjt.df`
`sju.mean.n`

Examples

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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

## Not run:
sjt.itemanalysis(df)
## End(Not run)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.itemanalysis(efc[,c(start:end)])
## End(Not run)

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

---

sjt.lm *Show linear regression as HTML table*

Description

Shows (multiple) fitted linear models (beta coefficients, std. beta values etc.) as HTML table, or saves them as file. The fitted lm's should have the same predictor variables and differ only in their response (dependent variable).

Usage

```
sjt.lm(..., file = NULL, labelPredictors = NULL,
  labelDependentVariables = NULL, stringPredictors = "Predictors",
  stringDependentVariables = "Dependent Variables", stringModel = "Model",
  showHeaderStrings = FALSE, stringIntercept = "(Intercept)",
  stringObservations = "Observations", stringB = "B",
  stringSB = "std. Beta", stringCI = "CI", stringSE = "std. Error",
  stringP = "p", showConfInt = TRUE, showStdBeta = FALSE,
  showStdError = FALSE, digits.est = 2, digits.p = 3, digits.ci = 2,
  digits.se = 2, digits.sb = 2, digits.summary = 3,
  pvaluesAsNumbers = FALSE, boldpvalues = TRUE,
  separateConfColumn = FALSE, newLineConf = TRUE, showAbbrHeadline = TRUE,
  showR2 = TRUE, showFStat = FALSE, showAIC = FALSE, cellSpacing = 0.2,
  encoding = "UTF-8", CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

`...`	One or more fitted lm-objects.
`file`	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.
`labelPredictors`	Labels of the predictor variables, provided as char vector.
`labelDependentVariables`	Labels of the dependent variables of all fitted models which have been used as first parameter(s), provided as char vector.
`stringPredictors`	String constant used as headline for the predictor column. Default is `"Predictors"`.
`stringDependentVariables`	String constant used as headline for the dependent variable columns. Default is `"Dependent Variables"`.
`showHeaderStrings`	If `TRUE`, the header strings `stringPredictors` and `stringDependentVariables` are shown. By default, they're hidden.
`stringModel`	String constant used as headline for the model names in case no labels for the dependent variables are provided (see labelDependentVariables). Default is `"Model"`.
`stringIntercept`	String constant used as headline for the Intercept row default is `"Intercept"`.
`stringObservations`	String constant used in the summary row for the count of observation (cases). Default is `"Observations"`.
`stringB`	String used for the column heading of beta coefficients. Default is `"B"`.
`stringSB`	String used for the column heading of standardized beta coefficients. Default is `"std. Beta"`.

stringCI String used for the column heading of confidence interval values. Default is "CI".

stringSE String used for the column heading of standard error values. Default is "std. Error".

stringP String used for the column heading of p values. Default is "p".

showConfInt If TRUE (default), the confidence intervall is also printed to the table. Use FALSE to omit the CI in the table.

showStdBeta If TRUE, the standardized beta-coefficients are also printed. Default is FALSE.

showStdError If TRUE, the standard errors are also printed. Default is FALSE.

digits.est Amount of decimals for estimators.

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.sb Amount of decimals for standardized beta.

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

pvaluesAsNumbers
If TRUE, p-values are shown as numbers. If FALSE (default), p-values are indicated by asterisks.

boldpvalues If TRUE (default), significant p-values are shown bold faced.

separateConfColumn
if TRUE, the CI values are shown in a separate table column. Default is FALSE.

newLineConf If TRUE and separateConfColumn is FALSE, inserts a line break between B and CI values. If FALSE, CI values are printed in the same line with B values.

showAbbrHeadline
If TRUE (default), the table data columns have a headline with abbreviations for beta- and std. beta-values, confidence interval and p-values.

showR2 If TRUE (default), the R2 and adjusted R2 values for each model are printed in the model summary.

showFStat If TRUE, the F-statistics for each model is printed in the model summary. Default is FALSE.

showAIC If TRUE, the AIC value for each model is printed in the model summary. Default is FALSE.

cellSpacing The inner padding of table cells. By default, this value is 0.2 (measure 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 parameter for the CSS parameter for changing cell spacing, which would be: CSS=list(css.thead="padding:0.2cm;", css.tzdata="padding:0.2cm;").

encoding The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (output.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 (specify parameter 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).

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2013/08/20/print-glm-output-to-html-table-rstats/

See Also

sjt.glm
sjp.lm

Examples

```
# Now fit the models. Note that both models share the same predictors
# and only differ in their dependent variable
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
## Not run:
sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
      labelPredictors=c("Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"))
## End(Not run)

# show HTML-table, indicating p-values as numbers
## Not run:
sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
      labelPredictors=c("Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"),
       showStdBeta=TRUE, pvaluesAsNumbers=TRUE)
## End(Not run)

# create and open HTML-table in RStudio Viewer Pane or web browser,
# printing CI in a separate column
## Not run:
sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
      labelPredictors=c("Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"),
       separateConfColumn=TRUE)
## End(Not run)

# show HTML-table, indicating p-values as numbers
# and printing CI in a separate column
## Not run:
sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
      labelPredictors=c("Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"),
       showStdBeta=TRUE, pvaluesAsNumbers=TRUE, separateConfColumn=TRUE)
## End(Not run)

# ------------------------------------------------------------
# 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)
## Not run:
# create and save first HTML-table
part1 <- sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
                labelPredictors=c("Carer's Age", "Hours of Care",
                                  "Carer's Sex", "Educational Status"))
# create and save second HTML-table
part2 <- sjt.lm(fit3, fit4, labelDependentVariables=c("Service Usage", "Elder's Dependency"),
```

```
                labelPredictors=c("Carer's Age", "Hours of Care",
                                  "Carer's Sex", "Educational Status"))
# 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)
## End(Not run)

# ------------------------------------------------------------
# User defined style sheet
# ------------------------------------------------------------
## Not run:
sjt.lm(fit1, fit2, labelDependentVariables=c("Barthel-Index", "Negative Impact"),
      labelPredictors=c("Carer's Age", "Hours of Care", "Carer's Sex", "Educational Status"),
       CSS=list(css.table="border: 2px solid;",
                css.tdata="border: 1px solid;",
                css.depvarhead="color:#003399;"))
## End(Not run)
```

`sjt.pca` *Show principal component analysis as HTML table*

Description

Performes a principle component analysis on a data frame or matrix 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, numberOfFactors = NULL, factorLoadingTolerance = 0.1,
  file = NULL, varlabels = NULL,
  title = "Principal Component Analysis (with varimax rotation)",
  breakLabelsAt = 40, digits = 2, showCronbachsAlpha = TRUE,
  showMSA = FALSE, showVariance = FALSE, alternateRowColors = FALSE,
  stringPov = "Proportion of Variance",
  stringCpov = "Cumulative Proportion", encoding = "UTF-8", CSS = NULL,
  useViewer = TRUE, no.output = FALSE)
```

Arguments

`data`	A data frame with factors (each columns one variable) that should be used to compute a PCA, or a `prcomp` object.

numberOfFactors
: A predefined number of factors to use for the calculating the varimax rotation. By default, this value is NULL and the amount of factors is calculated according to the Kaiser-criteria. See paramater plotEigenvalues.

factorLoadingTolerance
: 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

file
: The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.

varlabels
: The item labels that are printed in the first column. If no item labels are provided (default), the data frame's column names are used. Item labels must be a string vector, e.g.: varlabels=c("Var 1", "Var 2", "Var 3").

title
: A table caption. By default, "Principal Component Analysis (with varimax rotation)" is used as the table's title.

breakLabelsAt
: Wordwrap for diagram labels. Determines how many chars of the variable labels are displayed in one line and when a line break is inserted. Default is 20.

digits
: The amount of digits used the values inside table cells. Default is 2.

showCronbachsAlpha
: 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 and no prcomp object.

showMSA
: If TRUE, shows an additional column with the measure of sampling adequacy according dor each component.

showVariance
: If TRUE, the proportions of variances for each component as well as cumulative variance are shown in the table footer.

alternateRowColors
: If TRUE, alternating rows are highlighted with a light gray background color.

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

stringCpov
: The string for the table row that contains the cumulative variances. By default, "Cumulative Proportion" will be used.

encoding
: The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.cronbach='+color:green;' to add green color formatting to the Cronbach's Alpha value.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (output.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 factorLoadingTolerance's range.

for further use.

Note

The HTML tables can either be saved as file and manually opened (specify parameter 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).

This PCA uses the prcomp function and the varimax rotation.

References

- http://strengejacke.wordpress.com/sjplot-r-package/
- http://strengejacke.wordpress.com/2014/03/04/beautiful-table-outputs-in-r-part-2-rstats-sjplot

See Also

sjp.pca
sjs.reliability
sjt.itemanalysis
sjs.cronbach

Examples

```
# randomly create data frame with 7 items, each consisting of 4 categories
likert_4 <- data.frame(sample(1:4, 500, replace=TRUE, prob=c(0.2,0.3,0.1,0.4)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.5,0.25,0.15,0.1)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.4,0.15,0.25,0.2)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.25,0.1,0.4,0.25)),
                       sample(1:4, 500, replace=TRUE, prob=c(0.1,0.4,0.4,0.1)),
                       sample(1:4, 500, replace=TRUE),
                       sample(1:4, 500, replace=TRUE, prob=c(0.35,0.25,0.15,0.25)))

# Create variable labels
colnames(likert_4) <- c("V1", "V2", "V3", "V4", "V5", "V6", "V7")

# show table
## Not run:
sjt.pca(likert_4)
## End(Not run)

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

# retrieve variable and value labels
varlabs <- sji.getVariableLabels(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
df <- as.data.frame(efc[,c(start:end)])
colnames(df) <- varlabs[c(start:end)]

## Not run:
sjt.pca(df)
## End(Not run)
```

```
# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.pca(efc[,c(start:end)])
## End(Not run)
```

`sjt.stackfrq` *Show 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, weightBy = NULL, title = NULL, varlabels = NULL,
  breakLabelsAt = 40, valuelabels = NULL, breakValueLabelsAt = 20,
  orderBy = NULL, reverseOrder = FALSE, alternateRowColors = FALSE,
  showN = FALSE, showTotalN = FALSE, showNA = FALSE, labelNA = "NA",
  showSkew = FALSE, showKurtosis = FALSE, skewString = "Skew",
  kurtosisString = "Kurtosis", file = NULL, encoding = "UTF-8",
  CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

`items`	A `data.frame` with each column representing one (likert- or scale-)item.
`weightBy`	A weight factor that will be applied to weight all cases from `items`.
`title`	A table caption.
`varlabels`	A list or vector of strings with variable names. If not specified, row names of `items` will be used, resp. variable labels will automatically be detected, when they have a `"variable.lable"` attribute (see `sji.setVariableLabels`) for details).
`breakLabelsAt`	Wordwrap for variable labels. Determines how many chars of the variable labels are displayed in one line and when a line break is inserted. Default is 40.
`valuelabels`	A list or vector of strings that category/value labels, which appear in the header row.
`breakValueLabelsAt`	Wordwrap for value labels. Determines how many chars of the value labels are displayed in one line and when a line break is inserted. Default is 20.

orderBy — Indicates whether the items should be ordered by highest count of first or last category of items. Use "first" to order ascending by lowest count of first category, "last" to order ascending by lowest count of last category or NULL (default) for no sorting. You can specify just the initial letter. In case you want to revers order (descending from highest count), use reverseOrder parameter.

reverseOrder — If TRUE, the item order is reversed.

showN — If TRUE, each item's category N is printed in the table cells.

showTotalN — If TRUE, an additional column with each item's total N is printed.

showNA — If TRUE, NA's (missing values) are also printed in the table.

labelNA — The label for the missing column/row.

showSkew — If TRUE, an additional column with each item's skewness is printed. The skewness is retrieved from the describe function of the psych package.

showKurtosis — If TRUE, an additional column with each item's kurtosis is printed. The kurtosis is retrieved from the describe function of the psych package.

skewString — A character string, which is used as header for the skew column (see showSkew)). Default is "Skew".

kurtosisString — A character string, which is used as header for the kurtosis column (see showKurtosis)). Default is "Kurtosis".

alternateRowColors — If TRUE, alternating rows are highlighted with a light gray background color.

file — The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.

encoding — The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.caption='border-bottom: 1px dotted blue;' for a blue dotted border of the last table row.
- css.caption='+color:red;' to add red font-color to the default table caption style.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (output.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 (specify parameter 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).

References

- http://rpubs.com/sjPlot/sjtstackfrq
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

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, valuelabels=levels_4, varlabels=items)
## End(Not run)


# ------------------------------
# Data from the EUROFAMCARE sample dataset
# ------------------------------
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")
# retrieve variable labels
varlabs <- sji.getVariableLabels(efc)

# Note: Parameter "valuelabels" is only needed for datasets
# that have been imported from SPSS.
## Not run:
sjt.stackfrq(efc[,c(start:end)],
             varlabels=varlabs[c(start:end)],
             alternateRowColors=TRUE)
## End(Not run)

## Not run:
sjt.stackfrq(efc[,c(start:end)],
             varlabels=varlabs[c(start:end)],
             alternateRowColors=TRUE,
             showN=TRUE,
             showNA=TRUE)
## End(Not run)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, varlabs)
## Not run:
sjt.stackfrq(efc[,c(start:end)])
## End(Not run)

# ------------------------------
# User defined style sheet
# ------------------------------
## Not run:
sjt.stackfrq(efc[,c(start:end)],
             varlabels=varlabs[c(start:end)],
             alternateRowColors=TRUE,
             showTotalN=TRUE,
             showSkew=TRUE,
             showKurtosis=TRUE,
```

```
            CSS=list(css.ncol="border-left:1px dotted black;",
                     css.summary="font-style:italic;"))
## End(Not run)
```

---

sjt.xtab *Show 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, var.grp = NULL, weightBy = NULL, digits = 1,
  file = NULL, variableLabels = NULL, valueLabels = NULL,
  breakVariableLabelsAt = 40, breakValueLabelsAt = 20,
  stringTotal = "Total", showCellPerc = FALSE, showRowPerc = FALSE,
  showColPerc = FALSE, showExpected = FALSE, showHorizontalLine = FALSE,
  showSummary = TRUE, showLegend = TRUE, showNA = FALSE, labelNA = "NA",
  tdcol.n = "black", tdcol.expected = "#339999", tdcol.cell = "#993333",
  tdcol.row = "#333399", tdcol.col = "#339933", highlightTotal = FALSE,
  highlightColor = "#f8f8f8", percSign = " &#37;", hundret = "100.0",
  encoding = "UTF-8", CSS = NULL, useViewer = TRUE, no.output = FALSE)
```

Arguments

var.row	Variable that should be displayed in the table rows.
var.col	Variable that should be displayed in the table columns.
var.grp	An optional grouping variable that splits the data into several groups, depending on the amount of categories. See examples for details.
weightBy	A weight factor that will be applied to weight all cases. Default is NULL, so no weights are used.
digits	The amount of digits used for the percentage values inside table cells. Default is 1.
file	The destination file, which will be in html-format. If no filepath is specified, the file will be saved as temporary file and openend either in the RStudio View pane or in the default web browser.
variableLabels	A character vector of same length as supplied variables, with the associated variable names. Following order is needed: name of var.row, name of var.col, and - if var.grp is not NULL - name of var.grp. See examples for more details. variableLabels are detected automatically, if var.row or var.col have a "variable.label" attribute (see sji.setVariableLabels) for details).
valueLabels	A list of character vectors that indicate the value labels of the supplied variables. Following order is needed: value labels of var.row, value labels of var.col, and - if var.grp is not NULL - value labels of var.grp. valueLabels needs to be a list object. See examples for more details.

breakVariableLabelsAt
: Wordwrap for variable labels. Determines how many chars of the variable labels are displayed in one line and when a line break is inserted. Default is 40.

breakValueLabelsAt
: Wordwrap for value labels. Determines how many chars of the value labels are displayed in one line and when a line break is inserted. Default is 20.

stringTotal
: String label for the total column / row header.

showCellPerc
: If TRUE, cell percentage values are shown.

showRowPerc
: If TRUE, row percentage values are shown.

showColPerc
: If TRUE, column percentage values are shown.

showExpected
: If TRUE, expected values are also shown.

showHorizontalLine
: If TRUE, data rows are separated with a horizontal line.

showSummary
: If TRUE (default), a summary row with Chi-square 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-square test. If the table's matrix is larger than 2x2, Fisher's excact test with Monte Carlo simulation is computed.

showLegend
: If TRUE (default), the color legend for coloring observed and expected values as well as cell, row and column percentages is shown. See tdcol.n, tdcol.expected, tdcol.cell, tdcol.row and tdcol.col.

showNA
: If TRUE, NA's (missing values) are also printed in the table.

labelNA
: The label for the missing column/row.

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.

highlightTotal
: If TRUE, the total column and row will be highlighted with a different background color. See highlightColor.

highlightColor
: If highlightTotal is TRUE, this color value will be used for painting the background of the total column and row. Default is a light grey.

percSign
: 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".

encoding
: The charset encoding used for variable and value labels. Default is "UTF-8". 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 (see http://www.w3.org/Style/CSS/). See return value page.style for details of all style-sheet-classnames that are used in this function. Parameters for this list need:

1. the class-names with "css."-prefix as parameter 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 parameter 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.summary='+color:blue;' to add blue font color style to the summary row.

See further examples below and http://rpubs.com/sjPlot/sjtbasics.

useViewer — If TRUE, the function tries to show the HTML table in the IDE's viewer pane. If FALSE or no viewer available, the HTML table is opened in a web browser.

no.output — 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.

Value

Invisibly returns a structure with

- the web page style sheet (page.style),
- the web page content (page.content),
- the complete html-output (output.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 (specify parameter 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).

Since package version 1.3, the parameter valueLabels, which represent the value labels, is retrieved automatically if a) the variables var.col and var.row come from a data frame that was imported with the sji.SPSS function (because then value labels are attached as attributes to the data) or b) when the variables are factors with named factor levels (e.g., see column group in dataset PlantGrowth). However, you still can use own parameters variable labels.

References

- http://rpubs.com/sjPlot/sjtxtab
- http://strengejacke.wordpress.com/sjplot-r-package/

See Also

sjp.xtab
sjs.table.values

Examples

```
# prepare sample data set
data(efc)
efc.labels <- sji.getValueLabels(efc)

# print simple cross table w/o labels
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep)
## End(Not run)

# print cross table with labels and expected values
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "Elder's dependency"),
         valueLabels=list(efc.labels[['e16sex']], efc.labels[['e42dep']]),
         showExpected=TRUE)
## End(Not run)

# print minimal cross table with labels, total col/row highlighted
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "Elder's dependency"),
         valueLabels=list(efc.labels[['e16sex']], efc.labels[['e42dep']]),
         showHorizontalLine=FALSE,
         showCellPerc=FALSE,
         highlightTotal=TRUE)
## End(Not run)

# ------------------------------
# auto-detection of labels
# ------------------------------
efc <- sji.setVariableLabels(efc, sji.getVariableLabels(efc))
# print cross table with labels and all percentages
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep,
         showRowPerc=TRUE, showColPerc=TRUE)
## End(Not run)

# print cross table with labels and all percentages, including
# grouping variable
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep, efc$c161sex,
         variableLabels=c("Elder's gender",
                          "Elder's dependency",
                          "Carer's gender"),
         valueLabels=list(efc.labels[['e16sex']],
                          efc.labels[['e42dep']],
```

```
                          efc.labels[['c161sex']]),
        showRowPerc=TRUE, showColPerc=TRUE)
## End(Not run)

# ------------------------------------------------------------
# User defined style sheet
# ------------------------------------------------------------
## Not run:
sjt.xtab(efc$e16sex, efc$e42dep,
         variableLabels=c("Elder's gender", "Elder's dependency"),
         valueLabels=list(efc.labels[['e16sex']], efc.labels[['e42dep']]),
         CSS=list(css.table="border: 2px solid;",
                  css.tdata="border: 1px solid;",
                  css.horline="border-bottom: double blue;"))
## End(Not run)
```

`sju.adjustPlotRange.y` *Adjust y range of ggplot-objects*

Description

This method adjusts the y-range of a ggplot-object, which is useful when value labels are outside of the plot region. A modified ggplot-object will be returned with adjusted y-range so everything should be visible. Note that this function only works on `scale_y_continuous`.

Usage

```
sju.adjustPlotRange.y(gp, upperMargin = 1.05)
```

Arguments

`gp`	A ggplot-object. Usually, this will be returned by most of this package's plotting functions.
`upperMargin`	Defines the margin of the upper y bound of the plot. This value will be multiplied with the total y range. Default is 1.05, which means that the upper margin of the plot is about 5 percent of the "visible" plot area (i.e. the y-range is 105 percent of the actual needed range to make all object visible).

Value

The same ggplot-object, with adjusted y-range, so all graphics and labels should be visible.

Note

Note that this function only works on `scale_y_continuous`.

References

http://www.r-bloggers.com/setting-axis-limits-on-ggplot-charts/

Examples

```
# sample data set
data(efc)
# show frequencies of relationship-variable and
# retrieve plot object
gp <- sjp.frq(efc$e15relat, printPlot=FALSE)
# show current plot
plot(gp$plot)
# show adjusted plot
sju.adjustPlotRange.y(gp$plot)
```

---

sju.aov1.levene *Plot Levene-Test for One-Way-Anova*

Description

Plot results of Levene's Test for Equality of Variances for One-Way-Anova.

Usage

```
sju.aov1.levene(depVar, grpVar)
```

Arguments

depVar	The dependent variable. Will be used with following formular: `aov(depVar ~ grpVar)`
grpVar	The grouping variable, as unordered factor. Will be used with following formular: `aov(depVar ~ grpVar)`

See Also

sjp.aov1, sjs.chi2.gof, sjs.mwu and wilcox.test, ks.test, kruskal.test, t.test, chisq.test, fisher.test

Examples

```
data(efc)
sju.aov1.levene(efc$c12hour, efc$e42dep)
```

sju.dicho *Dichotomize variables*

Description

Dichotomizes variables into dummy variables (0/1). Dichotomization is either done by median or mean (see dichBy).

Usage

```
sju.dicho(var, dichBy = "median", dichVal = -1)
```

Arguments

var	The variable that should be dichotomized.
dichBy	Indicates the split criterion where the variable is dichotomized. By default, var is split into two groups at the median (dichBy="median" or dichBy="md"). Further values for dichBy are "mean" (or "m"), which splits into groups at the mean of var; and "value" (or "v"). In the latter case, you have to specifiy dichVal.
dichVal	Indicates a value where var is dichotomized when dichBy="value". Note that dichVal is inclusive, i.e. dichVal=10 will split var into one group with values from lowest to 10 and another group with values greater than 10.

Value

A dichotomized variable (0/1-coded).

Examples

```
data(efc)
summary(efc$c12hour)
table(sju.dicho(efc$c12hour))
table(sju.dicho(efc$c12hour, "mean"))
table(sju.dicho(efc$c12hour, "value", 30))
```

sju.groupString *Group near elements of string vectors*

Description

This function groups elements of a string vector (character or string variable) according to the element's distance. The more similar two string elements are, the higher is the chance to be combined into a group.

Usage

```
sju.groupString(strings, maxdist = 3, method = "lv", strict = FALSE,
  trim.whitespace = TRUE, remove.empty = TRUE, showProgressBar = FALSE)
```

Arguments

strings	a character vector with string elements
maxdist	the maximum distance between two string elements, which is allowed to treat two elements as similar or equal.
method	Method for distance calculation. The default is "lv". See stringdist package for details.
strict	if TRUE, value matching is more strictly. See examples for details.
trim.whitespace	if TRUE (default), leading and trailing white spaces will be removed from string values.
remove.empty	if TRUE (default), empty string values will be removed from the character vector strings.
showProgressBar	If TRUE, the progress bar is displayed when computing the distance matrix. Default in FALSE, hence the bar is hidden.

Value

A character vector where similar string elements (values) are recoded into a new, single value.

Examples

```
## Not run:
oldstring <- c("Hello", "Helo", "Hole", "Apple", "Ape", "New", "Old", "System", "Systemic")
newstring <- sju.groupString(oldstring)
sjt.frq(data.frame(oldstring, newstring), removeStringVectors = FALSE, autoGroupStrings = FALSE)

newstring <- sju.groupString(oldstring, strict = TRUE)
sjt.frq(data.frame(oldstring, newstring), removeStringVectors = FALSE, autoGroupStrings = FALSE)
## End(Not run)
```

---

sju.groupVar *Recode count variables into grouped factors*

Description

Recode count variables into grouped factors.

Usage

```
sju.groupVar(var, groupsize = 5, asNumeric = TRUE, rightInterval = FALSE,
  autoGroupCount = 30)
```

Arguments

var	The count variable, which should recoded into groups.
groupsize	The group-size, i.e. the range for grouping. By default, for each 5 categories a new group is defined, i.e. groupsize=5. Use groupsize="auto" to automatically resize a variable into a maximum of 30 groups (which is the ggplot-default grouping when plotting histograms). Use autoGroupCount to determin the amount of groups.
asNumeric	If TRUE (default), the recoded variable will be returned as numeric vector. If FALSE, a factor is returned.
rightInterval	If TRUE, grouping starts with the lower bound of groupsize. In this case, groups cover the ranges from 50-54, 55-59, 60-64 etc. If FALSE (default), grouping starts with the upper bound of groupsize. In this case, groups cover the ranges from 51-55, 56-60, 61-65 etc.
autoGroupCount	Sets the maximum number of groups that are defined when auto-grouping is on (groupsize="auto"). Default is 30. If groupsize is not set to "auto", this parameter will be ignored.

Value

A grouped variable, either as numeric or as factor (see paramter asNumeric).

See Also

sju.groupVarLabels

Examples

```
age <- abs(round(rnorm(100, 65, 20)))
age.grp <- sju.groupVar(age, 10)
hist(age)
hist(age.grp)

# histogram with EUROFAMCARE sample dataset
# variable not grouped
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
sjp.frq(efc$e17age,
        title=efc.var[['e17age']],
        type="h",
        showValueLabels=FALSE)

# bar plot with EUROFAMCARE sample dataset
# grouped variable
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
ageGrp <- sju.groupVar(efc$e17age)
ageGrpLab <- sju.groupVarLabels(efc$e17age)
```

```
sjp.frq(ageGrp,
        title=efc.var[['e17age']],
        axisLabels.x=ageGrpLab,
        maxYlim=FALSE)
```

`sju.groupVarLabels` *Create labels for recoded groups*

Description

Creates the related labels for the grouped variable created by the `sju.groupVar` function.

Usage

```
sju.groupVarLabels(var, groupsize = 5, rightInterval = FALSE,
  autoGroupCount = 30)
```

Arguments

`var`	The scale variable, which should recoded into groups.
`groupsize`	The group-size, i.e. the range for grouping. By default, for each 5 categories new group is built, i.e. `groupsize=5`. Use `groupsize="auto"` to automatically resize a variable into a maximum of 30 groups (which is the ggplot-default grouping when plotting histograms). Use parameter `autoGroupCount` to define the amount of groups.
`rightInterval`	If `TRUE`, grouping starts with the lower bound of `groupsize`. In this case, groups cover the ranges from 50-54, 55-59, 60-64 etc. If `FALSE` (default), grouping starts with the upper bound of `groupsize`. In this case, groups cover the ranges from 51-55, 56-60, 61-65 etc.
`autoGroupCount`	Sets the maximum number of groups that are built when auto-grouping is on (`groupsize="auto"`). Default is 30. If `groupsize` is not set to `"auto"`, this parameter will be ignored.

Value

A string vector containing labels based on the grouped counts of `var`, formatted as "from lower bound to upper bound", e.g. `"10-19" "20-29" "30-39"` etc. See example below.

Note

Usually you should use the same values for `groupsize` and `rightInterval` as used in the `sju.groupVar` function if you want to create labels for the related recoded variable.

See Also

`sju.groupVar`

Examples

```
age <- abs(round(rnorm(100, 65, 20)))
age.grp <- sju.groupVar(age, 10)
hist(age)
hist(age.grp)

age.grpvar <- sju.groupVarLabels(age, 10)
table(age.grp)
print(age.grpvar)

# histogram with EUROFAMCARE sample dataset
# variable not grouped
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
sjp.frq(efc$e17age,
        title=efc.var[['e17age']],
        type="h",
        showValueLabels=FALSE)

# bar plot with EUROFAMCARE sample dataset
# grouped variable
data(efc)
efc.val <- sji.getValueLabels(efc)
efc.var <- sji.getVariableLabels(efc)
ageGrp <- sju.groupVar(efc$e17age)
ageGrpLab <- sju.groupVarLabels(efc$e17age)
sjp.frq(ageGrp,
        title=efc.var[['e17age']],
        axisLabels.x=ageGrpLab,
        maxYlim=FALSE)
```

---

sju.mean.n *Compute row means with min amount of valid values*

Description

This function is similar to the SPSS `MEAN.n` function and computes row means from a data.frame or matrix if at least `n` values of a row a valid (and not NA).

Usage

```
sju.mean.n(df, n)
```

Arguments

`df`	a data.frame with at least two columns, where row means are applied.
`n`	the amount of valid values per row to calculate the row mean. If a row's valid values is smaller than `n`, NA will be returned as row mean value.

Value

A vector with row mean values of df for those rows with at least n valid values. Else, NA is returned.

References

- http://candrea.ch/blog/compute-spss-like-mean-index-variables/
- http://r4stats.com/2014/09/03/adding-the-spss-mean-n-function-to-r/

Examples

```
df <- data.frame(a=c(1,2,NA,4), b=c(NA,2,NA,5), c=c(NA,4,NA,NA), d=c(2,3,7,8))
sju.mean.n(df, 4) # 1 valid return value
sju.mean.n(df, 3) # 2 valid return values
sju.mean.n(df, 2)
sju.mean.n(df, 1) # all means are shown
```

---

sju.recode *Recode variable values.*

Description

Recodes the categories of a variables. Wrapper function that calls the recode function from the car package.

Usage

```
sju.recode(...)
```

Arguments

...	parameters, see recode function from the car package.

Value

A variable with recoded values.

Examples

```
data(efc)
table(efc$e42dep)
table(sju.recode(efc$e42dep, "1:2=1;3:4=2"))
```

sju.recodeTo *Recode variable categories into new values.*

Description

Recodes the categories of a variables var into new category values, beginning with the lowest value specified by parameter lowest. Useful if you want to recode dummy variables with 1/2 coding to 0/1 coding, or recoding scales from 1-4 to 0-3 etc.

Usage

```
sju.recodeTo(var, lowest = 0, highest = -1)
```

Arguments

var	The variable (vector) that should be recoded.
lowest	Indicating the lowest category value after recoding. Default is 0, so the new variable starts with the category value 0.
highest	If specified and larger than lowest, all category values larger than highest will be set to NA. Default is -1, i.e. this parameter is ignored and no NA's will be produced.

Value

A new variable with recoded category values, where lowest indicates the lowest value.

Examples

```
# recode 1-4 to 0-3
dummy <- sample(1:4, 10, replace=TRUE)
sju.recodeTo(dummy)

# recode 3-6 to 0-3
# note that numeric type is returned
dummy <- as.factor(3:6)
sju.recodeTo(dummy)

# lowest value starting with 1
dummy <- sample(11:15, 10, replace=TRUE)
sju.recodeTo(dummy, 1)

# lowest value starting with 1, highest with 3
# all others set to NA
dummy <- sample(11:15, 10, replace=TRUE)
sju.recodeTo(dummy, 1, 3)
```

sju.setNA *Set NA for specific variable values*

Description

This function sets specific values of a variable var as missings (NA).

Usage

```
sju.setNA(var, values)
```

Arguments

var	The variable where new missing values should be defined.
values	The values that should be replaced with NA's.

Value

The var with each values of values replaced by an NA.

Examples

```
# create random variable
dummy <- sample(1:8, 100, replace=TRUE)
# show value distribution
table(dummy)
# set value 1 and 8 as missings
dummy <- sju.setNA(dummy, c(1,8))
# show value distribution, including missings
table(dummy, exclude=NULL)
```

sju.strpos *Find partial matching and close distance elements in strings*

Description

This function finds the element indices of partial matching or similar strings in a character vector. Can be used to find exact or slightly mistyped elements in a string vector.

Usage

```
sju.strpos(searchString, findTerm, maxdist = 3, part.dist.match = 0,
  showProgressBar = FALSE)
```

Arguments

searchString	a character vector with string elements
findTerm	the string that should be matched against the elements of searchString.
maxdist	the maximum distance between two string elements, which is allowed to treat them as similar or equal.
part.dist.match	activates similar matching (close distance strings) for parts (substrings) of the searchString. Following values are accepted: • 0 for no partial distance matching • 1 for one-step matching, which means, only substrings of same length as findTerm are extracted from searchString matching • 2 for two-step matching, which means, substrings of same length as findTerm as well as strings with a slightly wider range are extracted from searchString matching Default value is 0.
showProgressBar	If TRUE, the progress bar is displayed when computing the distance matrix. Default in FALSE, hence the bar is hidden.

Value

A numeric vector with index position of elements in searchString that partially match or are similar to findTerm. Returns -1 if no match was found.

Note

this function does *not* return the position of a matching string *inside* another string, but the element's index of the searchString vector, where a (partial) match with findTerm was found. Thus, searching for "abc" in a string "this is abc" will not return 9 (the start position of the substring), but 1 (the element index, which is always 1 if searchString only has one element).

Examples

```
## Not run:
string <- c("Hello", "Helo", "Hole", "Apple", "Ape", "New", "Old", "System", "Systemic")
sju.strpos(string, "hel")   # partial match
sju.strpos(string, "stem")  # partial match
sju.strpos(string, "R")     # no match
sju.strpos(string, "saste") # similarity to "System"

# finds nothing
sju.strpos("We are Sex Pistols!", "postils")
# finds partial matching of similarity
sju.strpos("We are Sex Pistols!", "postils", part.dist.match = TRUE)
## End(Not run)
```

sju.weight *Weight a variable*

Description

This function weights the variable var by a specific vector of weights.

Usage

```
sju.weight(var, weights)
```

Arguments

var	The (unweighted) variable
weights	A vector with same length as var, which contains weight factors. Each value of var has a specific assigned weight in weights.

Value

The weighted var.

Note

The values of the returned vector are in sorted order, whereas the categories of the original var may be spread randomly. Hence, var can't be used, for instance, for further cross tabulation. In case you want to have weighted contingency tables or (grouped) box plots etc., use the weightBy parameter of most functions (like in sjt.xtab or sjp.grpfrq).

See Also

sju.weight2

Examples

```
v <- sample(1:4, 20, TRUE)
table(v)
w <- abs(rnorm(20))
table(sju.weight(v,w))
```

`sju.weight2` *Weight a variable*

Description

This function weights the variable `var` by a specific vector of `weights`. It's an alternative weight calculation to `sju.weight`, where `sju.weight` usage is recommended. This function sums up all `weights` values of the associated categories of `var`, whereas the `sju.weight` function uses a `xtabs` formula to weight cases. Thus, this function may return a value with a different length than that from `var`.

Usage

```
sju.weight2(var, weights)
```

Arguments

`var`	The (unweighted) variable
`weights`	A vector with same length as `var`, which contains weight factors. Each value of `var` has a specific assigned weight in `weights`.

Value

The weighted `var`.

Note

The values of the returned vector are in sorted order, whereas the categories of the original `var` may be spread randomly. Hence, `var` can't be used, for instance, for further cross tabulation. In case you want to have weighted contingency tables or (grouped) box plots etc., use the `weightBy` parameter of most functions (like in `sjt.xtab` or `sjp.grpfrq`).

See Also

`sju.weight`

Examples

```
v <- sample(1:4, 20, TRUE)
table(v)
w <- abs(rnorm(20))
table(sju.weight2(v,w))
```

sju.wordwrap *Insert line breaks in long labels*

Description

Insert line breaks in long character strings. Useful if you want to wordwrap plot labels.

Usage

```
sju.wordwrap(labels, wrap, linesep = NULL)
```

Arguments

labels	The label(s) (i.e. character string). You can also pass several strings as vector (e.g. labels=c("first long string", "second long string"))
wrap	The amount of chars per line (i.e. line length)
linesep	By default, this parameter is NULL and a regular new line string is used. For HTML-needs, for instance, linesep could be " ".

Value

New label(s) with line breaks inserted at every wrap's position.

Examples

```
sju.wordwrap(c("A very long string", "And another even longer string!"), 10)
```

Index