Package ‘lessR’

June 23, 2024

Version 4.3.6

Date 2024-06-22

Title Less Code, More Results

Author David Gerbing, The School of Business, Portland State University

Maintainer David W. Gerbing <gerbing@pdx.edu>

Depends R (>= 3.6.0)

Imports graphics, grDevices, stats, utils, methods, lattice, latticeExtra, robustbase, ellipse, leaps, openxlsx, colorspace, shiny, knitr

Suggests KernSmooth, rmarkdown, wesanderson, haven, readODS, triangle, arrow, xts, zoo

VignetteBuilder knitr

Description
Each function accomplishes the work of multiple standard R functions. For example, two function calls, Read() and CountAll(), read the data and generate summary statistics for all variables in the data frame, plus histograms and bar charts as appropriate. Other functions provide for comprehensive summary statistics via pivot tables, a comprehensive regression analysis, ANOVA and t-test, visualizations including the Violin/Box/Scatter plot for a numerical variable, bar chart, histogram, box plot, density curves, calibrated power curve, reading multiple data formats with the same function call, variable labels, color themes, and Trellis graphics. Also includes a confirmatory factor analysis of multiple indicator measurement models, pedagogical routines for data simulation such as for the Central Limit Theorem, generation and rendering of regression instructions for interpretative output, and interactive visualizations.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-22 22:20:02 UTC

Contents

. *Function . for Selecting Rows/Columns with base R Extract*

Description

Using the base R `Extract` function, with the unobtrusive function name, `.`, express a subsetting operation as
d[.(rows), .(cols)]
for a less annoying experience. With `.` to express a logical criterion to select rows, do not append the data frame name and `$` to variable names in expressions as otherwise required by `Extract`. Can also do a random selection of rows. For columns, no need to quote variable names, can include variable ranges defined by a colon, :, and add `-` to exclude designated columns. Also does not list rows missing data when not requested as does `Extract`.

Usage

```
.(x, ...)
```

Arguments

`x`	Logical expression to subset rows or columns.
`...`	Allows multiple expressions when selecting columns.

‘

Details

Eliminates the need to prepend the data frame name and a dollar sign to each variable name in the specified logical expression to select rows. For columns, no quoting variables, allow variable ranges.

Can create a character string called `rows` that expresses the logic of row selection. Can create a character string called `cols` that expresses the logic of column (variable) selection. To negate the `rows` expression, `.(!rows)`. Use `-.(cols)` to exclude designated variables.

Select a random selection of rows with the containing function `random(n)`, where `n` is the specified number of random rows to select from the full data frame and `.n` is the proportion of random rows to select.

Value

The row or columns names of the rows of data or columns of data that satisfy the specified logical conditions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Extract subset.

Examples

```
# see vignette

d <- Read("Employee", quiet=TRUE)

# no data frame name attached to variable names
#   as variables assumed in the data frame
d[.(Gender=="M" & Post>90), ]

# include first three rows and only the specified variables
# variable range permitted
d[1:3, .(Years:Salary, Post)]

#  include first three rows and delete the specified variables
d[1:3, -.(Years:Salary, Post)]

# select rows and columns
d[.(Gender=="M" & Post>90), .(Years:Salary, Post)]

# because of the default for the base R Extract function [ ],
# if only one variable retained,
# then add drop=FALSE to retain the result as a data frame
d[1:3, .(Salary), drop=FALSE]

# define character string arguments
cols <- "Gender:Salary, Post"
rows <- "Gender=='M' & Post>93"
d[.(rows), .(cols)]
# negate
d[.(!rows), -.(cols)]

# random selection of 4 rows, retain all variables
d[.(random(4)), ]
```

ANOVA *Analysis of Variance*

Description

Abbreviation: `av`, `av_brief`

Analysis of variance from the R `aov` function plus graphics and effect sizes. Included designs are one-way between groups, two-way between groups and randomized blocks with one treatment factor with one observation for each treatment and block combination.

Output is generated into distinct segments by topic, organized and displayed in sequence by default. When the output is assigned to an object, such as `a` in `a <- reg(Y ~ X)`, the full or partial output can be accessed for later analysis and/or viewing. A primary such analysis is with `knitr` for dynamic report generation. The input instructions to `knitr` are written comments and interpretation with embedded `R` code, called R~Markdown. Generate a complete, though preliminary at this time, R Markdown document from the `Rmd` option ready to knit. Simply specify the option with a file name, run the ANOVA function to create the file. Then open the newly created `.Rmd` file in `RStudio` and click the `knit` button to create a formatted document that consists of the statistical results and interpretative comments. See the sections `arguments`, `value` and `examples` for more information.

Usage

```
ANOVA(my_formula, data=d, filter=NULL,
         brief=getOption("brief"), digits_d=NULL,
         Rmd=NULL, jitter_x=0.4,
         res_rows=NULL, res_sort=c("zresid", "fitted", "off"),
         graphics=TRUE, pdf=FALSE, width=5, height=5,
         fun_call=NULL, ...)

av(...)

av_brief(..., brief=TRUE)
```

Arguments

`my_formula`	Standard R `formula` for specifying a model. Use an asterisk, `*`, separating the two factors for a two-way ANOVA, and a plus, `+`, separating the factors for a randomized blocks ANOVA with the blocking factor listed second.
`data`	The default name of the data frame that contains the data for analysis is `d`, otherwise explicitly specify.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`brief`	If set to `TRUE`, reduced text output with no Tukey multiple comparison of means and no residuals. Can change system default with `style` function.
`digits_d`	For the Basic Analysis, it provides the number of decimal digits. For the rest of the output, it is a suggestion only.
`Rmd`	File name for the file of R Markdown instructions to be written, if specified. The file type is .Rmd, which automatically opens in RStudio, but it is a simple text file that can be edited with any text editor, including RStudio.
`jitter_x`	Amount of horizontal jitter for points in the scatterplot of levels and response variable for a one-way ANOVA.
`res_rows`	Default is 20, which lists the first 20 rows of data and residuals sorted by the specified sort criterion. To disable residuals, specify a value of 0. To see the residuals output for all observations, specify a value of `"all"`.
`res_sort`	Default is `"zresid"`, for specifying standardized residuals as the sort criterion for the display of the rows of data and associated residuals. Other values are `"fitted"` for the fitted values and `"off"` to not sort the rows of data.
`graphics`	Produce graphics. Default is `TRUE`. In `Rmd` can be useful to set to `FALSE` so that `regPlot` can be used to place the graphics within the output file.

pdf	Indicator as to if the graphic files should be saved as pdf files instead of directed to the standard graphics windows.
width	Width of the pdf file in inches.
height	Height of the pdf file in inches.
fun_call	Function call. Used with Rmd to pass the function call when obtained from the abbreviated function call av.
...	Other parameter values for R function lm which provides the core computations.

Details

OVERVIEW
The one-way ANOVA with Tukey HSD and corresponding plot is based on the R functions aov, TukeyHSD, and provides summary statistics for each level. Two-factor ANOVA also provides an interaction plot of the means with interaction.plot as well as a table of means and other summary statistics. The two-factor analysis can be between groups or a randomized blocked design. Residuals are displayed by default. Tukey HSD comparisons and residuals are not displayed if brief=TRUE.

The filter parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than. See the Examples.

MODEL SPECIFICATION
In the following specifications, Y is the response variable, X is a treatment variable and Blocks is the blocking variable. The distinction between the one-way randomized blocks and the two-way between groups models is not the variable names, but rather the delimiter between the variable names. Use * to indicate a two-way crossed between groups design and + for a randomized blocks design.
one-way between groups: ANOVA(Y ~ X)
one-way randomized blocks: ANOVA(Y ~ X + Blocks)
two-way between groups: ANOVA(Y ~ X1 * X2)
For more complex designs, use the standard R function aov upon which ANOVA depends.

BALANCED DESIGN
The design for the two-factor analyses must be balanced. A check is performed and processing ceases if not balanced. For unbalanced designs, consider the function lmer in the lme4 package.

DECIMAL DIGITS
The number of decimal digits displayed on the output is, by default, the maximum number of decimal digits for all the data values of the response variable. Or, this value can be explicitly specified with the digits_d parameter.

Value

The output can optionally be returned and saved into an R object, otherwise it simply appears at the console. The components of this object are redesigned in lessR version 3.3.5 into (a) pieces of text that form the readable output and (b) a variety of statistics. The readable output are character strings such as tables amenable for viewing and interpretation. The statistics are numerical values amenable for further analysis, such as to be referenced in a subsequent R Markdown document.

The motivation of these two types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object followed by a $, can be inserted into the R markdown document (see `examples`).

TEXT OUTPUT
`out_background`: variables in the model, rows of data and retained
1-predictor: `out_descriptive`: descriptive stats
2-predictors: `out_cell.n`: cell sample size
2-predictors: `out_cell.means`: cell means
2-predictors: `out_cell.marginals`: marginal means
2-predictors: `out_cell.gm`: grand mean
2-predictors: `out_cell.sd`: cell standard deviations
`out_anova`: analysis of variance summary table
`out_effects`: effect sizes
`out_hsd`: Tukey's honestly significant different analysis
`out_res`: residuals
`out_plots`: list of plots generated if more than one

Separated from the rest of the text output are the major headings, which can then be deleted from custom collations of the output. `out_title_bck`: BACKGROUND
`out_title_des`: DESCRIPTIVE STATISTICS
`out_title_basic`: BASIC ANALYSIS
`out_title_res`: RESIDUALS

STATISTICS
`call`: function call that generated the analysis
`formula`: model formula that specifies the model
`n_vars`: number of variables in the model
`n_obs`: number of rows of data submitted for analysis
`n_keep`: number of rows of data retained in the analysis
1-predictor: `p_value`: p-value for the overall F-test `residuals`: residuals
`fitted`: fitted values

Although not typically needed for analysis, if the output is assigned to an object named, for example, `a`, then the complete contents of the object can be viewed directly with the `unclass` function, here as `unclass(a)`. Invoking the `class` function on the saved object reveals a class of `out_all`. The class of each of the text pieces of output is `out`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapters 8 and 9, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

See Also

aov, TukeyHSD, interaction.plot

Examples

```
# access the PlantGrowth data frame
ANOVA(weight ~ group, data=PlantGrowth)
#brief version
av_brief(weight ~ group, data=PlantGrowth)

# drop the second treatment, just control and 1 treatment
ANOVA(weight ~ group, data=PlantGrowth, filter=(group != "trt2"))

# variables of interest in a data frame that is not the default d
# two-factor between-groups ANOVA with replications and interaction
# warpbreaks is a data set provided with R
ANOVA(breaks ~ wool * tension, data=warpbreaks)

# randomized blocks design with the second term the blocking factor
#   data from Gerbing(2014, Sec 7.3.1)

# Each person is a block. Each person takes four weight-training
#   supplements on different days and then count the repetitions
#   of the bench presses.
d <- read.csv(header=TRUE, text="
Person,sup1,sup2,sup3,sup4
p1,2,4,4,3
p2,2,5,4,6
p3,8,6,7,9
p4,4,3,5,7
p5,2,1,2,3
p6,5,5,6,8
p7,2,3,2,4")

# reshape data from wide form to long form
# do not need the row names
d <- reshape(d, direction="long",
        idvar="Person", v.names="Reps",
        varying=list(2:5), timevar="Supplement")
rownames(data) <- NULL

ANOVA(Reps ~ Supplement + Person)
```

BarChart *Bar Chart for One or Two Variables*

Description

Abbreviation: `bc`

The function plots a bar chart, one categorical variable, `x`, against one numeric variable, `y`, possibly including an optional second categorical variable, `by`. The bar chart is constructed from the usually relatively brief table that pairs each level of the categorical variables with the corresponding numerical value of `y`. Usually, this table is a summary (pivot) table calculated as a data aggregation from the original data table of measurements, such as average salary of the employees in each department.

The calculation of this foundational summary table from which the bar chart is created can occur outside of the function. Or, probably the more usual situation, the table is implicitly calculated by the function in one of two ways. Accordingly, obtain the summary table from one of three possibilities.

1. Enter the summary table obtained from an external source directly as the value of the `data` parameter, indicated by specifying categorical variables `x` and possibly `by` with the numerical variable `y`.
2. Have the function implicitly summarize the entire data table. If only categorical variable `x`, and possibly categorical variable `by`, are specified without a value of numerical `y`, the entire data table must be input as the value of `data`. The function then computes numeric variable `y` as the computed frequency of values in each category or level of the specified categorical variables.
3. Have the function implicitly summarize the entire data table entered as the value of `data` by specifying a `y` variable. Obtain the summary table from which the bar chart is computed by summarizing (aggregating) the value of `y` at each level of `x`, and possibly `by`, with the chosen statistic specified by the `stat` parameter. The function will assess if the input data is a summary table or the entire data table. If the entire data table is entered, and the `stat` parameter is not entered, the value of `stat` defaults to the mean.

The function also displays the foundational summary table, such as frequency table for one or two variables. If a frequency table, also displayed are Cramer's V association, and the corresponding chi-square inferential analysis. For two variables, the frequencies include the joint and marginal frequencies. To activate Trellis graphics or facets, a multi-panel display, specify a `by1` variable in place of `by` for the second categorical variable. If the provided object to analyze is a set of multiple variables, including the name of an entire data frame, then a bar chart is calculated for *each* non-numeric variable in the data frame.

Usage

```
BarChart(

        # ------------------------------------------
        # Data from which to construct the bar chart
        x=NULL, y=NULL, by=NULL, data=d, filter=NULL,

        # -----------------------------
        # Bar chart from aggregated data
        stat=c("mean", "sum", "sd", "deviation", "min", "median", "max"),
        stat_x=c("count", "proportion"),
```

```
# ----------------------------------------------------
# Trellis (facet) plot, stratify on different panels
by1=NULL, n_row=NULL, n_col=NULL, aspect="fill",

# ---------------------------------
# Layout and ordering of the bars
horiz=FALSE, sort=c("0", "-", "+"),
beside=FALSE, stack100=FALSE,
gap=NULL, scale_y=NULL, one_plot=NULL,

# ----------------------------------------------------------------
# Analogy of physical Marks on paper to create the bars and labels
theme=getOption("theme"),
fill=NULL,
color=getOption("bar_color_discrete"),
transparency=getOption("trans_bar_fill"),
fill_split=NULL,

labels=c("%", "input", "off"),
labels_color=getOption("labels_color"),
labels_size=getOption("labels_size"),
labels_decimals=getOption("labels_decimals"),
labels_position=getOption("labels_position"),
labels_cut=NULL,

# ------------------------------------------------------------------
# Labels for axes, values, and legend if x and by variables, margins
xlab=NULL, ylab=NULL, main=NULL, sub=NULL,
lab_adjust=c(0,0), margin_adjust=c(0,0,0,0),
pad_y_min=0, pad_y_max=0,

rotate_x=getOption("rotate_x"), break_x=NULL,
offset=getOption("offset"),
label_max=100,

legend_title=NULL, legend_position="right_margin",
legend_labels=NULL, legend_horiz=FALSE,
legend_size=NULL, legend_abbrev=NULL, legend_adjust=0,

# ------------------------------------------------------
# Draw one or more objects, text, or geometric figures
add=NULL, x1=NULL, y1=NULL, x2=NULL, y2=NULL,

# --------------------------------------------------------------------
# Output: text or chart turned off, to PDF file, number decimal digits
quiet=getOption("quiet"), do_plot=TRUE,
pdf_file=NULL, width=6.5, height=6,
```

```
        digits_d=NULL, out_size=80,

        # ------------------------------------
        # Deprecated, removed in future versions
        n_cat=getOption("n_cat"), value_labels=NULL, rows=NULL,

        # -------------
        # Miscellaneous
        eval_df=NULL, ...)

bc(...)
```

Arguments

`x` Categorical **variable(s)** to analyze. Can be a single variable, either within a data frame or as a vector in the users workspace, or multiple variables in a data frame such as designated with the `c` function, or an entire data frame. If not specified, then defaults to all non-numerical variables in the specified data frame, `d` by default.

To manage large category values, unless `break_x` is `FALSE`, any space in each category value is converted to new line for the corresponding axis label in the plot. To keep two (small) words on the same line, replace the space that separates them with a tilde, which displays as a blank for the corresponding axis label.

`y` Numeric variable with a value for each level of the categorical variable with the value plotted proportional to the height of the corresponding bar. If specified for the original data table, then the corresponding `stat` parameter also must be set. If not specified, then its value is by default tabulated as the frequency of each category or joint category.

`by` A second categorical variable to create a two-variable bar chart for each level of the numeric primary variable `y` on the *same* plot. A similar concept applies to the panels of a Trellis (facet) plot if `by1` is specified.

`data` Optional data frame that contains the variables of interest. Can contain data from which frequencies or other statistics for a `y`-variable are computed, or can be a summary table that consists of two columns: the level of a categorical variable paired with the numeric value that determines the height of the corresponding bar.

`filter` A logical expression that specifies a subset of rows of the data frame to analyze.

`stat` Statistical transformation of the data for the `y`-axis across groups defined by the categorical variable(s), the data aggregation. Applicable values: `"sum"`, `"mean"`, `"sd"`, `"dev"` for mean deviations, `"min"`, `"median"`, and `"max"`.

`stat_x` When no `y` variable is specified, either do the default `count` of each group or the `proportion`.

`by1` A categorical variable called a conditioning variable that activates **Trellis graphics (facets)**, from the `lattice` package, to create a bar chart on a separate panel for each level of the variable. Contrast to the `by` parameter that plots on the same panel.

n_row	Optional specification for the number of rows in the layout of a multi-panel display with Trellis graphics (facets). Need not specify n_col.
n_col	Optional specification for the number of columns in the layout of a multi-panel display with Trellis graphics (facets). Need not specify n_row. If set to 1, then the strip that labels each group locates to the left of each plot instead of the top.
aspect	Lattice parameter for the aspect ratio of the panels in a Trellis plot (multi-panel display or facets), defined as height divided by width. The default value is "fill" to have the panels expand to occupy as much space as possible. Set to 1 for square panels. Set to "xy" to specify a ratio calculated to "bank" to 45 degrees, that is, with the line slope approximately 45 degrees.
horiz	**Bar orientation.** By default the value is FALSE so bars are vertical, unless one_plot is TRUE.
sort	Sort the categories by their frequency for one variable and by the column sums if a by variable. Not applicable to Trellis plots. By default "0" for no sort, or sort descending "-" or ascending "+", unless one_plot is TRUE, then is set to "+".
beside	For a two variable plot, set to TRUE for the levels of the first variable to be plotted as adjacent bars instead of stacked on each other.
stack100	100% stacked bar chart when a by variable is present, also activated by setting stat_x to "proportion" with a by variable.
gap	Gap between bars. Provides the value of the space option from the standard R barplot function with a default of 0.2 unless two variables are plotted and beside=TRUE, in which case the default is c(.1,1).
scale_y	If specified, a vector of three values that define the numerical values of the y-axis, the numerical axis, within the bounds of plot region: starting value, ending value, and number of intervals.
one_plot	For bar charts of multiple x-variables, indicates if a bar plot is produced for each x-variable, or all are combined into a single plot, such as for items that all share common responses such as survey data with a common Likert scale across variables. Default is if variables share a common response scale set to TRUE, otherwise FALSE.
theme	Theme for the **colors** for this analysis. Make persistent across analyses with style.
fill	Fill color of the bars. Default is the qualitative palette "hues" from default theme "colors", unless the categorical variable(s) is(are) ordinal where the default is the "blues" sequential gradient. For any other color theme the default is the corresponding color gradient, such as "reds" for theme "darkred". Can also specify any vector of colors to fill the bars, such as generated by getColors, or access more pre-defined gradients such as palettes that address color-blindness such as "virdis". Or set to the name of y to map the values of bar fill into the fill colors. Specified the name of y as (count) if tabulated from the data. Not applicable if fill_split is activated.
color	Border color of the bars, can be a vector to customize the color for each bar. Default is bar_color_discrete from the lessR style function.

transparency Transparency factor of the area of each slice from 0, no transparency to 1, full transparency. Default is trans_bar_fill from the lessR style function.

fill_split The value of the numeric variable y for which bars that correspond to values of y <= fill_split are displayed in the first fill color and other values displayed in the second fill color, or as specified by a vector of exactly two fill colors.

labels If not "off", adds the numerical results to the plot according to the default "%" for tabulated counts and "input" for the input values for a y-variable explicitly provided, unless there are more than 15 levels or y is present and non-integer in which case the default is "off". For tabulated counts, "prop" is also available for proportions, as well as "input" to show the computed values such as counts.

labels_color Color of the plotted text. Could be a vector to specify a unique color for each value. If fewer colors are specified than the number of categories, the colors are recycled.

labels_size Character expansion factor, the size, of the plotted text, for which the default value is 0.95, or 0.9 of value if beside is TRUE and labels_position is "in" because bars are narrower.

labels_decimals
Number of decimal digits for which to display the values. Default is 0, round to the nearest integer for "%", 2 for "prop", and if "input" and y is entered directly, display the literal value unless > 9999, in which case set to 0.

labels_position
Position of the plotted text. Default is "in" for inside the bar, or, if "out", the label for each value is placed outside of the bar, on top.

labels_cut Threshold for displaying the value. If labels_position equals "out", then default is 0.028 unless there is a by variable or multiple x-variables on the same plot, then default is 0.040

xlab **Axis label** for x-axis. If xlab is not specified, then the label becomes the name of the corresponding variable label if it exists, or, if not, the variable name. If xy_ticks is FALSE, then no label is displayed. If no y variable is specified, then xlab is set to Index unless xlab has been specified.

ylab Label for y-axis. If xlab is not specified, then the label becomes the name of the corresponding variable label if it exists, or, if not, the variable name. If xy_ticks is FALSE, then no label displayed.

main Label for the title of the graph. Can set size with main_cex and color with main_color from the lessR style function.

sub Sub-title of graph, below xlab. Not yet implemented.

lab_adjust Two-element vector – x-axis label, y-axis label – adjusts the position of the axis labels in approximate inches. + values move the labels away from plot edge. Not applicable to Trellis graphics.

margin_adjust Four-element vector – top, right, bottom and left – adjusts the margins of the plotted figure in approximate inches. + values move the corresponding margin away from plot edge. Not applicable to Trellis graphics.

pad_y_min Proportion of padding added to the left side of the y-axis. Value from 0 to 1.

pad_y_max
: Proportion of padding added to the right side of the y-axis. Value from 0 to 1.

rotate_x
: Degrees that the **axis values for the category values** axis are rotated, usually to accommodate longer values, typically used in conjunction with `offset`. When equal 90 the value labels are perpendicular to the x-axis and a different algorithm places the labels so that `offset` is not needed.

break_x
: Replace spaces in the category values with a new line and replace tildes with a blank so that there is no separation of words joined by a tilde. By default, `TRUE` for vertical bar charts with `rotate_x` set to `0`, and `FALSE` otherwise.

offset
: The amount of spacing between the axis values and the axis. Default is 0.5. Larger values such as 1.0 create space for the label when longer axis value names are rotated.

label_max
: To **improve readability** of text output, the maximum size of the value labels before the labels are abbreviated for text output only. Not a literal maximum as preserving unique values may require a larger number of characters than specified.

legend_title
: Title of the **legend**, which is usually set by default except when raw counts are entered as a matrix. Then a title must be specified to generate a legend.

legend_position
: When plotting two variables, location of the legend, with the default in the right margin. Additional options from standard R are "topleft", "top", "topright" and others as shown in the help for the `legend` function.

legend_labels
: When plotting two variables, labels for the legend, which by default are the levels for the second or `by` variable.

legend_horiz
: By default the legend is vertical, but can be changed to horizontal.

legend_size
: Size of legend text.

legend_abbrev
: If specified, abbreviate legend title and legend labels to the specified number of the maximum number of characters.

legend_adjust
: Shift legend for a two-categorical bar chart. A positive number shifts the legend to the right from its default placement.

add
: **Draw one or more objects**, text, or geometric figures, on the plot. Possible values are any text to be written, the first argument, which is `"text"`, or, to indicate a figure, `"rect"` (rectangle), `"line"`, `"arrow"`, `"v_line"` (vertical line), and `"h_line"` (horizontal line). The value `"means"` is short-hand for vertical and horizontal lines at the respective means. Does not apply to Trellis graphics. Customize with parameters such as `add_fill` and `add_color` from the `style` function.

x1
: First x coordinate to be considered for each object. All coordinates vary from -1 to 1.

y1
: First y coordinate to be considered for each object.

x2
: Second x coordinate to be considered for each object. Only used for `"rect"`, `"line"` and `arrow`.

y2 — Second y coordinate to be considered for each object. Only used for "rect", "line" and arrow.

quiet — If set to TRUE, no text output. Can change system default with style function.

do_plot — If TRUE, the default, then generate the plot.

pdf_file — Indicate to direct pdf graphics to the specified name of the pdf file.

width — Width of the plot window in inches, defaults to 4.5.

height — Height of the plot window in inches, defaults to 4.5.

digits_d — Provides the number of decimal digits, set by default to at least 2 or the largest number of digits in the values of the response variable plus 1.

out_size — To improve the readability of the frequency distribution of a single variable displayed at the console, the maximum number of characters on a line of output at the console for one variable before the frequency distribution is written vertically.

n_cat — When analyzing all the variables in a data frame, specifies the largest number of unique values of variable of a numeric data type for which the variable will be analyzed as a categorical. Default is 0. *[deprecated]*: Better to convert a categorical integer variable to a factor.

value_labels — For factors, default is the factor labels, and for character variables, default is the character values. Or, provide labels for the x-axis on the graph to override these values. If the variable is a factor and value_labels is not specified (is NULL), then the value_labels are set to the factor levels with each space replaced by a new line character. If x and y-axes have the same scale, they also apply to the y-axis. Control the plotted size with axis_cex and axis_x_cex from the lessR style function. *[deprecated]*: Better to convert a categorical integer variable to a factor.

rows — Deprecated old parameter name that is now called filter.

eval_df — Determines if to check for existing data frame and specified variables. By default is TRUE unless the shiny package is loaded then set to FALSE so that Shiny will run. Needs to be set to FALSE if using the pipe %\>% notation.

... — Other parameter values for graphics as defined by Base R barplot, legend, and par including xlim and ylim for setting the range of the x and y-axes
cex.main for the size of the title
col.main for the color of the title
"dotted", "dotdash"
sub and col.sub for a subtitle and its color
las=3 to reorient vertical axis labels
space for one variable only

Details

OVERVIEW
Plot a bar chart with default colors for one or two categorical variables, that is, with a relatively

small number of labels for each variable. By default, colors are selected for the bars, background and grid lines, all of which can be customized. The basic computations of the chart are provided with the standard R functions `barplot`, `chisq.test` and, for two variables, `legend`. Horizontal bar charts, specified by `horiz=TRUE`, list the value labels horizontally and automatically extend the left margin to accommodate both the value labels and the variable label.

DATA
Ultimately the bar chart is constructed from a simple summary table in which each row consists of a level of the categorical variable `x` paired with the corresponding value of the numerical variable, `y`, with as many rows as the number of levels of `x`. Provide these values of `x` and `y` directly, or just provide `x` for the original data of measurements to compute the counts of each category or provide `x` and `y` with a value of `stat` to define the statistic for which to aggregate the values of `y` over the levels of `x`. Also can have a second categorical variable, `by`.

The data may either be vectors from the global environment, the user's workspace, as illustrated in the examples below, or a variable in a data frame. The default input data frame is `d`. Specify a different data frame name with the `data` option. Regardless of its name, the variables in the data frame are referenced directly by their names.

If the name of the vector is in the global environment and of a variable in the input data frame has the same name, the vector from the global environment is analyzed, unless the data name frame is explicitly provided, not relying upon the default `d`. If two variables are specified, both variables should be in the data frame, or one of the variables is in the data frame and the other in the global environment.

To obtain a bar chart of each categorical variable in the `d` data frame, invoke `BarChart()`. Or, for a data frame with a different name, insert the data frame name between the parentheses as the first listed parameter value. To analyze a subset of the variables in a data frame, specify the variable list with either a : or the `c` function, such as m01:m03 or c(m01,m02,m03).

The `rows` parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in `Logic` such as `&` for and, `|` for or and `!` for not. Use the standard R relational operators as described in `Comparison`. Examples include `==` for logical equality, `!=` for not equals, and `>` for greater than. See the Examples.

The form of the entered data, the first variable, categorical `x`, and optionally a second variable, numerical `y`, is flexible. The data may be entered as factors, numeric values, characters, or a matrix. The data may be entered and the resulting frequencies computed, or the frequencies can be entered directly. The most natural type of data to enter, when entering the variables, is to enter factors.

STATISTICAL TRANSFORMATIONS
Ultimately the bar plot is constructed from a small table of data values with each row a level of the categorical variable `x` paired with the corresponding value of the numerical variable `y`, with as many rows as values of `x`. It is also possible to plot transformations of the values of `y` for each level of categorical variable `x` from a full data table with many replications of each value of `x` and corresponding `y`. Then reduce the larger data table down to the summary table with one of following transformations.

Transformation
"sum"
"mean"
"sd"

"dev"
"min"
"median"
"max"

The other statistical transformation is simply counting the number of occurrences of each level of x, which does not involve a v

COLORS
For a one variable plot, set the default color of the bars by the current color theme according to bar_fill_discrete argument of the function style, which includes the default color theme "hues" that defines a qualitative HCL color scale, or set the bar color with the fill parameter, which references a specified vector of color specifications, such as generated by the lessR getColors function.

Set fill to a single color or a color palette, of which there are many possibilities. Pre-defined sequential and divergent color ranges are available as implicit calls to getColors. Define the default qualitative color palette with "hues" that provides HCL colors of the same chroma (saturation) and luminance (brightness). The full list of pre-defined color ranges defined in 30 degree increments around the HCL color wheel: "reds", "rusts", "browns", "olives", "greens", "emeralds", "turquoises", "aquas", "blues", "purples","violets", "magentas", and "grays".

Define a *divergent color scale* with value of fill that consists of a vector of two such pre-defined ranges, such as c("purples", "rusts"). Divergent color palettes are applicable in particular for plotting multiple bar charts on the same plot such as for a set of Likert response items, all on a common response scale. Or, *manually specify colors*. For example, for a two-level by variable, could set fill to c("coral3","seagreen3"), where the specified colors are *not* pre-defined color ranges.

For the pre-defined color scales can obtain more control over the obtained color palettes with an explicit call to getColors for the argument to fill. Here the value of chroma (c) and luminance (l) can be explicitly manipulated in conjunction with the specification of a pre-defined color range. Or, create a custom color range for any value of hue (h). See getColors for more information.

The values of another variable can be mapped into the fill color of the bars. To do so, set fill to the value of the variable, which would usually be the name of the y variable if explicitly given. Or, if y is tabulated, refer to the variable name as (count). The larger the count for a level of x, the darker the bar.

Also available are the pre-specified R color palettes "rainbow", "terrain", and "heat". The pre-defined palette "distinct" maximally separates colors by hue. The family of color-blind family of viridis palettes are available as "viridis", "cividis", "magma", "inferno", and "plasma", as well as the "Okabe-Ito" palette. Pre-defined color palettes are available from many of Wes Anderson's movies such as "Moonrise1", "Royal1", "GrandBudapest1", "Darjeeling1" and "BottleRocket1". Can substitute a 2 for a 1 in the preceding references, and sometimes a 3.

LEGEND
When two variables are plotted, a legend is produced, with values for each level of the second or by variable. By default, the location is placed in the right margin of the plot. This position can be changed with the legend_position option, which, in addition to the lessR option of right_margin, accepts any valid value consistent with the standard R legend function, used to generate the legend.

The legend title can be abbreviated with the `legend_abbrev` parameter. Specify the maximum number of characters of the title. The legend is displayed vertically by default, but can be changed to horizontal with the `legend_horiz` option.

LONG CATEGORY NAMES
For many plots, the names of the categories are too long. To adjust the plot for these long names, they can be rotated using the `rotate_x` and `rotate_y` parameters, in conjunction with `offset`. The `offset` parameter moves the category name out from the axis to compensate for the rotation. The changes can also be specified from `style` to persist until further changes. To reset to the default after obtaining an analysis, use `style()`.

Also, the following codes are used to adjust line spacing:
1. Any space in a category name is converted to a new line.
2. If the space should not be converted to a new line, then replace with a tilde, ~, which will display as a space without a line break.

For the text output at the console, can specify the maximum number of characters in a label with `labels.max`. Longer value names are abbreviated to the specified length. This facilitates reading cross-tab tables. Also, a provided table pairs the abbreviated names with the actual names. For one variable frequency distributions, `out_size` provides the maximum number of characters for the text output before the horizontal display of the frequency distribution is shifted to a vertical presentation.

MULTIPLE BAR CHARTS ON THE SAME PANEL (PLOT)
For multiple `x`-variables, set the parameter `one_plot` to `TRUE` to specify that each bar chart should be produced on the same panel as all other bars. This is most meaningful when all items have the same set of responses, such as a common Likert scale found in survey data. By default the one panel plot is produced when a common response scale is detected.

The algorithm to detect if the response scale is common first identifies the first variable with the largest set of responses, then checks the responses of all other variables. If all responses to all other variables are contained within the set of responses to the reference variable, then the response scales are the same. This means that on a Likert scale, for example, some items may not contain all possible responses, such as no one selects Strongly Disagree for an item. However, for the response scales to be deemed the same, at least one item (variable) must contain all possible responses.

Regardless, the `one_plot` parameter can be set to either `TRUE` or `FALSE` regardless of the commonality of responses. Setting this parameter explicitly saves some CPU time as the algorithm to evaluate the communality of responses need not be activated.

ENTER NUMERIC VARIABLE DIRECTLY
Instead of calculating the counts from the data, the values of any numerical variable, including the counts, can be entered directly as the `y`-variable, in addition to the categorical `x`-variable, and perhaps a categorical `by`-variable. See the examples below.

Or, include the already tabulated counts as the data which is read into R, either as a matrix or a data frame.

STATISTICS
In addition to the bar chart, descriptive and optional inferential statistics are also presented. First, the frequency table for one variable or the joint frequency table for two variables is displayed. Second, the corresponding Cramer's V and chi-square test are also displayed by default.

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is listed as the label for the horizontal

axis unless xlab is specified in the function call. If there are two variables to plot, the title of the resulting plot is based on the two variable labels, unless a specific title is listed with the `main` option. The variable label is also listed in the text output, next to the variable name. If the analysis is for two variables, then labels for both variables are included.

PDF OUTPUT
To obtain pdf output, use the `pdf_file` option, perhaps with the optional `width` and `height` options. These files are written to the default working directory, which can be explicitly specified with the R `setwd` function.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a `lessR` function can only be a variable name (or list of variable names). This referenced variable must exist in either the referenced data frame, such as the default `d`, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

`> BarChart(cut(rnorm(50), breaks=seq(-5,5))) # does NOT work`

Instead, do the following:

```
> Y <- cut(rnorm(50), breaks=seq(-5,5))   # create vector Y in user workspace
 > BarChart(Y)     # directly reference Y
```

Value

The output can optionally be saved into an `R` object, otherwise it only appears in the console (unless `quiet` is set to `TRUE`). Two different types of components are provided: the pieces of readable output, and a variety of statistics. The readable output are character strings such as tables amenable for display. The statistics are numerical values amenable for further analysis. The motivation of these types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object and a `$`, can be inserted into the R~Markdown document (see `examples`), interspersed with explanation and interpretation.

Tabulated numerical variable y

READABLE OUTPUT
`out_title`: Title
`out_lbl`: Variable label
`out_counts`: Two-way frequency distribution
`out_chi`: Chi-square test
One variable: `out_miss`: Number of missing values
Two variables: `out_prop`: Cell proportions
Two variables: `out_row`: Cell proportions within each row
Two variables: `out_col`: Cell proportions within each col

STATISTICS
`n_dim`: Number of dimensions, 1 or 2
`p_value`: p-value for null of equal proportions or independence
`freq`: Data frame of the frequency distribution
One variable: `freq`: Frequency distribution
One variable: `values`: y-values read directly
One variable: `prop`: Frequency distribution of proportions

One variable: `n_miss`: Number of missing values

Numerical variable y read from data

`out_y`: Values of y
`n_dim`: Number of dimensions, 1 or 2

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapter 4, NY: Routledge.

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 3, NY: CRC Press.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

See Also

getColors, barplot, table, legend.

Examples

```
# get the data
d <- rd("Employee")

# ---------------------------------------------------------
# bar chart from tabulating the data for a single variable
# ---------------------------------------------------------

# for each level of Dept, display the frequencies
BarChart(Dept)
# short name
# bc(Dept)

# save the values output by BarChart into the myOutput list
myOutput <- BarChart(Dept)
# display the saved output
myOutput

# just males with salaries larger than 75,000 USD
BarChart(Dept, rows=(Gender=="M" & Salary > 85000))

# rotate and offset the axis labels, sort categories by frequencies
BarChart(Dept, rotate_x=45, offset=1, sort="-")
```

```
# set bars to a single color of blue with some transparency
BarChart(Dept, fill="blue", transparency=0.3)
# progressive (sequential) color scale of blues
BarChart(Dept, fill="blues")

# viridis palate
BarChart(Dept, fill="viridis")

# change the theme just for this analysis, as opposed to style()
BarChart(Dept, theme="darkgreen")

# set bar color to hcl custom hues with chroma and luminance
#   at the values provided by the default hcl colors from
#   the getColors function, which defaults to h=240 and h=60
#   for the first two colors on the qualitative scale
bc(Gender, fill=c(hcl(h=180,c=100,l=55), hcl(h=0,c=100,l=55)))

# or set to unique colors via color names
BarChart(Gender, fill=c("palegreen3","tan"))

# darken the colors with an explicit call to getColors,
#   do a lower value of luminance, set to l=25
BarChart(Dept, fill=getColors(l=25), transparency=0.4)

# column proportions instead of frequencies
BarChart(Gender, stat_x="proportion")

# map value of tabulated count to bar fill
BarChart(Dept, fill=(count))

# data with many values of categorical variable Make and large labels
myd <- Read("Cars93")
# perpendicular labels
bc(Make, rotate_x=90, data=myd)
# manage size of horizontal value labels
bc(Make, horiz=TRUE, label_max=4, data=myd)

# read y variable, Salary
# display bars for values of count <= 0 in a different color
#  than values above
BarChart(Dept, Salary, stat="dev", sort="+", fill_split=0)


# ---------------------------------------------------
# bar chart from tabulating the data for two variables
# ---------------------------------------------------

# at each level of Dept, show the frequencies of the Gender levels
BarChart(Dept, by=Gender)

# Trellis (facet) plot
BarChart(Dept, by1=Gender)
```

```
# at each level of Dept, show the row proportions of the Gender levels
#   i.e., 100% stacked bar graph
BarChart(Dept, by=Gender, stack100=TRUE)

# at each level of Gender, show the frequencies of the Dept levels
# do not display percentages directly on the bars
BarChart(Gender, by=JobSat, fill="reds", labels="off")

# specify two fill colors for Gender
BarChart(Dept, by=Gender, fill=c("deepskyblue", "black"))

# display bars beside each other instead of stacked, Female and Male
# the levels of Dept are included within each respective bar
# plot horizontally, display the value for each bar at the
#   top of each bar
BarChart(Gender, by=Dept, beside=TRUE, horiz=TRUE, labels_position="out")

# horizontal bar chart of two variables, put legend on the top
BarChart(Gender, by=Dept, horiz=TRUE, legend_position="top")

# for more info on base R graphic options, enter:  help(par)
# for lessR options, enter:  style(show=TRUE)
# here fill is set in the style function instead of BarChart
#   along with the others
style(fill=c("coral3","seagreen3"), lab_color="wheat4", lab_cex=1.2,
      panel_fill="wheat1", main_color="wheat4")
BarChart(Dept, by=Gender,
         legend_position="topleft", legend_labels=c("Girls", "Boys"),
         xlab="Dept Level", main="Gender for Different Dept Levels",
         value_labels=c("None", "Some", "Much", "Ouch!"))
style()


# ----------------------------------------------------------------
# multiple bar charts tabulated from data across multiple variables
# ----------------------------------------------------------------

# bar charts for all non-numeric variables in the data frame called d
#   and all numeric variables with a small number of values, < n_cat
# BarChart(one_plot=FALSE)

d <- rd("Mach4", quiet=TRUE)

# all on the same plot, bar charts for 20 6-pt Likert scale items
# default scale is divergent from "browns" to "blues"
BarChart(m01:m20, horiz=TRUE, labels="off", sort="+")




# custom scale with explicit call to getColors, HCL chroma at 50
clrs <- getColors("greens", "purples", c=50)
BarChart(m01:m20, horiz=TRUE, labels="off", sort="+", fill=clrs)
```

```
# custom divergent scale with pre-defined color palettes
#  with implicit call to getColors
BarChart(m01:m20, horiz=TRUE, labels="off", fill=c("aquas", "rusts"))


# ----------------------------
# can enter many types of data
# ----------------------------

# generate and enter integer data
X1 <- sample(1:4, size=100, replace=TRUE)
X2 <- sample(1:4, size=100, replace=TRUE)
BarChart(X1)
BarChart(X1, by=X2)

# generate and enter type double data
X1 <- sample(c(1,2,3,4), size=100, replace=TRUE)
X2 <- sample(c(1,2,3,4), size=100, replace=TRUE)
BarChart(X1)
BarChart(X1, by=X2)

# generate and enter character string data
# that is, without first converting to a factor
Travel <- sample(c("Bike", "Bus", "Car", "Motorcycle"), size=25, replace=TRUE)
BarChart(Travel, horiz=TRUE)


# ----------------------------
# bar chart directly from data
# ----------------------------

# include a y-variable, here Salary, in the data table to read directly
d <- read.csv(text="
Dept, Salary
ACCT,51792.78
ADMN,71277.12
FINC,59010.68
MKTG,60257.13
SALE,68830.06", header=TRUE)
BarChart(Dept, Salary)

# specify two variables for a two variable bar chart
# also specify a y-variable to provide the counts directly
# when reading y values directly, must be a summary table,
#   one row of data for each combination of levels with
#   a numerical value of y
# use lessR pivot function to get summary table, cannot process missing data
#   so set na_show_group to FALSE
d <- Read("Employee")
a <- pivot(d, mean, Salary, c(Dept,Gender), na_group_show=FALSE)
BarChart(Dept, Salary_mean, by=Gender, data=a)
# do so just with BarChart, display bars in grayscale
```

```
# How does average salary vary by gender across the various departments?
BarChart(Dept, Salary, by=Gender, stat="mean", data=d, fill="grays")


# -----------
# annotations
# -----------

d <- rd("Employee")

# Place a message in the center of the plot
# \n indicates a new line
BarChart(Dept, add="Employees by\nDepartment", x1=3, y1=10)

# Use style to change some parameter values
style(add_trans=.8, add_fill="gold", add_color="gold4", add_lwd=0.5)
# Add a rectangle around the message centered at <3,10>
BarChart(Dept, add=c("rect", "Employees by\nDepartment"),
                     x1=c(2,3), y1=c(11, 10), x2=4, y2=9)
```

---

corCFA *Confirmatory Factor Analysis of a Multiple Indicator Measurement Model*

Description

Abbreviation: `cfa`

A multiple indicator measurement model partitions a set of indicators, such as items on a survey, into mutually exclusive groups with one common factor per group of indicators. From the input correlation matrix of the indicator variables, this procedure uses iterated centroid estimation to estimate the coefficients of the model, the factor pattern and factor-factor correlations, as well as the correlations of each factor with each indicator. The analysis is an adaptation and extension of John Hunter's program PACKAGE (Hunter and Cohen, 1969).

Corresponding scale reliabilities are provided, as well as the residuals, the difference between the indicator correlations and those predicted by the model. To visualize the relationships, a heat map of the re-ordered correlation matrix is also provided, with indicator communalities in the diagonal. To understand the meaning of each factor, the corresponding indicator content is displayed for each factor if the indicators have been read as variable labels. Also provides the code to obtain the maximum likelihood solution of the corresponding multiple indicator measurement model (MIMM) with the `cfa` function from the `lavaan` package.

The `scales` is a wrapper that retains 1's in the diagonal of the indicator correlation matrix, so provides scale reliabilities and observed indicator-scale and scale-scale correlations.

Output is generated into distinct pieces by topic, organized and displayed in sequence by default. When the output is assigned to an object, such as `f` in `f <- cfa(Fac =~ X1 + X2 + X3)`, the full or partial output can be accessed for later analysis and/or viewing. A primary such analysis is with `knitr` for dynamic report generation, run from, for example, `RStudio`. The input instructions written to the R~Markdown file are written comments and interpretation with embedded R code. Doing a knitr analysis is to "knit" these comments and subsequent output together so that the R output is embedded in the resulting document, either html, pdf or Word, by default with explanation and interpretation. Generate a complete R~Markdown set of instructions ready to knit from the Rmd option. Simply specify the option and create the file and then open in RStudio and click the knit button to create a formatted document that consists of the statistical results and interpretative comments. See the following sections arguments, value and examples for more information.

Usage

```
corCFA(mimm=NULL, R=mycor, data=d, fac.names=NULL,

         Rmd=NULL, explain=getOption("explain"),
         interpret=getOption("interpret"), results=getOption("results"),

         labels=c("include", "exclude", "only"),

         min_cor=.10, min_res=.05, iter=50, grid=TRUE,

         resid=TRUE, item_cor=TRUE, sort=TRUE,

         main=NULL, heat_map=TRUE, bottom=NULL, right=NULL,

         pdf_file=NULL, width=5, height=5,

         F1=NULL, F2=NULL, F3=NULL, F4=NULL, F5=NULL,
         F6=NULL, F7=NULL, F8=NULL, F9=NULL, F10=NULL,
         F11=NULL, F12=NULL, F13=NULL, F14=NULL, F15=NULL,
         F16=NULL, F17=NULL, F18=NULL, F19=NULL, F20=NULL,

         fun_call=NULL, ...)

cfa(...)

scales(..., iter=0, resid=FALSE, item_cor=FALSE, sort=FALSE, heat_map=FALSE)
```

Arguments

mimm	Multiple indicator measurement model, a character string with the specification of each factor on a separate line: the factor name, an equals sign, and the indicators separated by plus signs. Each indicator is assigned to only one factor.
R	Correlation matrix to be analyzed.
data	Data frame of the original data to be checked for any variable labels, usually indicator (item) content. This is not to calculate correlations, which is separately provided for by the lessR function Correlation.
fac.names	Optional factor names for the original, non-lavaan model specification.
Rmd	File name for the file of R Markdown instructions to be written, if specified. The file type is .Rmd, which automatically opens in RStudio, but it is a simple text

file that can be edited with any text editor, including RStudio.

`explain`	If set to `FALSE` the explanations of the results are not provided in the R~Markdown file. Set globally with options(explain=FALSE).
`interpret`	If set to `FALSE` the interpretations of the results are not provided in the R~Markdown file. Set globally with options(interpret=FALSE).
`results`	If set to `FALSE` the results are not provided in the R~Markdown file, relying upon the interpretations. Set globally with options(results=FALSE).
`labels`	If "include" or "exclude" then variable labels are displayed (if available) or not, organized by the items within each factor. If "only" then no data analysis performed, only the display of the labels by factor.
`min_cor`	Minimum correlation to display. To display all, set to 0.
`min_res`	Minimum residual to display. To display all, set to 0.
`iter`	Number of iterations for communality estimates.
`grid`	If `TRUE`, then separate items in different factors by a grid of horizontal and vertical lines in the output correlation matrix.
`resid`	If `TRUE`, then calculate and print the residuals.
`item_cor`	If `TRUE`, display the indicator correlations.
`sort`	If `TRUE`, re-order the output correlation matrix so that indicators within each factor are sorted by their factor loadings on their own factor.
`main`	Graph title of heat map. Set to `main=""` to turn off.
`heat_map`	If `TRUE`, display a heat map of the indicator correlations with indicator communalities in the diagonal.
`bottom`	Number of lines of bottom margin of heat map.
`right`	Number of lines of right margin of heat map.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`F1`	Variables that define Factor 1.
`F2`	Variables that define Factor 2.
`F3`	Variables that define Factor 3.
`F4`	Variables that define Factor 4.
`F5`	Variables that define Factor 5.
`F6`	Variables that define Factor 6.
`F7`	Variables that define Factor 7.
`F8`	Variables that define Factor 8.
`F9`	Variables that define Factor 9.
`F10`	Variables that define Factor 10.
`F11`	Variables that define Factor 11.
`F12`	Variables that define Factor 12.

F13	Variables that define Factor 13.
F14	Variables that define Factor 14.
F15	Variables that define Factor 15.
F16	Variables that define Factor 16.
F17	Variables that define Factor 17.
F18	Variables that define Factor 18.
F19	Variables that define Factor 19.
F20	Variables that define Factor 20.
fun_call	Function call. Used internally with knitr to pass the function call when obtained from the abbreviated function call cfa. Not usually invoked by the user.
...	Parameter values_

Details

OVERVIEW
A multiple indicator measurement model defines one or more latent variables, called factors, in terms of mutually exclusive sets of indicator variables, such as items from a questionnaire or survey. That is, each factor is defined by a unique set or group of indicators, and each indicator only contributes to the definition of one factor. Two sets of parameters are estimated by the model, the factor pattern coefficients, the lambda's, and the factor-factor correlations, the phi's. Also estimated here are the correlations of each indicator with the other factors.

INPUT
Unless `labels="only"`, the analysis requires the correlation matrix of the indicators and the specification of the groups of indicators, each of which defines a factor in the multiple indicator measurement model. The default name for the indicator correlation matrix is `mycor`, which is also the default name of the matrix produced by the `lessR` function `Correlation` that computes the correlations from the data, as well as the name of the matrix read by the `lessR` function `corRead` that reads the already computed correlation matrix from an external file.

For versions of `lessR` after 3.3, the correlation matrix computed by `Correlation` is now a list element called `R` within the returned list. For example, `mycor$R` from `mycor <- cr(d)`. The function `corCFA` automatically finds this correlation matrix from just entering the entire list name of the returned list, `mycor`, or the specific location, `mycor$R`, or as a stand-alone numerical matrix as done in versions of `lessR` previous to 3.3.

The data frame from which the correlation matrix was computed is required only if any associated variable labels are listed, organized by the items within each factor. By default, `labels="include"`, these labels are listed as part of the analysis if they are available.

Define the constituent variables, the indicators, of each factor with a listing of each variable by its name in the correlation matrix. Each of the up to 20 factors is named by default F1, F2, etc. If the specified variables of a factor are in consecutive order in the input correlation matrix, the list can be specified by listing the first variable, a colon, and then the last variable. To specify multiple variables, a single variable or a list, separate each by a comma, then invoke the R combine or `c` function, preceded by the factor's name and an equals sign. For example, if the first factor is defined by variables in the input correlation matrix from m02 through m05, and the variable Anxiety, then define the factor in the `corCFA` function call according to `F1=c(m02:m05,Anxiety)`.

OUTPUT
The result of the analysis is the correlation matrix of the indicator variables and resulting factors, plus the reliability analysis of the observed total scores or scale that corresponds to each factor. Each scale is defined as an unweighted composite. The corresponding code to analyze the model with the `cfa` function from the `lavaan` package is also provided with the default maximum likelihood estimation procedure. The comparable `lavaan` solution appears in the column that represents the fully standardized solution, factors and indicators, `Std.all`, the last column of the solution output. If the `lavaan` library is loaded, then explicitly refer to the `lessR` function `cfa` with `lessR::cfa` and the corresponding `lavaan` function with `lavaan::cfa`.

VARIABLE LABELS
To display the indicator content, first read the indicators as variable labels with the `lessR` function `Read`. If this labels data frame exists, then the corresponding variable labels, such as the actual items on a survey, are listed by factor. For more information, see `Read`.

HEAT MAP
To help visualize the overall pattering of the correlations, the corresponding heat map of the item correlation matrix with communalities is produced when `heat_map=TRUE`, the default. As is true of the output correlation matrix, the correlations illustrated in the heat map are also sorted by their ordering within each factor. The corresponding color scheme is dictated by the system setting, according to the `lessR` function `style`. The default color scheme is `blue`.

ESTIMATION PROCEDURE
The estimation procedure is centroid factor analysis, which defines each factor, parallel to the definition of each scale score, as the unweighted composite of the corresponding items for that scale. The latent variables are obtained by replacing the 1's in the diagonal of the indicator variable correlation matrix with communality estimates. These estimates are obtained by iterating the solution to the specified number of iterations according to `iter`, which defaults to 50.

A communality is the percentage of the item's correlation attributable to, in this situation of a multiple indicator measurement model, its one underlying factor. As such, the communality is comparable to the item correlations for items within the same factor, which are also due only to the influence of the one common, underlying factor. A value of 0 for `iter` implies that the 1's remain in the observed variable correlation matrix, which then means that there are no latent factors defined. Instead the resulting correlation matrix is of the observed scale scores and the component items.

Value

TEXT OUTPUT
`out_labels`: variables in the model
`out_reliability`: reliability analysis with alpha and omega
`out_indicators`: solution in terms of the analysis of each indicator
`out_solution`: full solution
`out_residuals`: residuals
`out_res_stats`: stats for residuals
`out_lavaan`: lavaan model specification

Separated from the rest of the text output are the major headings, which can then be deleted from custom collations of the output. `out_title_scales`: scales
`out_title_rel`: reliability analysis
`out_title_solution`: solution

`out_title_residuals`: residual analysis
`out_title_lavaan`: lavaan specification

STATISTICS
Returns a list of six components.
1. `ff.cor`: matrix of the factor correlations
2. `if.cor`: matrix of the indicator-factor correlations that includes the estimated pattern coefficients of the model that link a factor to its indicators
3. `diag.cor`: the indicator communalities
4. `alpha`: coefficient alpha for each set of indicators
5. `omega`: if a factor analysis with communality estimates (`iter` > 0), contains coefficient omega for each set of indicators
6. `pred`: matrix of correlations predicted by the model and its estimates 7. `resid`: matrix of raw indicator residuals defined as the observed correlation minus that predicted by the model and its estimates

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapter 11, NY: Routledge.

Gerbing, D. W., & Hamilton, J. G. (1994). The surprising viability of a simple alternate estimation procedure for the construction of large-scale structural equation measurement models. Structural Equation Modeling: A Multidisciplinary Journal, 1, 103-115.

Hunter, J. E., Gerbing, D. W., & Boster, F. J. (1982). Machiavellian beliefs and personality: The construct invalidity of the Machiavellian dimension. Journal of Personality and Social Psychology, 43, 1293-1305.

Hunter, J. & Cohen, J. (1969). PACKAGE: A system of computer routines for the analysis of correlational data. Educational and Psychological Measurement, 1969, 29, 697-700.

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

See Also

`Correlation`.

Examples

```
# perfect input correlation matrix for two-factor model
# Population Factor Pattern of the 3 items for each respective
#   Factor: 0.8, 0.6, 0.4
# Population Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
```

```
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("X1", "X2", "X3", "X4", "X5", "X6")
rownames(mycor) <- colnames(mycor)

# the confirmatory factor analysis
# first three variables with first factor, last three with second
# default correlation matrix is mycor
MeasModel <-
"
   First =~ X1 + X2 + X3
   Second =~ X4 + X5 + X6
"
c <- cfa(MeasModel)

# access the solution directly by saving to an object called fit
cfa(MeasModel)
fit <- cfa(MeasModel)
fit
# get the pattern coefficients from the communalities
lambda <- sqrt(fit$diag.cor)
lambda

# alternative specification described in Gerbing(2014),
#   retained to be consistent with that description
# can specify the items with a colon and with commas
# abbreviated form of function name: cfa
cfa(F1=c(X4,X5,X6), F2=X1:X3)

# component analysis, show observed scale correlations
scales(F1=X1:X3, F2=X4:X6)

# produce a gray scale heat map of the item correlations
#   with communalities in the diagonal
# all subsequent graphics are in gray scale until changed
style("gray")
corCFA(F1=X1:X3, F2=X4:X6)

# access the lessR data set called datMach4
# read the optional variable labels
d <- Read("Mach4", quiet=TRUE)
l <- Read("Mach4_lbl", var_labels=TRUE)
# calculate the correlations and store in mycor
mycor <- cr(m01:m20)
R <- mycor$R
# specify measurement model in Lavaan notation
MeasModel <-
"
   Deceit =~ m07 + m06 + m10 + m09
   Trust =~ m12 + m05 + m13 + m01
   Cynicism =~ m11 + m16 + m04
   Flattery =~ m15 + m02
```

```
"
# confirmatory factor analysis of 4-factor solution of Mach IV scale
# Hunter, Gerbing and Boster (1982)
# generate R Markdown instructions with the option: Rmd
# Output file will be m4.Rmd, a simple text file that can
#   be edited with any text editor including RStudio, from which it
#   can be knit to generate dynamic output such as to a Word document
#c <- cfa(MeasModel, R, Rmd="m4")
# view all the output
#c
# view just the scale reliabilities
#c$out_reliability

# analysis of item content only
cfa(MeasModel, labels="only")


# bad fitting model to illustrate indicator diagnostics
mycor <- corReflect(vars=c(m20))
MeasModel <-
"
   F1 =~ m06 + m09 + m19
   F2 =~ m07
   F3 =~ m04 + m11 + m16
   F4 =~ m15 + m12 + m20 + m18
"
cfa(MeasModel)
```

`corEFA` *Exploratory Factor Analysis and Multiple Indicator Measurement Model*

Description

Abbreviation: `efa`

A maximum likelihood exploratory factor analysis of an input correlation matrix, provided by the standard R exploratory factor analysis `factanal`, which requires the specified number of factors as an input to the analysis. Then constructs the code to run the corresponding multiple indicator measurement model (MIMM) suggested by the exploratory factor analysis loadings in terms of both the `lessR` `corCFA` and the `cfa` function from the `lavaan` package.

Usage

```
corEFA(R=mycor, n_factors, rotate=c("promax", "varimax", "none"),
          min_loading=.2, sort=TRUE, Rmd=NULL, ...)

efa(...)
```

Arguments

`R`	Correlation matrix.
`n_factors`	Number of factors.
`rotate`	Rotation method, if any. Choices are promax (obique) or varimax (orthogonal.
`min_loading`	Minimum loading to include in suggested factor for confirmatory analysis and for the display of the loadings for the exploratory analysis. To ignore, set to 0.
`sort`	Sort the input variables by their highest factor loadings (but only first just list those items with loadings larger than 0.5).
`Rmd`	File name for the file of R markdown to be written, if specified. The file type is .Rmd, which automatically opens in RStudio, but it is a simple text file that can be edited with any text editor, including RStudio.
`...`	Parameter values_

Details

Only the loadings from the exploratory factor analysis are provided, with either an oblique (promax), by default, or an orthogonal (varimax) rotation. If more information is desired, run `factanal` directly.

Also provides the associated multiple indicator measurement model suggested by the exploratory factor analysis. Each MIMM factor is defined by the items that have the highest loading on the corresponding exploratory factor.

For versions of `lessR` after 3.3, the correlation matrix computed by `Correlation` is now a list element called `R` within the returned list. For example, `mycor$R` from `mycor <- cr(d)`. The function `corEFA` automatically finds this correlation matrix from just entering the entire list name of the returned list, `mycor`, or the specific location, `mycor$R`, or as a stand-alone numerical matrix as done in versions of `lessR` previous to 3.3.

Value

The output can optionally be returned and saved into an `R` object, otherwise it simply appears at the console. The components of this object are redesigned in `lessR` version 3.3 into three different types: pieces of text that form the readable output, a variety of statistics, and R markdown instructions. The readable output are character strings such as tables amenable for viewing and interpretation. The statistics are numerical values amenable for further analysis, such as to be referenced in a subsequent R markdown document. The R~Markdown input is available for entry direct into `knitr`, such as in RStudio. The motivation of these three types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object followed by a dollar sign, can be inserted into the R markdown document (see `examples`).

READABLE OUTPUT
`out_title`: Variables in the model, rows of data and retained
`out_loadings`: Estimated coefficients, hypothesis tests and confidence intervals
`out_sum_squares`: Fit indices
`out_cfa_title`: Analysis of variance
`out_ice`: Correlations among all variables in the model
`out_lavaan`: Collinearity analysis

out_deleted: R squared adjusted for all (or many) possible subsets

STATISTICS
Rmd: Instructions to run through knitr, such as copy and paste, to obtain output in the form of a web file, pdf document or Word document. Can also obtain these instructions with the Rmd option, which writes them directly to the specified text file. Obtain a less detailed Rmd file by setting explain=FALSE.

Although not typically needed for analysis, if the output is assigned to an object named, for example, fa, then the complete contents of the object can be viewed directly with the unclass function, here as unclass(fa). Invoking the class function on the saved object reveals a class of out_all. The class of each of the text pieces of output is out.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapter 11, NY: Routledge.

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

See Also

Correlation.

Examples

```
# input correlation matrix of perfect two-factor model
# Factor Pattern for each Factor: 0.8, 0.6, 0.4
# Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("X1", "X2", "X3", "X4", "X5", "X6")
rownames(mycor) <- colnames(mycor)

# default factor analysis of default correlation matrix mycor
#   with two factors extracted
corEFA(n_factors=2)

# abbreviated form
# use all items to construct the MIMM, regardless of their loadings
#  and show all loadings
# show the initial factor extraction
efa(n_factors=2, min_loading=0, show_initial=TRUE)
```

corProp — *Proportionality Coefficients from Correlations*

Description

Abbreviation: `cp`

In the population, indicators of the same factor or latent variable have parallel correlations with all other variables. Of course, in the presence of sampling error, this parallelism will only be approximate. To assess this parallelism, proportionality coefficients are computed for each pair of variables in the input correlation matrix. Also output is a heat map of the resulting matrix of proportionality coefficients. Each graph is based on a default color theme. The original default is `lightbronze`, but other color palettes can be generated as well.

Usage

```
corProp(R=mycor,
          main=NULL, heat_map=TRUE, bottom=NULL, right=NULL,
          pdf_file=NULL, width=5, height=5, ...)

cp(...)
```

Arguments

`R`	Correlation matrix.
`main`	Graph title. Set to `main=""` to turn off.
`heat_map`	If `TRUE`, display a heat map of the item correlations with the diagonal ignored.
`bottom`	Number of lines of bottom margin.
`right`	Number of lines of right margin.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Parameter values_

Details

Proportionality coefficients indicate the extent of proportionality between two variables. Perfect proportionality of two variables is consistent with both variables being indicators of the same latent variable or factor and indicators of no other factor.

In the current version the diagonal of the input correlation matrix is ignored. To maintain parallelism, the diagonal element of 1.00 would need to be replaced the corresponding communalities, which first requires a factor analysis.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapter 11, NY: Routledge.

See Also

Correlation.

Examples

```
# input correlation matrix of perfect two-factor model
# Factor Pattern for each Factor: 0.8, 0.6, 0.4
# Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("X1", "X2", "X3", "X4", "X5", "X6")
rownames(mycor) <- colnames(mycor)

# proportionality coefficients of correlation matrix mycor
# indicators of the same factor have proportional correlations
corProp()

# abbreviated form
cp()

# calculate and store proportionality coefficients in myprop
# order the proportionality coefficients to help identify factors
myprop <- corProp()
corReorder(myprop)
```

---

corRead — *Read Specified Correlation Matrix*

Description

Abbreviation: `rd.cor`

A wrapper for base~R `read.table`. Read a correlation matrix into R. All coefficients for each variable must be on one physical row. No variable names are in the file to be read.

Usage

```
corRead(from=NULL, var_names=NULL, ...)

rd.cor(...)
```

Arguments

`from`	File reference, either omitted to browse for the data file, or a full path name or web URL, included in quotes. A URL begins with `http://`.
`var_names`	The names of the variables in the matrix.
`...`	Parameter values for base R `read.table`.

Details

Read a correlation, or any square, matrix into R. All coefficients for each variable must be on one row. No variable names are in the file to be read. The coefficients within each row, that is, for a single variable, are delimited by a white space, such as one or more blanks.

The standard R function that reads the matrix is `read.table`.

By default the variables are named X1, X2, etc. If the `var_names` option is invoked, then the specified names refer to the respective rows and columns of the matrix. Here it may be convenient to name the variables with the `lessR` function `to`.

The alternative is to calculate the correlations from the data, such as with the `lessR` function `Correlation` or the standard R function `cor`.

Author(s)

David W. Gerbing (Portland State University; <`gerbing@pdx.edu`>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapter 8, NY: Routledge.

See Also

`Correlation`, `read.table`.

Examples

```
# browse for the data file because ref is omitted
# name the variables with the lessR function to
# mycor <- corRead(var_names=to("m",20))

# abbreviated form
# read a matrix with 4 variables and specify the names
# mycor <- rd.cor(var_names=c("m06","m07","m09","m10"))
```

`corReflect` *Reflect Specified Variables in a Correlation Matrix*

Description

Abbreviation: `reflect`

Reflects the specified variables by multiplying each correlation of the variable by -1. Usually a prelude to a factor analysis, such as provided by `corCFA`.

Usage

```
corReflect(R=mycor, vars,
           main=NULL, heat_map=TRUE, bottom=NULL,right=NULL,
           pdf_file=NULL, width=5, height=5, ...)

reflect(...)
```

Arguments

`R`	Correlation matrix.
`vars`	List of the re-ordered variables, each variable listed by its ordinal position in the input correlation matrix.
`main`	Graph title. Set to `main=""` to turn off.
`heat_map`	If `TRUE`, display a heat map of the item correlations with item communalities in the diagonal.
`bottom`	Number of lines of bottom margin.
`right`	Number of lines of right margin.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Parameter values_

Details

Reflects the specified variables by multiplying each correlation of the variable by -1. The original data from which the correlations are computed is unmodified unless the output of the function is written into the input correlation matrix, by default `mycor`.

Define the constituent variables, the items, with a listing of each variable by its name in the correlation matrix. If the specified variables are in consecutive order in the input correlation matrix, the list can be specified by listing the first variable, a colon, and then the last variable. To specify multiple variables, a single variable or a list, separate each by a comma, then invoke the R combine or `c` function. For example, if the list of variables in the input correlation matrix is from m02 through m05, and the variable Anxiety, then define the list in the `corReflect` function call according to `vars=c(m02:m05,Anxiety)`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Correlation, recode.

Examples

```
# input correlation matrix of perfect two-factor model
# Factor Pattern for each Factor: 0.8, 0.6, 0.4
# Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("V1", "V2", "V3", "V4", "V5", "V6")
rownames(mycor) <- colnames(mycor)

# reflect all 3 indicators of the second factor
mynewcor <- corReflect(vars=c(V4,V5,V6))

# abbreviated form
# replace original mycor
mycor <- reflect(vars=c(V4,V5,V6))
```

Correlation *Correlation Analysis*

Description

Abbreviation: cr, cr_brief

For two variables, yields the correlation coefficient with hypothesis test and confidence interval. For a data frame or list of variables from a data frame, yields the correlation matrix. The default computed coefficient(s) are the standard Pearson's product-moment correlation, with Spearman and Kendall coefficients available. For the default missing data technique of pairwise deletion, an analysis of missing data for each computed correlation coefficient is provided. For a correlation matrix a statistical summary of the missing data across all cells is provided.

Versions of this function from lessR 3.3 or earlier returned just a correlation matrix. Now other values are returned as well so that the correlation matrix is now stored as part of a returned list in R, directly available, for example, as mycor$R from mycor <- cr(d). This revision is automatically adjusted for in the lessR routines that read the subsequent correlation matrix, so all pre-existing code continues to work. That is, the input into any of these routines could be, for example, mycor, mycor$R or a stand-alone correlation matrix such as in pre-lessR 3.3.

Usage

```
Correlation(x, y, data=d,
         miss=c("pairwise", "listwise", "everything"),
         fill_low=NULL, fill_hi=NULL,
         show_n=NULL, brief=FALSE,
         digits_d=NULL, heat_map=TRUE,
         main=NULL, bottom=3, right=3,
         pdf=FALSE, width=5, height=5, ...)

cr_brief(..., brief=TRUE)

cr(...)
```

Arguments

`x`	First variable, or list of variables for a correlation matrix.
`y`	Second variable or not specified if the first argument is a list.
`data`	Optional data frame that contains the variables of interest, default is `d`.
`miss`	Basis for deleting missing data values_
`fill_low`	Starting color for a custom sequential palette.
`fill_hi`	Ending color for a custom sequential palette.
`show_n`	For pairwise deletion, show the matrix of sample sizes for each correlation coefficient, regardless of sample size.
`brief`	Pertains to a single correlation coefficient analysis. If `FALSE`, then the sample covariance and number of non-missing and missing observations are displayed.
`digits_d`	Specifies the number of decimal digits to display in the output.
`heat_map`	If `TRUE`, generate a heat map.
`main`	Graph title of heat map. Set to `main=""` to turn off.
`bottom`	Number of lines of bottom margin of heat map.
`right`	Number of lines of right margin of heat map.
`pdf`	If `TRUE`, generate the heat map and write to pdf files.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values for internally called functions, which include `method="spearman"` and `method="kendall"` and also `alternative="less"` and `alternative="more"`.

Details

When two variables are specified, both `x` and `y`, the output is the correlation coefficient with hypothesis test, for a null hypothesis of 0, and confidence interval. Also displays the sample covariance. Based on R functions `cor`, `cor.test`, `cov`.

In place of two variables `x` and `y`, `x` can be a complete data frame, either specified with the name of a data frame, or blank to rely upon the default data frame `d`. Or, `x` can be a list of variables from the input data frame. In these situations y is missing. Any non-numeric variables in the data frame or specified variable list are automatically deleted from the analysis.

When heat_map=TRUE, generate a heat map to standard graphics windows. Set pdf=TRUE to generate these graphics but have them directed to their respective pdf files.

For treating missing data, the default is pairwise, which means that an observation is deleted only for the computation of a specific correlation coefficient if one or both variables are missing the value for the relevant variable(s). For listwise deletion, the entire observation is deleted from the analysis if any of its data values are missing. For the more extreme everything option, any missing data values for a variable result in all correlations for that variable reported as missing.

Value

From versions of lessR of 3.3 and earlier, if a correlation matrix is computed, the matrix is returned. Now more values are returned, so the matrix is embedded in a list of returned elements.

READABLE OUTPUT

single coefficient
out_background: Variables in the model, any variable labels
out_describe: Estimated coefficients
out_inference: Hypothesis test and confidence interval estimated coefficient

matrix
out_background: Variables in the model, any variable labels
out_missing: Missing values analysis
out_cor: Correlations

STATISTICS

single coefficient
r: Model formula that specifies the model
tvalue: t-statistic of estimated value of null hypothesis of no relationship
df: Degrees of freedom of hypothesis test pvalue: Number of rows of data submitted for analysis
lb: Lower bound of confidence interval
ub: Upper bound of confidence interval

matrix
R: Correlations

Usually assign the name of mycor to the output matrix, as in following examples. This matrix is ready for input into any of the lessR functions that analyze correlational data, including confirmatory factor analysis by corCFA and also exploratory factor analysis, either the standard R function factanal or the lessR function corEFA

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapter 10, NY: Routledge.

See Also

cor.test, cov.

Examples

```
# data
n <- 12
f <- sample(c("Group1","Group2"), size=n, replace=TRUE)
x1 <- round(rnorm(n=n, mean=50, sd=10), 2)
x2 <- round(rnorm(n=n, mean=50, sd=10), 2)
x3 <- round(rnorm(n=n, mean=50, sd=10), 2)
x4 <- round(rnorm(n=n, mean=50, sd=10), 2)
d <- data.frame(f,x1, x2, x3, x4)
rm(f); rm(x1); rm(x2); rm(x3); rm(x4)

# correlation and covariance
Correlation(x1, x2)
# short name
cr(x1, x2)
# brief form of output
cr_brief(x1, x2)

# Spearman rank correlation, one-sided test
Correlation(x1, x2, method="spearman", alternative="less")

# correlation matrix of the numerical variables in mycor
mycor <- Correlation()

# correlation matrix of Kendall's tau coefficients
mycor <- cr(method="kendall")

# correlation matrix of specified variables in mycor with heat_map
mycor <- Correlation(x1:x3, heat_map=TRUE)

# analysis with data not from data frame mycor
data(attitude)
mycor <- Correlation(rating, learning, data=attitude)

# analysis of entire data frame that is not mycor
data(attitude)
mycor <- Correlation(attitude)
```

corReorder *Reorder Variables in a Correlation Matrix*

Description

Abbreviation: `reord`

Re-arranges the order of the variables in the input correlation matrix. If no variable list is specified then by default the variables are re-ordered according to hierarchical clustering. Or, re-order with the Hunter (1973) chain method in which the first variable is the variable with the largest sum of squared correlations of all the variables, then the next variable is that with the highest correlation with the first variable, and so forth. Or, re-order manually.

Usage

```
corReorder(R=mycor, order=c("hclust", "chain", "manual", "as_is"),
          hclust_type = c("complete", "ward.D", "ward.D2", "single",
                          "average", "mcquitty", "median", "centroid"),
          dist_type=c("R", "dist"),
          n_clusters=NULL, vars=NULL, chain_first=0,
          heat_map=TRUE, dendrogram=TRUE, diagonal_new=TRUE,
          main=NULL, bottom=NULL, right=NULL,
          pdf=FALSE, width=5, height=5, ...)

reord(...)
```

Arguments

`R`	Correlation matrix.
`order`	Source of ordering (seriation): Default of hierarchical cluster analysis, Hunter(1973) chain method, manually specified with `vars`, or left `"as_is"`.
`hclust_type`	Type of hierarchical cluster analysis.
`dist_type`	Default is a correlation matrix of similarities, otherwise a distance matrix.
`n_clusters`	For a hierarchical cluster analysis, optionally specify the cluster membership for the specified number of clusters.
`vars`	List of the re-ordered variables, each variable listed by its ordinal position in the input correlation matrix. If this is set, then `order` set to `"manual"`.
`chain_first`	The first variable listed in the ordered matrix with the chain method.
`main`	Graph title. Set to `main=""` to turn off.
`heat_map`	If `TRUE`, display a heat map of the item correlations.
`dendrogram`	If `TRUE`, display a heat map of the item correlations for a hierarchical cluster analysis.
`diagonal_new`	If `TRUE`, replace diagonal for the heat map only with an average of the correlation of item on the diagonal with the two adjacent items.

`bottom`	Number of lines of bottom margin.
`right`	Number of lines of right margin.
`pdf`	Set to `TRUE` if graphics files written to pdf, the heat map of the re-ordered matrix, and, if an hierarchical cluster analysis, the dendrogram.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Parameter values_

Details

Reorder and/or delete variables in the input correlation matrix.

Define the constituent variables, the items, with a listing of each variable by its name in the correlation matrix. If the specified variables are in consecutive order in the input correlation matrix, the list can be specified by listing the first variable, a colon, and then the last variable. To specify multiple variables, a single variable or a list, separate each by a comma, then invoke the R combine or `c` function. For example, if the list of variables in the input correlation matrix is from m02 through m05, and the variable Anxiety, then define the list in the `corReorder` function call according to `vars=c(m02:m05,Anxiety)`.

Or, define the ordering with a hierarchical cluster analysis from the base R function `hclust()`. The same default type of `"complete"` is provided, though this can be changed with the parameter `hclust_type` according to `hclust`. Default input is a correlation matrix, converted to a matrix of dissimilarities by subtracting each element from 1.

Or, use the Hunter (1973) chain method. Define the ordering of the variables according to the following algorithm. If no variable list is specified then the variables are re-ordered such that the first variable is that which has the largest sum of squared correlations of all the variables, then the variable that has the highest correlation with the first variable, and so forth.

In the absence of a variable list, the first variable in the re-ordered matrix can be specified with the `chain_first` option.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Hunter, J.E. (1973), Methods of reordering the correlation matrix to facilitate visual inspection and preliminary cluster analysis, Journal of Educational Measurement, 10, p51-61.

See Also

`Correlation`, `hclust`.

Examples

```
# input correlation matrix of perfect two-factor model
# Factor Pattern for each Factor: 0.8, 0.6, 0.4
# Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("V1", "V2", "V3", "V4", "V5", "V6")
rownames(mycor) <- colnames(mycor)

# leave only the 3 indicators of the second factor
#  in reverse order
#replace original mycor
mycor <- corReorder(vars=c(V6,V5,V4))

#  reorder according to results of a hierarchical cluster analysis
mynewcor <- corReorder()

#  get cluster membership for two clusters
# specify each parameter
mynewcor <- corReorder(mycor, order="hclust", n_clusters=2)

#  reorder with first variable with largest sums of squares
mynewcor <- corReorder(order="chain")

# reorder the variables according to the ordering algorithm
#  with the 4th variable listed first
# no heat map
mynewcor <- corReorder(chain_first=2, heat_map=FALSE)

mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("V1", "V2", "V3", "V4", "V5", "V6")
rownames(mycor) <- colnames(mycor)

# can also re=order with index position of each variable
mycor <- corReorder(vars=c(4,5,6,1,2,3))
```

`corScree` *Eigenvalue Plot of a Correlation Matrix*

Description

Abbreviation: `scree`

Plots the successive eigenvalues of an input correlation matrix. Also plots the successive differences of the eigenvalues. The purpose is usually to help determine the number of factors that explain the correlations in a correlation matrix. So usually a prelude to an exploratory factor analysis, such as provided by the `lessR` function `corEFA`. This program relies upon the standard R exploratory factor analysis `factanal`, which requires the specified number of factors as an input to the analysis.

Usage

```
corScree(R=mycor,
          main=NULL, pdf=FALSE, width=5, height=5, ...)

scree(...)
```

Arguments

`R`	Correlation matrix.
`main`	Graph title, which is blank by default.
`pdf`	Indicator as to if the graphic files should be saved as pdf files instead of directed to the standard graphics windows.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Parameter values_

Details

Interpretation of the scree plot to assist in the assessment of the number of factors that account for the structure of a correlation matrix depends primarily on the analysis of the differences between the successive eigenvalues_ The differences begin to diminish where the "scree" begins, analogous to the debris that falls off of a hill top. Accordingly both the scree plot itself, the plot of the successive eigenvalues, and the plot of the differences of the successive eigenvalues are presented.

PDF OUTPUT
Because of the customized graphic windowing system that maintains a unique graphic window for the Help function, the standard graphic output functions such as `pdf` do not work with the `lessR` graphics functions. Instead, to obtain pdf output, use the `pdf_file` option, perhaps with the optional `width` and `height` options. These files are written to the default working directory, which can be explicitly specified with the R `setwd` function.

If the two plots, of the population and sample distributions respectively, are written to pdf files, according to `pdf=TRUE`, they are named Scree.pdf and ScreeDiff.pdf. Their names and the directory to which they are written are provided as part the console output.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapters 9 and 10, NY: Routledge.

See Also

Correlation.

Examples

```
# input correlation matrix of perfect two-factor model
# Factor Pattern for each Factor: 0.8, 0.6, 0.4
# Factor-Factor correlation: 0.3
mycor <- matrix(nrow=6, ncol=6, byrow=TRUE,
c(1.000,0.480,0.320,0.192,0.144,0.096,
  0.480,1.000,0.240,0.144,0.108,0.072,
  0.320,0.240,1.000,0.096,0.072,0.048,
  0.192,0.144,0.096,1.000,0.480,0.320,
  0.144,0.108,0.072,0.480,1.000,0.240,
  0.096,0.072,0.048,0.320,0.240,1.000))
colnames(mycor) <- c("V1", "V2", "V3", "V4", "V5", "V6")
rownames(mycor) <- colnames(mycor)

# obtain the scree plots
corScree()

# abbreviated form
scree()
```

`CountAll` *CountAll Descriptive Analysis of all Variables in the Data Frame*

Description

Automatically call the following functions in this package: `SummaryStats`, `Histogram` and `BarChart`. The result is set of summary statistics for every variable in the data frame, by default called `d`, a histogram for each numerical variable and a bar chart for each categorical variable.

Usage

```
CountAll(x=d, quiet=FALSE, ...)

ca(...)
```

Arguments

`x`	Data frame that contains the variables to analyze, by default `d`.
`quiet`	Suppress text output if `TRUE`.
`...`	Other parameter values for graphics.

Details

CountAll is designed to work in conjunction with the lessR function Read, which reads a csv or other formatted data file into the data frame d. All the bar charts and associated summary statistics are written to one file and all the histograms and associated summary statistics for the numerical variables are written to another file.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

SummaryStats, Histogram, BarChart.

Examples

```
# create data frame called d
n <- 12
X <- sample(c("Group1","Group2"), size=n, replace=TRUE)
Y <- rnorm(n=n, mean=50, sd=10)
d <- data.frame(X,Y)
rm(X); rm(Y);

# CountAll descriptive analysis of d
CountAll()
# short name
ca()
```

---

dataAnova_1way   *Data for a One-Way ANOVA*

Description

To study the impact of arousal on the ability to complete a task, 24 laboratory rats were randomly and equally divided into three groups of eight. Each rat was administered one of three dosages of an arousal inducing drug: 0, 5, and 10 milligrams. Following the dosage, each rat completed a maze to obtain a food reward. The response (dependent) variable is the Time in seconds to complete the maze.

Format

A data table with 24 rows of data and 2 columns, with variables Dosage and Time.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Source

author

References

Gerbing, D. W. (later in 2022). R Data Analysis without Programming, 2nd Edition, Chapter 7, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Examples

```
d <- Read("Anova_1way")
ANOVA(Time ~ Dosage)
```

---

dataAnova_2way *Data for a Two-Way Balanced Factorial Design*

Description

Laboratory rats were randomly and equally divided into groups, and then given one of three dosages of an arousal inducing drug: 0, 5, and 10 milligrams. Following the dosage, each rat completed either an easy or a hard maze to obtain a food reward. The response (dependent) variable is the Time in seconds to complete the maze.

Format

A data table with 48 rows of data and 3 columns: Difficulty, Dosage, and Time.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Source

author

References

Gerbing, D. W. (later in 2022). R Data Analysis without Programming, 2nd Edition, Chapter 7, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Examples

```
d <- Read("Anova_2way")
ANOVA(Time ~ Dosage * Difficulty)
```

---

dataAnova_rb *Data for a Randomized Block ANOVA*

Description

Seven people, with differing amounts of muscle strength, took one of four different pre-workout supplements and then did a bench press of 125 lbs as many times as possible. Each person did four workouts at four different times, with a different supplement before each workout. The experimenter randomized the presentation order of the supplements to each person to avoid any artifacts from presentation order.

Format

A data table with 7 rows of data and 5 columns: Person, and sup1 through sup4 for the four supplements.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Source

author

References

Gerbing, D. W. (later in 2022). R Data Analysis without Programming, 2nd Edition, Chapter 7, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Examples

```
d <- Read("Anova_rb")
d <- reshape_long(d, sup1:sup4, group="Supplement", response="Reps")
ANOVA(Reps ~ Supplement + Person)
```

`dataAnova_rbf` *Data for a Randomized Block Factorial ANOVA*

Description

The data for this randomized blocks factorial is a partitioning of 48 rats into 8 groups of 6 based on an initial assessment of each rat's ability to navigate a maze. That is, some rats in general do better than others. A trial maze served as a sort of a pre-test in which the rats were sorted on the basis of their ability to solve the maze. The first block of 6 rats ran the trial maze the fastest, and the last block the slowest. Within each block the rats were randomly assigned to each of the 6 treatment combinations. Each block of matched rats provides a score on each of the six treatment combinations.

This design is within-subjects because similar rats in terms of maze running ability provide the data for each block of data values. Each rat in this block only experiences one of the 6 cells, but all the rats in a block are evaluated across all 6 combinations of the levels of the two treatment variables.

Format

A data table in wide format with 48 rows of data and 4 columns: Difficulty, Dosage, Block, and Time.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Source

author

References

Gerbing, D. W. (later in 2022). R Data Analysis without Programming, 2nd Edition, Chapter 7, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Examples

```
d <- Read("Anova_rbf")
fit <- aov(Time ~ (Dosage*Difficulty) + Error(Block), data=d)
summary(fit)
```

dataAnova_sp *Data for a Split-Plot ANOVA*

Description

A study of four different pre-workout supplements analyzed their effectiveness in terms of the number of repetitions of a given exercise and weight. Each of 14 participants were randomly assigned to one of two groups: Hi quality Food, a nutritious breakfast, and Low quality Food, a less nutritious breakfast. Each group of 7 participants took all four Supplements, each in randomized order, one for each workout. The result is a total of 28 data values for each group, 56 data values overall.

Type of Supplements is a within- groups treatment variable. The other treatment variable, Food quality, is a between-groups treatment variable.

Format

A data table with 56 rows of data and 4 columns: Person, Food, Supplement, and Reps.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Source

author

References

Gerbing, D. W. (later in 2022). R Data Analysis without Programming, 2nd Edition, Chapter 7, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Examples

```
d <- Read("Anova_sp")
fit <- aov(Reps ~ (Food*Supplement) + Error(Person/Food), data=d)
summary(fit)
```

dataBodyMeas — *Data: Body Measurements*

Description

Body measurements of 170 women and 170 men who purchased motorcycle clothing.

Usage

```
data(dataBodyMeas)
```

Format

A data table with 340 observations and the following 7 variables.

Gender, "M" or "F" (factor)
Weight (integer in pounds)
Height (integer in inches)
Waist (integer in inches)
Hips (integer in inches)
Chest (integer in inches)
Hand (numeric, circumference of hand in inches to nearest quarter of an inch)
Shoe (numeric, size including half sizes)

Source

author

dataCars93 — *Data: Cars93*

Description

1993 New Car Data.

Usage

```
data(dataCars93)
```

Format

A data table with 93 observations and 25 variables.

Variables
——————————————————-
Make: Model
Type: Small, Sporty, Compact, Midsize, Large, Van
MinPrice: Minimum Price (in $1,000) - Price for basic version of this model
MidPrice: Midrange Price (in $1,000) - Average of Min and Max prices
MaxPrice: Maximum Price (in $1,000) - Price for a premium version
MPGcity: City MPG
MPGhiway: Highway MPG
Airbags: 0 = none, 1 = driver only, 2 = driver & passenger
DriveTrain: 0 = rear wheel drive, 1 = front wheel drive, 2 = all wheel drive
Cylinders: Number of cylinders
Engine: Engine size (liters)
HP: maximum Horsepower
RPM: revolutions per minute at maximum horsepower
RevMile: Engine revolutions per mile in highest gear
Manual: Manual transmission available, 0 = No, 1 = Yes
FuelCap: Fuel tank capacity (gallons)
PassCap: Passenger capacity (persons)
Length: Length (inches)
Wheelbase: Wheelbase (inches)
Width: Width (inches)
Uturn: U-turn space (feet)
RearSeat: Rear seat room (inches)
LugCap: Luggage capacity (cu. ft.)
Weight: Weight (pounds)
Source: 0=non-USA manufacturer, 1=USA manufacturer

Source

Lock, R. H. (1993). 1993 new car data. Journal of Statistics Education, 1(1).

`dataEmployee` *Data: Employees*

Description

Some human resource data on 37 employees with 6 variables. Variable labels and variable units are included in the data file.

Usage

```
data(dataEmployee)
```

Format

A data table with 37 observations.

Years,"Years Employed in the Company"
Gender,"Male or Female"
Dept,"Department Employed"
Salary,"Annual Salary (USD)"
JobSat,"JobSat with Work Environment"
Plan,"1=GoodHealth, 2=YellowCross, 3=BestCare"
Pre,"Test score on legal issues before instruction"
Post,"Test score on legal issues after instruction"

Source

author

---

dataEmployee_lbl *VariableLabels: Employee Data Set*

Description

For the data on 37 employees with 6 variables, includes the variable labels and variable units.

Usage

```
data(dataEmployee_lbl)
```

Format

Variable labels, and some unites.

Years,"Years Employed in the Company"
Gender,"Male or Female"
Dept,"Department Employed"
Salary,"Annual Salary (USD)"
JobSat,"JobSat with Work Environment"
Plan,"1=GoodHealth, 2=YellowCross, 3=BestCare"
Pre,"Test score on legal issues before instruction"
Post,"Test score on legal issues after instruction"

Source

author

`dataFreqTable99` *Data: Joint Frequency Table*

Description

Based on a survey of university students, the joint frequencies for two variables are reported. One variable is Race and the other is undergraduate Class.

`Level Asian Latino Black White FR 33 58 6 105 SO 41 79 9 207 JR 86 179 27 484 SR 143 214 31 824`

The data file consists just of the frequencies, the numbers, without any labels.

Usage

```
data(dataFreqTable99)
```

Format

A table of joint frequencies or Race and Level.

Race: Asian, Latino, Black, White
Class: FR, SO, JR, SR

Source

author

`dataJackets` *Data: Motorcycle Type and Thickness of Jacket*

Description

Two variables, one is type of motorcycle and the other is the thickness of the purchased jacket.

Usage

```
data(dataJackets)
```

Format

A data table with 1025 observations.

Bike,"Type of Motorcycle, Honda or BMW"
Jacket,"Lite, Med or Thick"

Source

author

dataLearn — *Data: Distributed vs Massed Practice*

Description

Completely Randomized design, one-factor with two levels (CR-2): One grouping variable that specifies type of learning, distributed or massed practice, and one response variable, Learning.

Usage

```
data(dataLearn)
```

Format

A data table with 34 observations.

Source

author

dataMach4 — *Data: Machiavellianism*

Description

Likert data responses to Christie and Geiss's (1970) Mach~IV scale from Hunter, Gerbing and Boster (1982).

All Likert items assessed on a 6-point scale from 0: Strongly Disagree to 5: Strongly Agree. Variable labels, the item content, are included.

To construct composite scale scores, such as the Mach~IV total score, the following items should first be reverse scored: m03, m04, m06, m07, m09, m10, m11, m14, m16, m17, m19.

Usage

```
data(dataMach4)
```

Format

A data table with 351 observations.

Gender, 1 column, 0:Male, 1:Female
Mach IV, 20 Likert items: m01, m02, . . . , m20, see `dataMach4_lbl` for the item content.

Source

author

References

Christie, R., & Geis, F. L., (1970). Studies in Machiavellianism. New York: Academic Press.

Hunter, J. E., Gerbing, D. W., and Boster, F. J. (1982). Machiavellian beliefs and personality: The construct invalidity of the Machiavellian dimension. Journal of Personality and Social Psychology, 43, 1293-1305.

Examples

```
# Read data and variable labels (items)
d <- Read("Mach4")
l <- Read("Mach4_lbl")

# Convert to factors, i.e., categorical with value labels
d <- factors(m01:m20,
            levels=0:5,
            labels=c("Strongly Disagree", "Disagree", "Slightly Disagree",
                     "Slightly Agree", "Agree", "Strongly Agree"))
```

`dataMach4_lbl` *VariableLabels: Mach4 Data Set*

Description

For the data of 351 responses to the 20-item Mach IV scale.

Usage

```
data(dataMach4_lbl)
```

Format

Variable labels, the items of the Christie and Geiss Mach IV Scale

m01: Never tell anyone the real reason you did something unless it is useful to do so
m02: The best way to handle people is to tell them what they want to hear
m03: One should take action only when sure it is morally right
m04: Most people are basically good and kind
m05: It is safest to assume that all people have a vicious streak and it will come out when they are given a chance
m06: Honesty is the best policy in all cases
m07: There is no excuse for lying to someone else
m08: Generally speaking, people won't work hard unless they're forced to do so
m09: All in all, it is better to be humble and honest than to be important and dishonest
m10: When you ask someone to do something for you, it is best to give the real reasons for wanting it rather than giving reasons which carry more weight
m11: Most people who get ahead in the world lead clean, moral lives
m12: Anyone who completely trusts anyone else is asking for trouble

m13: The biggest difference between most criminals and other people is that the criminals are stupid enough to get caught
m14: Most people are brave
m15: It is wise to flatter important people
m16: It is possible to be good in all respects
m17: Barnum was wrong when he said that there's a sucker born every minute
m18: It is hard to get ahead without cutting corners here and there
m19: People suffering from incurable diseases should have the choice of being put painlessly to death
m20: Most people forget more easily the death of a parent than the loss of their property

Source

author

References

Christie, R., & Geis, F. L., (1970). Studies in Machiavellianism. New York: Academic Press.

Hunter, J. E., Gerbing, D. W., and Boster, F. J. (1982). Machiavellian beliefs and personality: The construct invalidity of the Machiavellian dimension. Journal of Personality

Examples

```
# Read data and variable labels (items)
d <- Read("Mach4")
l <- Read("Mach4_lbl")

# Convert to factors, i.e., categorical with value labels
d <- factors(m01:m20,
            levels=0:5,
            labels=c("Strongly Disagree", "Disagree", "Slightly Disagree",
                     "Slightly Agree", "Agree", "Strongly Agree"))
```

---

dataReading *Data: Reading Ability*

Description

Reading ability test score and also verbal aptitude test score, number of absences from school and family income in USD $1000's. Data are simulated.

Usage

```
data(dataReading)
```

Format

A data table with 100 observations.

Source

author

`dataStockPrice` *Data: Stock price of Apple, IBM and Intel from 1985 through May of 2024*

Description

Monthly adjusted stock price of Apple, IBM and Intel from 1985 through May of 2024 from finance.yahoo.com.

Usage

```
data(dataStockPrice)
```

Format

A data table in long format with four variables: Month, Company, Price, and Volume. The variable Month is a date expression expressed in the ISO standard as a four-digit year, followed by the two-digit month, then the two-digit day, separated by dashes. A total of 1419 rows, 473 rows per company.

Source

author

`dataWeightLoss` *Data: WeightLoss*

Description

The weights of 10 people were recorded. Then they entered a weight loss program. Following completion of the program, their weights were once again recorded. Data are simulated.

Usage

```
data(dataWeightLoss)
```

Format

A data table with 10 observations.

Source

author

Density *Density Curves from Data plus Histogram*

Description

Abbreviation: `dn`

«<DEPRECATED in favor of Histogram(x, density=TRUE) »>

Plots a normal density curve and/or a general density curve superimposed over a histogram, all estimated from the data. Also reports the Shapiro-Wilk normality test and summary statistics.

If the provided object to analyze is a set of multiple variables, including an entire data frame, then each non-numeric variable in the data frame is analyzed and the results written to the current graphics device or to a pdf file in the current working directory. The name of each output pdf file that contains a bar chart and its path are specified in the output.

When output is assigned into an object, such as `d` in `d <- dn(Y)`, the pieces of output can be accessed for later analysis. A primary such analysis is `knitr` for dynamic report generation from an R markdown document in which R output is embedded in documents, facilitated by the `Rmd` option. See `value` below.

Usage

```
Density(x, data=d, rows=NULL,
        n_cat=getOption("n_cat"), Rmd=NULL,

       bw=NULL, type=c("general", "normal", "both"),
       histogram=TRUE, bin_start=NULL, bin_width=NULL,

       color_nrm="gray20", color_gen="gray20",
       fill_nrm=NULL, fill_gen=NULL,

       rotate_x=0, rotate_y=0, offset=0.5,

       x.pt=NULL, xlab=NULL, main=NULL, sub=NULL, y_axis=FALSE,
       x.min=NULL, x.max=NULL,
       rug=FALSE, color_rug="black", size_rug=0.5,

       eval_df=NULL, digits_d=NULL, quiet=getOption("quiet"),
       width=4.5, height=4.5, pdf_file=NULL,
       fun_call=NULL, ...)

dn(...)
```

Arguments

`x` Variable(s) to analyze. Can be a single numerical variable, either within a data frame or as a vector in the user's workspace, or multiple variables in a data frame

	such as designated with the c function, or an entire data frame. If not specified, then defaults to all numerical variables in the specified data frame, d by default.
data	Optional data frame that contains the variable(s) of interest, default is d.
rows	A logical expression that specifies a subset of rows of the data frame to analyze.
n_cat	For the analysis of multiple variables, such as a data frame, specifies the largest number of unique values of variable of a numeric data type for which the variable will be analyzed as categorical. Default is 0.
Rmd	File name for the file of R markdown to be written, if specified. The file type is .Rmd, which automatically opens in RStudio, but it is a simple text file that can be edited with any text editor, including RStudio.
bw	Bandwidth of kernel estimation. Initial value is "nrd0", but unless specified, then may be iterated upward to create a smoother curve.
type	Type of density curve plotted. By default, the general density is plotted, though can request the normal density and both densities.
histogram	If TRUE overlay the density plot over a histogram.
bin_start	Optional specified starting value of the bins.
bin_width	Optional specified bin width, which can be specified with or without a bin_start value.
color_nrm	Color of the normal curve.
color_gen	Color of the general density curve.
fill_nrm	Fill color for the estimated normal curve, with a partially transparent blue as the default, and transparent for the gray theme.
fill_gen	Fill color for the estimated general density curve, with a partially transparent light red as the default, and a light transparent gray for the gray theme.
rotate_x	Degrees that the x-axis values are rotated, usually to accommodate longer values, typically used in conjunction with offset.
rotate_y	Degrees that the y-axis values are rotated.
offset	The amount of spacing between the axis values and the axis_ Default is 0.5. Larger values such as 1.0 are used to create space for the label when longer axis value names are rotated.
x.pt	Value of the point on the x-axis for which to draw a unit interval around illustrating the corresponding area under the general density curve. Only applies when requesting type=general.
xlab	Label for x-axis_ Defaults to variable name unless variable labels are present, the defaults to also include the corresponding variable label. Can style with the lessR style function.
main	Label for the title of the graph. Can set size with main_cex and color with main_color from the lessR style function.

`sub`	Sub-title of graph, below xlab_
`y_axis`	Specifies if the y-axis, the density axis, should be included.
`x.min`	Smallest value of the variable x plotted on the x-axis_
`x.max`	Largest value of the variable x plotted on the x-axis_
`rug`	If `TRUE`, add a rug plot, a direct display of density in the form of a narrow band beneath the density curve.
`color_rug`	Color of the rug ticks.
`size_rug`	Line width of the rug ticks.
`eval_df`	Determines if to check for existing data frame and specified variables. By default is `TRUE` unless the `shiny` package is loaded then set to `FALSE` so that Shiny will run. Needs to be set to `FALSE` if using the pipe `%\>%` notation.
`digits_d`	Number of significant digits for each of the displayed summary statistics.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`width`	Width of the plot window in inches, defaults to 4.5.
`height`	Height of the plot window in inches, defaults to 4.5.
`pdf_file`	Indicate to direct pdf graphics to the specified name of the pdf file.
`fun_call`	Function call. Used with `knitr` to pass the function call when obtained from the abbreviated function call `dn`.
`...`	Other parameter values for graphics as defined processed by `plot`, including `xlim`, `ylim`, `lwd` and `lab_cex`, `color_main`, `color_lab`, `sub`, `color_sub`, and `color_ticks` to specify the color of the ticks used to label the axis values, `density`, for the general density calculations, can set bandwidth with the standard `bw`.

Details

OVERVIEW
Results are based on the standard `dnorm` function and `density` R functions for estimating densities from data, as well as the `hist` function for calculating a histogram. Colors are provided by default and can also be specified.

The default histogram can be modified with the `bin_start` and `bin_width` options. Use the `Histogram` function in this package for more control over the parameters of the histogram.

The limits for the axes are automatically calculated so as to provide sufficient space for the density curves and histogram, and should generally not require user intervention. Also, the curves are centered over the plot window so that the resulting density curves are symmetric even if the underlying histogram is not. The estimated normal curve is based on the corresponding sample mean and standard deviation.

If `x.pt` is specified, then `type` is set to general and `y_axis` set to `TRUE`.

DATA
The data may either be a vector from the global environment, the user's workspace, as illustrated in the examples below, or one or more variable's in a data frame, or a complete data frame. The

default input data frame is d. Can specify the source data frame name with the data option. If multiple variables are specified, only the numerical variables in the list of variables are analyzed. The variables in the data frame are referenced directly by their names, that is, no need to invoke the standard R mechanisms of the d$name notation, the with function or the attach function. If the name of the vector in the global environment and of a variable in the input data frame are the same, the vector is analyzed.

The rows parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than.

COLOR THEME
Individual colors in the plot can be manipulated with options such as color_bars for the color of the histogram bars. A color theme for all the colors can be chosen for a specific plot with the colors option with the lessR function style. The default color theme is blue, but a gray scale is available with "gray", and other themes are available as explained in style, such as "red" and "green". Use the option style(sub_theme="black") for a black background and partial transparency of plotted colors.

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see Read.

PDF OUTPUT
To obtain pdf output, use the pdf option, perhaps with the optional width and height options. These files are written to the default working directory, which can be explicitly specified with the R setwd function.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a lessR function can only be a variable name (or list of variable names). This referenced variable must exist in either the referenced data frame, such as the default d, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

```
> Density(rnorm(50)) # does NOT work
```

Instead, do the following:

```
> Y <- rnorm(50)   # create vector Y in user workspace
> Density(Y)     # directly reference Y
```

Value

The output can optionally be saved into an R object, otherwise it simply appears in the console. Redesigned in lessR version 3.3 to provide two different types of components: the pieces of readable output, and a variety of statistics. The readable output are character strings such as tables amenable for reading. The statistics are numerical values amenable for further analysis. The motivation of these types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object and a $, can be inserted into the R~Markdown document (see examples).

READABLE OUTPUT
out_title: Title of output

out_stats: Statistics
out_file: Name and location of optional R markdown file

STATISTICS
bw: Bandwidth parameter
n: Number of data values analyzed
n.miss: Number of missing data values
W: W statistic for Shapiro-Wilk normality test
pvalue: p-value for W statistic

Although not typically needed, if the output is assigned to an object named, for example, h, then the contents of the object can be viewed directly with the unclass function, here as unclass(h).

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

dnorm, density, hist, plot, rgb, shapiro.test.

Examples

```
# make sure default style is active
style()

# create data frame, d, to mimic reading data with Read function
# d contains both numeric and non-numeric data
d <- data.frame(rnorm(50), rnorm(50), rnorm(50), rep(c("A","B"),25))
names(d) <- c("X","Y","Z","C")

#  general density curves superimposed over histogram, all defaults
Histogram(Y, density=TRUE)

# see Histogram for more examples, also the corresponding vignette
```

---

details *Display Contents of a Data File and Optional Variable Labels*

Description

Abbreviation: db

Provides feedback regarding a data frame which includes the variable names, the dimensions of the resulting data frame, the data type for each variable, and the values of the variables in the data file for the first and last rows of the data. In addition, an analysis of missing data is provided, listing the number of missing values for each variable and for each observation.

Usage

```
details(data=d, n_mcut=1, max_lines=30,
         miss_show=30, miss_matrix=FALSE, var_labels=FALSE,
         brief=getOption("brief"))

db(..., brief=TRUE)
```

Arguments

`data`	Data frame for which to provide the details.
`n_mcut`	For the missing value analysis, list the row name and number of missing values if the number of missing exceeds or equals this cutoff.
`max_lines`	Maximum number of lines to list of the data and labels.
`miss_show`	For the missing value analysis, the number of rows, one row per observation, that has as many or missing values as `n_mcut`.
`miss_matrix`	For the missing value analysis, if there is any missing data, list a version of the complete data table with a 0 for a non-missing] value and a 1 for a missing value.
`var_labels`	The data frame consists of variable labels if `TRUE`, so the message about a column of unique values is not displayed.
`brief`	If `TRUE`, display only variable names table plus any variable labels. The default for "details brief" abbreviation `db`.
`...`	Further arguments to be passed to or from methods consistent with the R `read.table` function. For example, can set `stringsAsFactors` as `TRUE`.

Details

MISSING DATA
When `brief` is set to `FALSE`, `details` provides a list the row of data with missing values, indicated by the standard R missing value code NA. To view the entire data table in terms of 0's and 1's for non-missing and missing data, respectively, invoke the `miss_matrix=TRUE` option.

VARIABLE LABELS
Standard R does not provide for variable labels, but `lessR` does. Variable labels can be provided for some or all of the variables in the data frames. One way to enter the variable labels is to read them from their own file with `details` with `labels` set to the full path name or `URL` of the labels file, or just the file name if the labels file is in the same directory as the data file. Another method is to include the labels directly in the data file. To to this, specify the file of variable labels with the `label="row2"` option. The web survey application Qualtrics downloads `csv` files in this format.

For a file that contains only labels, each row of the file, including the first row, consists of the variable name, a comma, and then the label, that is, standard `csv` format such as obtained with the `csv` option from a standard worksheet application such as Microsoft Excel or LibreOffice Calc. Not all variables in the data frame that contains the data, usually `d`, need have a label, and the variables with their corresponding labels can be listed in any order. An example follows.

I2,This instructor presents material in a clear and organized manner.
I4,Overall, this instructor was highly effective in this class.
I1,This instructor has command of the subject.

I3,This instructor relates course materials to real world situations.

If there is a comma in the variable label, then the label needs to be enclosed in quotes.

The lessR functions that provide analysis, such as Histogram for a histogram, automatically include the variable labels in their output, such as the title of a graph. Standard R functions can also use these variable labels by invoking the label function, such as setting main=label(I4) to put the variable label for a variable named I4 in the title of a graph.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Read.

Examples

```
# read the built-in data set datEmployee
# this provides an automatic call to details
d <- Read("Employee")

# manually request the details for d
details()

# manually request just variable names, labels for d
db()
```

---

factors *Create Factor Variables Across a Sequential Range or Vector of Variables*

Description

Creates factors for many variables. Specify a range from a given start variable and end variable. Applies only to variables in a data frame, d by default, and outputs the entire data frame including the factor transformation.

Usage

```
factors(x, levels, labels=NULL, data=d, ordered=FALSE,
          new=FALSE, suffix="_f", var_labels=FALSE, ...)
```

Arguments

x	Name of variable(s) to convert to a factor. List a single variable or a vector
levels	Levels for which to define the factor.
labels	Value labels to assign to the levels. If not present then assumes the character version of the levels.
data	The data frame of interest.
ordered	If `FALSE`, factor levels are not ordered.
new	If `FALSE`, original variables are replaced, otherwise new factor variables are created.
suffix	The appended suffix to newly created variables from the original variable names when `new` is `TRUE`.
var_labels	Just create new variable labels for newly created factor variables, without doing a factor conversion, presumably after a previous run with factors converted to new factor variables.
...	Other parameter values_

Details

Returns the entire data frame if applied to one or more variables in a data frame, including the new factors.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# get the data, variables Gender plus m01 through m20, 20 Mach IV items
# coded as integers from 0 to 5 on 6-pt Likert scales
d <- rd("Mach4", quiet=TRUE)

# single variable converted to a factor
d <- factors(Gender, 0:1, c("Male", "Female"))

# Define the labels
LikertCats <- c("Strongly Disagree", "Disagree", "Slightly Disagree",
                "Slightly Agree", "Agree", "Strongly Agree")

# Convert the integer responses to factors for the 20 Mach IV items
d <- factors(m01:m20, levels=0:5, labels=LikertCats)

# read the data again and this time also the variable labels
d <- rd("Mach4", quiet=TRUE)
l <- rd("Mach4_lbl")

# convert specified variables to factors according to the given vector
#   of three variables only
# leave the original variables unmodified, create new variables
```

```
d <- factors(c(m06, m07, m20), levels=0:5, labels=LikertCats, new=TRUE)
# now copy the variable labels from the original integer variables to the
#   newly created factor variables
l <- factors(c(m06, m07, m20), var_labels=TRUE)
```

---

getColors *Hue, Chroma, Luminance (HCL) Color Wheel or Specified Colors*

Description

Generates color vectors, including HCL colors for qualitative and sequential color scales, and displays these internally generated as well as manually specified arbitrary colors. To avoid bias in comparing differently colored regions of a visualization, generates HCL colors by default with fixed values of chroma (saturation) and luminance (brightness) for a range of hues, by default ordered so that adjacent colors are as separated as possible. Also generates a sequence of HCL colors according to any chosen hue value in which implicit calls can vary chroma and luminance to Zeileis's et al. `sequential_hcl` function from Ihaka's et al. `colorspace` package, and also with pre-defined values such as `"blues"`. The function also processes any arbitrarily specified set of colors or colors generated from a custom range according to a beginning and ending specified color. The function also includes color palettes from the `viridis` and `wesanderson` packages.

In terms of workflow, use the function to select a set of colors from the resulting color rectangle/wheel. The function outputs the colors so that the function call can serve as an argument to parameters in other functions that require a sequence of one or more colors as input. The visualization of the color wheel or rectangle is not generated in this situation. After selecting the colors, pass to an argument for a visualization function such as for the `fill` parameter.

Usage

```
getColors(pal=NULL, end_pal=NULL,
       n=12, h=0, h2=NULL, c=NULL, l=NULL, transparency=0,
       in_order=NULL, fixup=TRUE, power=NULL,
       shape=c("rectangle", "wheel"), radius=0.9, border="lightgray",
       main=NULL, labels=NULL, labels_cex=0.8, lty="solid",
       output=NULL, quiet=getOption("quiet"), ...)
```

Arguments

`pal`	Palette of specified colors to plot. If specified colors, then the following parameters are not relevant. Can also be pre-defined color sequences that trigger a sequence of colors from light to dark, such as `"blues"`, or `"distinct"` to maximize color separation.
`end_pal`	If specified, then generate a color continuum that begins at `pal` and ends at `end_pal`.
`n`	Number of colors to display.
`h`	Beginning HCL hue, 0 to 360.

h2	Ending HCL hue, 0 to 360. Defaults to a value close to 360. Requires `in_order` to be `FALSE`.
c	Value of HCL chroma (saturation). Respective default values for qualitative, sequential, and divergent scales are 65, c(35,75), and 50.
l	Value of HCL luminance (brightness). Respective default values for qualitative, sequential, and divergent scales are 60, c(80,25), and c(40,70).
transparency	Transparency factor of the area of each slice from 0, no transparency to 1, full transparency.
in_order	If `TRUE`, orders the colors in order of their HCL hue values, the default for a "wheel". Otherwise maximizes the difference between adjacent colors hues to prepare for inclusion in visualizations with qualitative, discrete color scales.
fixup	R parameter name. If `TRUE`, then HCL values outside of the displayable RGB color space are transformed to fit into that space so as to display.
power	Power for generating a sequential or divergent HCL scale (via colorspace package) for potentially non-linear changes in chroma and luminance across the scale. Default for sequential is 1 and for divergent 0.75.
shape	Default is a `"rectangle"`, or specify a `"wheel"`.
radius	Size of wheel. Not applicable to the rectangular shape.
border	Color of the borders of each slice. Set to `"off"` or `"transparent"` to turn off.
main	Title. Unlike other `lessR` functions, there is a default title, turned off by explicitly setting to `NULL` or `""`.
labels	If `TRUE`, then displayed. For HCL qualitative scale, default is `TRUE`, otherwise `FALSE`
.	
labels_cex	Character expansion factor of labels relative to 1.
lty	Line type of the border.
output	Default to produce text and graphics output when called directly from the console but not when called from a visualization function or a direct call in R Markdown, which requires `output=TRUE`.
quiet	If set to `TRUE`, no text output. Can change system default with `style` function.
...	Other parameter values.

Details

I. HCL COLORS
Generate a palette of colors according to the parameter `pal` in the form of a character string vector of their names, and also as a color wheel if not called from another function. The default value (for all but grayscale or white color themes) of `pal` is `"hues"`, which generates a qualitative palette of the specified number, `n`, of discrete HCL colors at the same chroma and luminance, respective default values of 65 and 60. With constant chroma and luminance the HCL color space provides a

palette of colors with the same gray-scale intensities if desaturated. That means no brightness bias for viewing different colors that represent different areas, such as in a bar chart of two variables, or a pie chart. The primary constraint is that the HCL color space is not in a one-to-one correspondence with the RGB color space of computer monitors, so some HCL colors are approximated (with the default setting of the `fixup` parameter set to `TRUE`).

For `"hues"`, the default, the hue values and associated colors are expressed as HEX and RGB values. The first 12 generated discrete colors are blue (240), brown (60), green (120), red (0), purple (275), turquoise (180), rust (30), olive (90), aqua (210), mulberry (330), emerald (150), and violet (300).

To have the generated colors be in the sequential order of hues, set `in_order` to `TRUE`, the default when `shape` is set to `"wheel"`. For about up to five or six colors adjacent values are still reasonably well distinguished even if in sequential order of hue number in the hcl space.

II. COLOR SEQUENCE
A second possibility generates a sequence of colors according to the value of `n` from a given start color to an ending color. To specify a custom range, set `pal` as the value of the first color, and then `end_pal` as the value of the last color in the color range. The colors in the sequence may or may not be of the same hue.

Or, access implicit calls Zeileis (2009) `sequential_hcl` and `diverge_hcl` functions from the `colorspace` package to access pre-defined color ranges including `"grays"`, which is the default if the color theme is `"gray"` or `"white"`. Other predefined sequences are shown in the following table. Also can invoke the standard R color ranges of `"heat"`, `"terrain"`, and `"rainbow"`, or, preferably, their `colorspace` equivalents: `"rainbow_hcl"`, `"heat_hcl"`, and `"terrain_hcl"`.

Can specify any value of hue with `h`. Can also provide custom values of chroma (c) and luminance (l), with either one a range of values defined as a vector of two values_ Default values are `c=100` and `l=c(75,35)`. That is, the color sequence is generated according to the given hue, `h`, with a chroma of 100 and luminance varying from 75 to the darker 45.

The predefined sequences consist of the following hues and color names, defined in 30 degree increments around the HCL color wheel. Visualize the color wheel with then discrete colors below with the lessR function `getColors`, specifically the function call `getColors(shape="wheel")`. Visualize sequential color scales for each of the colors below with the lessR function `showPalettes`.

colors	param	value
`"reds"`	h	0
`"rusts"`	h	30
`"browns"`	h	60
`"olives"`	h	90
`"greens"`	h	120
`"emeralds"`	h	150
`"turquoises"`	h	180
`"aquas"`	h	210
`"blues"`	h	240
`"purples"`	h	270
`"violets"`	h	300
`"magentas"`	h	330
`"grays"`	c	0

The predefined color name can be provided as the first argument of the function call, that is, the value of `pal`, or the corresponding value of `h` (or `c` for gray scale) can be specified. The specifications are equivalent. To specify a divergent color scale, provide both the value of `pal` as the beginning value and the value of `end_pal` as the last value, such that both values are one of the pre-specified color ranges. In either situation, of sequential or divergent color scales, custom values of `c` and `l` can be provided.

III. SPECIFIED COLORS
The third possibility is to generate a color wheel from a specified set of color values. Set the value of `pal` according to the vector of these values. Specify the values with R color names (see the lessR function `showColors`), RGB values according to the `rgb` function or from related R color space functions such as `hcl`, or as hexadecimal codes.

IV. OTHER INCLUDED COLOR PALETTES
The following palettes are based on those from the viridis package: `"viridis"`, `"cividis"`, `"plasma"`, and `"spectral"`, though the palettes here are generated from the base R function `hcl.colors`. These palettes were developed to be more useable for varying types of color-blindness, as is the included palette `"Okabe-Ito"`. The Tableau default qualitative color palette is also included, identified by `"Tableau"`.

Movie director Wes Anderson is known for is innovative color themes in his movies, which feature a combination of pastel colors and bold primary colors. The following palettes are from the wesanderson package, based the colors from his movies: "BottleRocket1", "BottleRocket2", "Rushmore1", "Rushmore", "Royal1", "Royal2", "Zissou1", "Darjeeling1", "Darjeeling2 ", "Chevalier1", "FantasticFox1", "Moonrise1", "Moonrise2", "Moonrise3", "Cavalcanti1", "GrandBudapest1", "GrandBudapest2", "IsleofDogs1", "IsleofDogs2". The generation of the corresponding palettes are with `type` set to `"continuous"` to generalize to palettes of any length. Note that this package is suggested, which means to use the package for the first time you will be prompted to install the package.

The palette `"distinct"` specifies a sequence of 20 colors manually chosen for the distinctiveness. The first five colors are from the qualitative sequence of `hcl` colors with `c=90` and `l=50`. To maximise color separation, the remaining 15 colors do not satisfy constance levels of `c` and `l`. Use such as for plotting with a `by` variable with up to 20 levels.

FUNCTION USAGE
Use the function on its own, in which case the color rectangle/wheel visualization is generated as are the color values. The vector of color values may be saved in its own R object as the output of the function call. Or, use the function to set colors for other parameter values in other function calls. See the examples.

Value

Colors are invisibly returned as a character string vector.

References

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 10, NY: CRC Press.

See Also

hcl, palette.colors, hcl.colors, showColors

Examples

```
# HCL color wheels/rectangles
#----------------------------
# set in_order to TRUE for hues ordered by their number

# color spectrum of 12 hcl colors presented in the order
#  in which they are assigned to discrete levels of a
#  categorical variable
getColors()

# color spectrum of 12 hcl colors ordered by hue from 0
#   by intervals of 360/12 = 30 degrees
getColors(in_order=TRUE)

# pastel hcl colors, set luminance to 85 from default of 50
getColors(in_order=TRUE, l=85)

# color wheel of 36 ordered hues around the wheel
getColors(n=36, shape="wheel", border="off")

# ggplot qualitative colors, here for 3 colors generated
#   in order of their hue numbers across the color wheel
#   starting at a hue of 15 degrees and luminance of 60
getColors(h=15, n=3, l=60, in_order=TRUE)


# HCL Qualitative Scale
# ---------------------
# default pre-defined 12 hcl colors that were manually reordered
#   so that adjacent colors achieve maximum separation
getColors()

# deep rich colors for HCL qualitative scale
getColors(c=90, l=45)


# HCL Sequential Scales
# ---------------------
# generate hcl blue sequence with c=60 and vary l
getColors("blues", labels=FALSE)

# generate yellow hcl sequence with varying chroma
getColors("browns", c=c(20,90), l=60)

# non-linear grayscale, more concentration of colors at the beginning
getColors("black", "white", n=24, power=0.75)

# generate custom hue color sequence close to colorbrewer Blues
```

```
# library(RColorBrewer)
# getColors(brewer.pal(6,"Blues"))
# compare, vary both l and c
getColors(h=230, n=6, l=c(96,30), c=c(5,80))

# a standard R color sequence
getColors("heat")

# from viridis
getColors("viridis", n=12)

# maximally distinct
getColors("distinct", n=20)


# HCL Divergent Scales
# --------------------
# seven colors from rust to blue
getColors("rusts", "blues", n=7)

# add a custom value of chroma, c, to make less saturated
getColors("rusts", "blues", n=7, c=45)


# Manual Specification of Colors
# ------------------------------
# individually specified colors
getColors(c("black", "blue", "red"))

# custom sequential range of colors
getColors(pal="aliceblue", end_pal="blue")


# Plots
# -----
d <- rd("Employee")

# default quantitative scale
bc(Dept, fill=getColors())
# or with implicit call to getColors
bc(Dept, fill="colors")
# or an implicit call with the blues
bc(Dept, fill="blues")
# or explicit call
bc(Dept, fill=getColors("blues"))

# custom hue with different chroma levels (saturations)
BarChart(Dept, fill=getColors(h=230, c=c(20,60), l=60))

# custom hue with different luminance levels (brightness)
# if explicitly calling getColors need to also specify n
Histogram(Salary, fill=getColors(h=230, c=60, l=c(90,30), n=10))
```

```
# use the default qualitative viridis color scale
bc(Dept, fill="viridis")
```

Histogram *Histogram*

Description

Abbreviation: hs

From the standard R function hist, the function plots a frequency histogram with default colors, including background color and grid lines plus an option for a relative frequency and/or cumulative histogram, as well as summary statistics and a table that provides the bins, midpoints, counts, proportions, cumulative counts and cumulative proportions. Bins can be selected several different ways besides the default, including specifying just the bin width and/or the bin start. Also provides improved error diagnostics and feedback for the user on how to correct the problem when the bins do not contain all of the specified data.

If a set of multiple variables is provided, including an entire data frame, then each numeric variable in that set of variables is analyzed, with the option to write the resulting histograms to separate pdf files. The related CountAll function does the same for all variables in the set of variables, histograms for continuous variables and bar charts for categorical variables. Specifying a by1 or by2 variable implements Trellis graphics.

When output is assigned into an object, such as h in h <- hs(Y), can assess the pieces of output for later analysis. A primary such analysis is knitr for dynamic report generation from a generated R markdown file according to the Rmd option in which interpretative R output is embedded in documents. See value below.

Usage

```
Histogram(

    # ------------------------------------------
    # Data from which to construct the histogram
    x=NULL, data=d, filter=NULL,
    stat_x=c("count", "proportion"),

    # --------------------
    # Trellis (facet) plot
    by1=NULL, by2=NULL,
    n_row=NULL, n_col=NULL, aspect="fill",

    # ----------------------------------------------------------------
    # Analogy of physical Marks on paper that create the bars and labels
    theme=getOption("theme"),
    fill=getOption("bar_fill_cont"),
    color=getOption("bar_color_cont"),
    transparency=getOption("trans_bar_fill"),
```

```
values=FALSE,

# --------------------------------------
# Form of the histogram
# --------------------------------------
# Binning the continuous variable x
bin_start=NULL, bin_width=NULL, bin_end=NULL, breaks="Sturges",

# Cumulative histogram
cumulate=c("off", "on", "both"), reg="snow2",

# Density (smooth curve) plot
density=FALSE, show_histogram=TRUE,
bandwidth=NULL, type=c("general", "normal", "both"),
fill_general=NULL, fill_normal=NULL, fill_hist=getOption("se_fill"),
color_general="gray20", color_normal="gray20",
x.pt=NULL, y_axis=FALSE,
rug=FALSE, color_rug="black", size_rug=0.5,

# ----------------------------------------------------------------
# Labels for axes, values, and legend if x and by variables, margins
xlab=NULL, ylab=NULL, main=NULL, sub=NULL,
lab_adjust=c(0,0), margin_adjust=c(0,0,0,0),

rotate_x=getOption("rotate_x"), rotate_y=getOption("rotate_y"),
offset=getOption("offset"),
scale_x=NULL, scale_y=NULL,

# ---------------------------------------------------------------------
# Draw one or more objects, text, or geometric figures, on the histogram
add=NULL, x1=NULL, y1=NULL, x2=NULL, y2=NULL,

# --------------------------------------
# Output: turn off, chart to PDF file, decimal digits, markdown file
quiet=getOption("quiet"), do_plot=TRUE,
pdf_file=NULL, width=6.5, height=6,
digits_d=NULL,
Rmd=NULL,

# -------------------------------------
# Deprecated, removed in future versions
n_cat=getOption("n_cat"), rows=NULL,

# ---------------------------------
# Miscellaneous
eval_df=NULL, fun_call=NULL, ...)
```

```
hs(...)
```

Arguments

`x`	Variable(s) to analyze. Can be a single numerical variable, either within a data frame or as a vector in the users workspace, or multiple variables in a data frame such as designated with the `c` function, or an entire data frame. If not specified, then defaults to all numerical variables in the specified data frame, `d` by default.
`data`	Optional data frame that contains the variable(s) of interest, default is `d`.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`stat_x`	Bin and transform values of variable `x` into `"counts"` by default or `"proportion"` if specified, that is, frequencies or relative frequencies.
`by1`	A categorical variable called a conditioning variable that activates **Trellis graphics**, from the lattice package, to provide a separate scatterplot (panel) of numeric primary variables `x` and `y` for each level of the variable.
`by2`	A second conditioning variable to generate Trellis plots jointly conditioned on both the `by1` and `by2` variables, with `by2` as the row variable, which yields a scatterplot (panel) for each cross-classification of the levels of numeric `x` and `y` variables.
`n_row`	Optional specification for the number of rows in the layout of a multi-panel display with Trellis graphics. Need not specify `ncols`.
`n_col`	Optional specification for the number of columns in the layout a multi-panel display with Trellis graphics. Need not specify `n_row` If set to 1, then the strip that labels each group is moved to the left of each plot instead of the top.
`aspect`	Lattice parameter for the aspect ratio of the panels, defined as height divided by width. The default value is `"fill"` to have the panels expand to occupy as much space as possible. Set to 1 for square panels. Set to `"xy"` to specify a ratio calculated to "bank" to 45 degrees, that is, with the line slope approximately 45 degrees.
`theme`	**Color** theme for this analysis. Make persistent across analyses with `style`.
`fill`	Fill color of the bars. Can explicitly choose `"grays"` or `"hcl"` colors, or pre-specified R color schemes `"rainbow"`, `"terrain"`, and `"heat"`. Can also provide pre-defined color ranges `"blues"`, `"reds"` and `"greens"`, as well as custom colors, such as generated by `getColors`. Default is `bar_color` from the lessR `style` function.
`color`	Border color of the bars, can be a vector to customize the color for each bar. Default is `bar_color` from the lessR `style` function.
`transparency`	Transparency factor of the area of each slice. Default is `trans_bar_fill` from the lessR `style` function.
`values`	Replaces standard R `labels` options, which has multiple definitions in R. Specifies to display the count of each bin.

`bin_start`	Optional specified starting value of the bins.
`bin_width`	Optional specified bin width, which can be specified with or without a `bin_start` value.
`bin_end`	Optional specified value that is within the last bin, so the actual endpoint of the last bin may be larger than the specified value.
`breaks`	The method for calculating the bins, or an explicit specification of the bins, such as with the standard R `seq` function or other options provided by the `hist` function that include the default `"Sturges"` plus `"Scott"` and `"FD"`. Not applicable and so not allowed if `density` is `TRUE`.
`cumulate`	Specify a **cumulative histogram**. The value of `"on"` displays the cumulative histogram, with default of `"off"`. The value of `"both"` superimposes the regular histogram.
`reg`	The color of the superimposed, regular histogram when `cumulate="both"`.
`density`	If `TRUE`, plot the smoothed **kernel density** estimate.
`show_histogram`	When `density` is `TRUE`, plot a histogram behind the density curve.
`bandwidth`	Bandwidth of kernel density estimation, which determines the smoothness of the resulting density curve, larger values yield more smooth curves. Initial value is "nrd0", but unless specified, may be automatically iterated upward to create a smoother curve.
`type`	Type of density curve plotted. By default, the general density is plotted, though can request the normal density and both densities.
`fill_general`	Fill color for the estimated general density curve, with a partially transparent light red as the default, and a light transparent gray for the gray theme. Supplied color names are automatically revised with moderate transparency, the same level as the default.
`fill_normal`	Fill color for the estimated normal curve, with a partially transparent blue as the default, and transparent for the gray theme.
`fill_hist`	Fill color for the histogram behind density curve, defaults to a light gray.
`color_general`	Color of the general density curve border.
`color_normal`	Color of the normal curve border.
`x.pt`	Value of the point on the x-axis for which to draw a unit interval around illustrating the corresponding area under the general density curve. Only applies when requesting `type=general`.
`y_axis`	Specifies if the y-axis, the density axis, should be included.
`rug`	If `TRUE`, add a rug plot, a direct display of density as a narrow band beneath the density curve.
`color_rug`	Color of the rug ticks.
`size_rug`	Line width of the rug ticks.

`xlab`	Label for x-axis_ Defaults to variable name unless variable labels are present, the defaults to also include the corresponding variable label. Can style with the lessR `style` function
`ylab`	Label for y-axis_ Defaults to Frequency or Proportion. Can style with the lessR `style` function.
`main`	Label for the title of the graph. Can set size with `main_cex` and color with `main_color` from the lessR `style` function.
`sub`	Sub-title of graph, below xlab. Not yet implemented.
`lab_adjust`	Two-element vector – x-axis label, y-axis label – adjusts the position of the axis labels in approximate inches. + values move the labels away from plot edge. Not applicable to Trellis graphics.
`margin_adjust`	Four-element vector – top, right, bottom and left – adjusts the margins of the plotted figure in approximate inches. + values move the corresponding margin away from plot edge. Not applicable to Trellis graphics.
`rotate_x`	Degrees that the `x`-**axis values** are rotated, usually to accommodate longer values, typically used in conjunction with `offset`. Can set persistently with the lessR `style` function.
`rotate_y`	Degrees that the y-axis values are rotated. Can set persistently with the lessR `style` function.
`offset`	The amount of spacing between the axis values and the axis_ Default is 0.5. Larger values such as 1.0 are used to create space for the label when longer axis value names are rotated. Can set persistently with the lessR `style` function.
`scale_x`	If specified, a vector of three values that define the numerical values of the x-axis: starting, ending and number of intervals, within the bounds of plot region.
`scale_y`	Applies to the y-axis_ See `scale_x`.
`add`	**Draw one or more objects**, text or a geometric figures, on the plot. Possible values are any text to be written, the first argument, which is `"text"`, or, to indicate a figure, `"rect"` (rectangle), `"line"`, `"arrow"`, `"v.line"` (vertical line), and `"h.line"` (horizontal line). The value `"means"` is short-hand for vertical and horizontal lines at the respective means. Does not apply to Trellis graphics. Customize with parameters such as `add_fill` and `add_color` from the `style` function.
`x1`	First x coordinate to be considered for each object. All coordinates vary from -1 to 1.
`y1`	First y coordinate to be considered for each object.
`x2`	Second x coordinate to be considered for each object. Only used for `"rect"`, `"line"` and `arrow`.
`y2`	Second y coordinate to be considered for each object. Only used for `"rect"`, `"line"` and `arrow`.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.

do_plot	If TRUE, the default, then generate the plot.
pdf_file	Indicate to direct pdf graphics to the specified name of the pdf file.
width	Width of the plot window in inches, defaults to 4.5.
height	Height of the plot window in inches, defaults to 4.5.
digits_d	Number of significant digits for each of the displayed summary statistics.
Rmd	File name for the file of R markdown to be written, if specified. The file type is .Rmd, which automatically opens in RStudio, but it is a simple text file that can be edited with any text editor, including RStudio.
n_cat	For the analysis of multiple variables, such as a data frame, specifies the largest number of unique values of variable of a numeric data type for which the variable will be analyzed as a categorical. Default is 0. *[deprecated]*: Best to convert a categorical integer variable to a factor.
rows	Deprecated old parameter name that is now called `filter`.
eval_df	Determines if to check for existing data frame and specified variables. By default is TRUE unless the `shiny` package is loaded then set to FALSE so that Shiny will run. Needs to be set to FALSE if using the pipe `%\>%` notation.
fun_call	Function call. Used with `knitr` to pass the function call when obtained from the abbreviated function call `hs`.
...	Other parameter values for graphics as defined processed by `hist` and `par` for general graphics, `xlim` and `ylim` for setting the range of the x and y-axes `cex.main` for the size of the title `col.main` for the color of the title `cex` for the size of the axis value labels `col.lab` for the color of the axis labels

Details

OVERVIEW
Results are based on the standard R `hist` function to calculate and plot a histogram, or a multi-panel display of histograms with Trellis graphics, plus the additional provided color capabilities, a relative frequency histogram, summary statistics and outlier analysis. The `freq` option from the standard R `hist` function has no effect as it is always set to FALSE in each internal call to `hist`. To plot densities, set the parameter `density` to TRUE.

VARIABLES and TRELLIS PLOTS
At a minimum there is one primary variable, `x`, which results in a single histogram. Trellis graphics, from Deepayan Sarkar's `lattice` package, may be implemented in which multiple panels are displayed according to the levels of one or two categorical variables, called conditioning variables. A variable specified with `by1` is a conditioning variable that results in a Trellis plot, the histogram of `x` produced at *each* level of the `by1` variable. Inclusion of a second conditioning variable, `by2`, results in a separate histogram for *each* combination of cross-classified values of both `by1` and `by2`.

DATA
The data may either be a vector from the global environment, the user's workspace, as illustrated

in the examples below, or one or more variable's in a data frame, or a complete data frame. The default input data frame is d. Can specify the source data frame name with the data option. If multiple variables are specified, only the numerical variables in the list of variables are analyzed. The variables in the data frame are referenced directly by their names, that is, no need to invoke the standard R mechanisms of the d$name notation, the with function or the attach function. If the name of the vector in the global environment and of a variable in the input data frame are the same, the vector is analyzed.

To obtain a histogram of each numerical variable in the d data frame, use Histogram(). Or, for a data frame with a different name, insert the name between the parentheses. To analyze a subset of the variables in a data frame, specify the list with either a : or the c function, such as m01:m03 or c(m01,m02,m03).

The rows parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than. See the Examples.

COLORS
Individual colors in the plot can be manipulated with options such as color_bars for the color of the histogram bars. A color theme for all the colors can be chosen for a specific plot with the colors option with the lessR function style. The default color theme is lightbronze, but a gray scale is available with "gray", and other themes are available as explained in style, such as "red" and "green". Use the option style(sub_theme="black") for a black background and partial transparency of plotted colors.

For the color options, such as fill, the value of "off" is the same as "transparent".

Set fill to a single color or a color range, of which there are many possibilities. For "hues" colors of the same chroma and luminance set fill to multiple colors all with the same saturation and brightness. Also available are the pre-specified R color schemes "rainbow", "terrain", and "heat". Can also provide pre-defined color ranges "blues", "reds" and "greens", or generate custom colors, such as from the lessR function getColors.

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see Read.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a lessR function can only be a variable name (or list of variable names). This referenced variable must exist in either the referenced data frame, such as the default d, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

```
> Histogram(rnorm(50)) # does NOT work
```

Instead, do the following:

```
> Y <- rnorm(50)   # create vector Y in user workspace
> Histogram(Y)     # directly reference Y
```

ERROR DETECTION
A somewhat relatively common error by beginning users of the base R hist function may encounter is to manually specify a sequence of bins with the seq function that does not fully span the range of specified data values_ The result is a rather cryptic error message and program termination. Here,

Histogram detects this problem before attempting to generate the histogram with hist, and then informs the user of the problem with a more detailed and explanatory error message. Moreover, the entire range of bins need not be specified to customize the bins. Instead, just a bin width need be specified, bin_width, and/or a value that begins the first bin, bin_start. If a starting value is specified without a bin width, the default Sturges method provides the bin width.

PDF OUTPUT
To obtain pdf output, use the pdf_file option, perhaps with the optional width and height options. These files are written to the default working directory, which can be explicitly specified with the R setwd function.

Value

The output can optionally be saved into an R object, otherwise it simply appears in the console. Two different types of components are provided: the pieces of readable output, and a variety of statistics. The readable output are character strings such as tables amenable for display. The statistics are numerical values amenable for further analysis. The motivation of these types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object and a $, can be inserted into the R~Markdown document (see examples), interspersed with explanation and interpretation.

READABLE OUTPUT
out_suggest: Suggestions for other similar analyses
out_summary: Summary statistics
out_freq: Frequency distribution
out_outliers: Outlier analysis

STATISTICS
bin_width: Bin width
n_bins: Number of bins
breaks: Breaks of the bins
mids: Bin midpoints
counts: Bin counts
prop: Bin proportion
cumulate: Bin cumulative counts
cprop: Bin cumulative proportion

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapter 5, NY: Routledge.

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 4, NY: CRC Press.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Sarkar, Deepayan (2008) Lattice: Multivariate Data Visualization with R, Springer. http://lmdvr.r-forge.r-project.org/

See Also

getColors, hist, plot, par, style.

Examples

```
# get the data
d <- rd("Employee")


# make sure default style is active
style()


# --------------------
# different histograms
# --------------------

# histogram with all defaults
Histogram(Salary)
# short form
#hs(Salary)

# output saved for later analysis into object h
h <- hs(Salary)
# view full text output
h
# view just the outlier analysis
h$out_outliers
# list the names of all the components
names(h)

# histogram with no borders for the bars
Histogram(Salary, color="off")

# save the histogram to a pdf file
#Histogram(Salary, pdf=TRUE)

# just males employed more than 5 years
Histogram(Salary, rows=(Gender=="M" & Years > 5))

# histogram with red bars, black background, and black border
style(panel_fill="black", fill="red", panel_color="black")
Histogram(Salary)
# or use a lessR pre-defined sequential color palette
#   with some transparency
Histogram(Salary, fill="rusts", color="brown", transparency=.1)

# histogram with purple color theme, translucent gold bars
style("purple", sub_theme="black")
```

```
Histogram(Salary)
# back to default color theme
style()

# histogram with specified bin width
# can also use bin_start
Histogram(Salary, bin_width=12000)

# histogram with rotated axis values, offset more from axis
# suppress text output
style(rotate_x=45, offset=1)
Histogram(Salary, quiet=TRUE)
style()

# histogram with specified bins and grid lines displayed over the histogram
Histogram(Salary, breaks=seq(0,150000,20000), xlab="My Variable")

# histogram with bins calculated with the Scott method and values displayed
Histogram(Salary, breaks="Scott", values=TRUE, quiet=TRUE)

# histogram with the number of suggested bins, with proportions
Histogram(Salary, breaks=15, stat_x="proportion")

# histogram with non-default values for x- and y-axes
d[2,4] <- 45000
Histogram(Salary, scale_x=c(30000,130000,5), scale_y=c(0,9.5,5))

# ----------------
# Trellis graphics
# ----------------
Histogram(Salary, by1=Dept)


# --------------------
# cumulative histograms
# --------------------

# cumulative histogram with superimposed regular histogram, all defaults
Histogram(Salary, cumulate="both")

# cumulative histogram plus regular histogram
Histogram(Salary, cumulate="both", reg="mistyrose")

# -------------
# density plots
# -------------

# default density plot
Histogram(Salary, density=TRUE)

# normal curve and general density curves superimposed over histogram
# all defaults
Histogram(Salary, density=TRUE, type="both")
```

```
# display only the general estimated density
#  so do not display the estimated normal curve
# specify the bandwidth for the general density curve,
#  use the standard bandwidth option for the density function
Histogram(Salary, density=TRUE, bandwidth=8000)

# display only the general estimated density and a corresponding
#  interval of unit width around x.pt
Histogram(Salary, density=TRUE, x.pt=40000)

# densities for all specified numeric variables in a list of variables
# e.g., use the combine or c function to specify a list of variables
Histogram(c(Years,Salary), density=TRUE)


# -------------------------------------------------
# histograms for data frames and multiple variables
# -------------------------------------------------

# create data frame, d, to mimic reading data with Read function
# d contains both numeric and non-numeric data
d <- data.frame(rnorm(50), rnorm(50), rnorm(50), rep(c("A","B"),25))
names(d) <- c("X","Y","Z","C")

# although data not attached, access the variable directly by its name
Histogram(X)

# histograms for all numeric variables in data frame called d
#  except for numeric variables with unique values < n_cat
# d is the default name, so does not need to be specified with data
Histogram()

# histogram with specified options, including red axis labels
style(fill="palegreen1", panel_fill="ivory", axis_color="red")
Histogram(values=TRUE)
style()  # reset

# histograms for all specified numeric variables
# use the combine or c function to specify a list of variables
Histogram(c(X,Y))


# -----------
# annotations
# -----------

d <- rd("Employee")

# Place a message in the top-right of the graph
# Use \n to indicate a new line
hs(Salary, add="Salaries\nin our Company", x1=100000, y1=7)
```

```
# Use style to change some parameter values
style(add_trans=.8, add_fill="gold", add_color="gold4",
      add_lwd=0.5, add_cex=1.1)
# Add a rectangle around the message centered at <100000,7>
hs(Salary, add=c("rect", "Salaries\nin our Company"),
      x1=c(82000, 100000), y1=c(7.7, 7), x2=118000, y2=6.2)
```

---

interact *Run Interactive Shiny Data Visualizations*

Description

Interactive data visualizations. Choose your data, choose your variables, and set the parameters as desired.

Usage

```
interact(app)
```

Arguments

app	Name of the shiny app to run, enclosed in quotes.

Details

Valid names are "BarChart", "PieChart", "Histogram", "ScatterPlot", "Trellis". If missing, then the valid names are listed. Valid abbreviations, respectively, are "bc", "pc", "hs", and "Plot".

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# Commented out as the analyses are interactive
#interact()
#interact("BarChart")
```

`kurtosis` *Kurtosis*

Description

Kurtosis computed from the from the unbiased estimates of variance and of the fourth moment about the mean.

Usage

```
kurtosis(x, na.rm=TRUE)
```

Arguments

`x`	Variable from which to compute kurtosis.
`na.rm`	A logical value indicating whether NA values should be stripped before the computation proceeds.

Details

Kurtosis as implemented by SAS, Type 2 formula as classified by Joanes and Gill (1998). This version of the formula relies upon the unbiased estimates of variance and of the fourth moment about the mean. A perfect normal distribution would have a kurtosis of 0.

Value

kurtosis.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Joanes, D.N. and Gill, C.A (1998). Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183-189.

Examples

```
x <- rnorm(100)
kurtosis(x)
```

label *Assign Variable Labels [Superseded by VariableLabels]*

Description

Deprecated, replaced by `VariableLabels`. Display a variable label for output, either text output at the console or graphics, such as a title on a graph. To return a variable label generally applies to standard R functions such that the functions can access `lessR` variable labels. Or, the variable name and label can be listed on the standard output. To assign a variable label, invoke the `value` option and assign the output to a specified data frame.

Usage

```
label(x, value=NULL, data=d)
```

Arguments

`x`	The variable for which to obtain the corresponding variable label.
`value`	If assigned, then the specified data frame is updated with this assigned label.
`data`	Data frame that contains the variable of interest. The output of the function is assigned to this data frame.

Details

Standard R does not provide for variable labels, but `lessR` does. Read the labels with the `lessR` `Read` function, as explained in the corresponding documentation. Individual variable labels can also be assigned with this function. Not all variables need have a label, and the variables with their corresponding labels can be listed or assigned in any order.

The function provides two different modes. The first mode is to return the variable name and label for an existing variable label. One such use is to provide the function as an argument to an existing R function call to access a `lessR` variable label. For example, use the function as the argument for `main` in graphics output, where `main` is the title of the graph. This mode is triggered by not invoking the `value` option.

The second mode is to assign a variable label to an existing variable. Invoke this mode by specifying a variable label with the `value` option. The function accesses the entire specified data frame, and then modifies the specified variable label. As such, assign the output of the function to the data frame of interest, such as the default `d`. One use of this function is to add a variable label to a data frame that contains a new variable created by a transformation of the existing variables.

Value

The specified value of `value` is returned.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Read.

Examples

```
# read the data and variable labels
#d <- rd("http://lessRstats.com/data/employee.xlsx")
#l <- vl("http://lessRstats.com/data/employee_lbl.xlsx")

# variable label as the title of a graph for non-lessR functions
# base R
#hist(d$Salary, xlab=label(Salary))
# ggplot2
#ggplot(d, aes(Salary)) + geom_histogram(binwidth=10000) + labs(x=label(Salary))

# assign a new label for the variable Years in d
#d <- label(Years, "Years Worked")
# verify
#label(Years)
# or view all variable labels in d
#db()

#d <- Read("Employee")
# specify a label of variable in a data frame other than d
#myd <- Subset(Gender=="M")
#myd <- label(Gender, "Only is Male", data=myd)
#db(myd)
```

`LineChart` *Line Chart such as a Run Chart or Time-Series Chart*

Description

Abbreviation: `lc`

Plots a line chart, the values of the variable ordered according to their order in the data frame. Usually this ordering would be an ordering according to time, which yields a run chart. The default run chart provides the index, that is, sequential position, of each value of the variable from 1 to the last value. Optionally dates can be provided so that a time-series plot is produced.

For data of one variable exhibiting little trend, the center line is provided for the generation of a run chart, plotting the values of a variable in order of occurrence over time_ When the center line, the median by default, is plotted, the analyses of the number and composition of the individual runs, number of consecutive values above or below the center line, is also displayed. Also, the defaults change for each of the types of plots. The intent is to rely on the default values for a relatively sophisticated plot, particularly when compared to the default values of the standard R `plot` function called with a single variable.

If the provided object to analyze is a set of multiple variables, including an entire data frame, then each non-numeric variable in the data frame is analyzed and the results written to a pdf file in the current working directory. The name of each output pdf file that contains a bar chart and its path are specified in the output.

Usage

```
LineChart(x, data=d, rows=NULL,
          n_cat=getOption("n_cat"), type=NULL,

         line_color=getOption("pt_color"), area=NULL,

         shape_pts=21, lab_cex=1.0, axis_cex=0.75,
         axis_text_color=getOption("axis_x_text_color"),

         rotate_x=0, rotate_y=0, offset=.5,

         xy_ticks=TRUE, line_width=1,
         xlab=NULL, ylab=NULL, main=NULL, sub=NULL, cex=NULL,

         time_start=NULL, time_by=NULL, time_reverse=FALSE,

         center_line=c("default", "mean", "median", "zero", "off"),

         show_runs=FALSE, eval_df=NULL, quiet=getOption("quiet"),
         width=6, height=6, pdf=FALSE,
         ...)

lc(...)
```

Arguments

`x`	Variable(s) to analyze. Can be a single numerical variable, either within a data frame or as a vector in the user's workspace, or multiple variables in a data frame such as designated with the `c` function, or an entire data frame. If not specified, then defaults to all numerical variables in the specified data frame, `d` by default.
`data`	Optional data frame that contains the variable(s) of interest, default is `d`.
`rows`	A logical expression that specifies a subset of rows of the data frame to analyze.
`n_cat`	For the analysis of multiple variables, such as a data frame, specifies the largest number of unique values of variable of a numeric data type for which the variable will be analyzed as a categorical. Default is 0.
`type`	Character string that indicates the type of plot, either `"p"` for points, `"l"` for line, or `"b"` for both. The default is `"b"` for both points and lines.
`line_color`	Color of the plotted line.
`area`	Color of area under the plotted line segments, which by default is not applied, equivalent to a color of `"transparent"`.
`shape_pts`	The standard plot character, with values defined in help(points). The default value is 21, a circle with both a border and filled area, specified here as `color` and `fill`. `fill` defaults to `color`.
`lab_cex`	Scale magnification factor for axis labels.

`axis_cex`	Scale magnification factor, which by defaults displays the axis values to be smaller than the axis labels.
`axis_text_color`	Color of the font used to label the axis values_
`rotate_x`	Degrees that the x-axis values are rotated, usually to accommodate longer values, typically used in conjunction with `offset`.
`rotate_y`	Degrees that the y-axis values are rotated.
`offset`	The amount of spacing between the axis values and the axis_ Default is 0.5. Larger values such as 1.0 are used to create space for the label when longer axis value names are rotated.
`xy_ticks`	Flag that indicates if tick marks and associated values on the axes are to be displayed.
`line_width`	Width of the line segments_
`xlab`	Label for x-axis_ For two variables specified, x and y, if `xlab` not specified, then the label becomes the name of the corresponding variable. If `xy_ticks` is `FALSE`, then no label is displayed. If no y variable is specified, then `xlab` is set to Index unless `xlab` has been specified.
`ylab`	Label for y-axis_ If not specified, then the label becomes the name of the corresponding variable. If `xy_ticks` is `FALSE`, then no label displayed.
`main`	Label for the title of the graph. If the corresponding variable labels exist, then the title is set by default from the corresponding variable labels.
`sub`	Sub-title of graph, below xlab_
`cex`	Magnification factor for any displayed points, with default of cex=1.0.
`time_start`	Optional starting date for first data value. Format must be "%Y-%m-%d" or "%Y/%m/%d". If using with `x.reverse`, the first date is after the data are reverse sorted. Not needed if data are a time series with `ts` function.
`time_by`	Accompanies the `time_start` specification, the interval to increment the date for each sequential data value. A character string, containing one of `"day"`, `"week"`, `"month"` or `"year"`. This string can optionally be preceded by a positive or negative integer and a space, or followed by "s", as specified in `seq.Date`. Not needed if data are a time series.
`time_reverse`	When `TRUE`, reverse the ordering of the dates, particularly when the data are listed such that first row of data is the newest. Accompanies the `time_start` specification.
`center_line`	Plots a dashed line through the middle of a run chart. The two possible values for the line are `"mean"` and `"median"`. Provides a centerline for the `"median"` by default when the values randomly vary about the mean. A value of `"zero"` specifies the center line should go through zero.
`show_runs`	If `TRUE`, display the individual runs in the run analysis.
`eval_df`	Determines if to check for existing data frame and specified variables. By default is `TRUE` unless the `shiny` package is loaded then set to `FALSE` so that Shiny will run. Needs to be set to `FALSE` if using the pipe `%\>%` notation.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.

width	Width of the plot window in inches, defaults to 4.5.
height	Height of the plot window in inches, defaults to 4.5.
pdf	If TRUE, the pdf file is to be redirected to a pdf file.
...	Other parameters such as from par, col.lab, sub, color_sub, color_ticks to set the color of the ticks used to label the axis values, and srt to rotate the axis value labels.

Details

OVERVIEW
The line chart is based on the standard R function plot when called with only a single variable.

The values on the horizontal axis of the line chart are automatically generated. The default is the index variable, the ordinal position of each data value, in which case this version of the line chart is a run chart. Or, dates on the horizontal axis can be specified from the specified starting date given by x.start and the accompanying increment as given by x.by, in which case the line chart is typically referred to as a time series chart.

If the data values randomly vary about the mean, the default is to plot the mean as the center line of the graph, otherwise the default is to ignore the center line. The default plot type for the line chart is type="b", for both points and the corresponding connected line segments_ The size of the points is automatically reduced according to the number of points of points plotted, and the cex option can override the computed default. If the area below the plotted values is specified to be filled in with color, then the default line type changes to type="l".

DATA
The data may either be a vector from the global environment, the user's workspace, as illustrated in the examples below, or one or more variable's in a data frame, or a complete data frame. The default input data frame is d. Can specify the source data frame name with the data option. If multiple variables are specified, only the numerical variables in the list of variables are analyzed. The variables in the data frame are referenced directly by their names, that is, no need to invoke the standard R mechanisms of the d$name notation, the with function or the attach function. If the name of the vector in the global environment and of a variable in the input data frame are the same, the vector is analyzed.

The rows parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than.

COLORS
Individual colors in the plot can be manipulated with options such as color. A color theme for all the colors can be chosen for a specific plot with the colors option with the lessR function style. The default color theme is dodgerblue, but a gray scale is available with "gray", and other themes are available as explained in style, such as "red" and "green". Use the option style(sub_theme="black") for a black background and partial transparency of plotted colors.

For the color options, such as grid_color, the value of "off" is the same as "transparent".

VARIABLE LABELS
Although standard R does not provide for variable labels, lessR does, obtained from the Read function. If the variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see Read.

PDF OUTPUT
To obtain pdf output, use the `pdf` option, perhaps with the optional `width` and `height` options. These files are written to the default working directory, which can be explicitly specified with the R `setwd` function.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a `lessR` function can only be a variable name (or list of variable names). This referenced variable must exist in either the referenced data frame, such as the default `d`, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

`> LineChart(rnorm(50)) # does NOT work`

Instead, do the following:

```
> Y <- rnorm(50)   # create vector Y in user workspace
> LineChart(Y)     # directly reference Y
```

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

`plot`, `style`.

Examples

```
# create data frame, d, to mimic reading data with Read function
# d contains both numeric and non-numeric data
d <- data.frame(rnorm(50), rnorm(50), rnorm(50), rep(c("A","B"),25))
names(d) <- c("X","Y","Z","C")

# default run chart
LineChart(Y)
# short name
lc(Y)

# save run chart to a pdf file
#LineChart(Y, pdf=TRUE)

# LineChart in gray scale, then back to default theme
style("gray")
LineChart(Y)
style()

# customize run chart with LineChart options
style(panel_fill="mintcream", color="sienna3")
LineChart(Y, line_width=2, area="slategray3", center_line="median")
style()  # reset style

# customize run chart with R par parameters
# 24 is the R value for a half-triangle pointing up
```

```
lc(Y, xlab="My xaxis", ylab="My yaxis", main="My Best Title",
   cex.main=1.5, font.main=3, ylim=c(-4,4), shape_pts=24)

# generate steadily increasing values
# get a variable named A in the user workspace
A <- sort(rexp(50))
# default line chart
LineChart(A)
# line chart with border around plotted values
LineChart(A, area="off")
# time series chart, i.e., with dates, and filled area
# with option label for the x-axis
LineChart(A, time_start="2000/09/01", time_by="3 months")
# time series chart from a time series object
y.ts <- ts(A, start=c(2000, 9), frequency=4)
LineChart(y.ts)

# LineChart with built-in data set
LineChart(breaks, data=warpbreaks)

# Line charts for all numeric variables in a data frame
LineChart()

# Line charts for all specified numeric variables in a list of variables
# e.g., use the combine or c function to specify a list of variables
LineChart(c(X,Y))
```

Logit *Logit Regression Analysis*

Description

Abbreviation: `lr`

A wrapper for the standard R `glm` function with `family="binomial"`, automatically provides a logit regression analysis with graphics from a single, simple function call with many default settings, each of which can be re-specified. By default the data exists as a data frame with the default name of `d`, such as data read by the `lessR` `Read` function. Specify the model in the function call according to an R `formula`, that is, the response variable followed by a tilde, followed by the list of predictor variables, each pair separated by a plus sign.

The response variable for analysis has values only of 0 and 1, with 1 designating the reference group. If the response variable is a factor with two levels, they factor levels are automatically converted to a numeric variable with values of 0 and 1.

Default output includes the inferential analysis of the estimated coefficients and model, sorted residuals and Cook's Distance, and sorted fitted values for existing data or new data. For a single predictor variable model, the scatterplot of the data with plotted logit function is provided.

Can also be called from the more general `model` function.

Usage

```
Logit(my_formula, data=d, filter=NULL, ref_group=NULL,
      digits_d=4, text_width=120,

     brief=getOption("brief"),

     res_rows=NULL, res_sort=c("cooks","rstudent","dffits","off"),
     pred=TRUE, pred_all=FALSE, prob_cut=0.5, cooks_cut=1,

     X1_new=NULL, X2_new=NULL, X3_new=NULL, X4_new=NULL,
     X5_new=NULL, X6_new=NULL,

     pdf_file=NULL, width=5, height=5, ...)

lr(...)
```

Arguments

`my_formula`	Standard R `formula` for specifying a model. For example, for a response variable named Y and two predictor variables, X1 and X2, specify the corresponding linear model as Y ~ X1 + X2.
`data`	The default name of the data frame that contains the data for analysis is `d`, otherwise explicitly specify.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`ref_group`	Value of the response variable that is the reference group, otherwise set by default as the value that yields a + slope for one predictor variable or the largest alphabetical/numerical value if more than one predictor.
`digits_d`	For the Basic Analysis, it provides the number of decimal digits. For the rest of the output, it is a suggestion only.
`text_width`	Width of the text output at the console.
`brief`	If set to `TRUE`, reduced text output. Can change system default with `style` function.
`res_rows`	Default is 25, which lists the first 25 rows of data sorted by the specified sort criterion. To turn this option off, specify a value of 0. To see the output for all observations, specify a value of `"all"`.
`res_sort`	Default is `"cooks"`, for specifying Cook's distance as the sort criterion for the display of the rows of data and associated residuals. Other values are `"rstudent"` for Studentized residuals, and `"off"` to not provide the analysis.
`pred`	Default is `TRUE`, which, produces confidence and prediction intervals for each row, or selected rows, of data.
`pred_all`	Default is `FALSE`, which produces prediction intervals only for the first, middle and last five rows of data.

prob_cut — Probability threshold for classifying an observation into the reference group (1) or not (0), applied to the forecasts with prediction intervals as well as to the confusion matrix. Can be a vector, in which case if multiple predictors, the forecasts are for a threshold of 0.5, then the confusion matrices according to the specified values. If a single specified value, then both the forecasts and the one confusion matrix are computed with that value.

cooks_cut — Cutoff value of Cook's Distance at which observations with a larger value are flagged in red and labeled in the resulting scatterplot of Residuals and Fitted Values. Default value is 1.0.

X1_new — Values of the first listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

X2_new — Values of the second listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

X3_new — Values of the third listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

X4_new — Values of the fourth listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

X5_new — Values of the fifth listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

X6_new — Values of the sixth listed predictor variable for which forecasted values and corresponding prediction intervals are calculated.

pdf_file — Name of the pdf file to which graphics are redirected.

width — Width of the pdf file in inches.

height — Height of the pdf file in inches.

... — Other parameter values for R function `glm` which provides the core computations.

Details

OVERVIEW

`Logit` combines the following function calls into one, as well as provide ancillary analyses such as as graphics, organizing output into tables and sorting to assist interpretation of the output. The basic analysis successively invokes several standard R functions beginning with the standard R function for estimation of the logit model, `glm` with `family="binomial"`. The output of the analysis is stored in the object `lm.out`, available for further analysis in the R environment upon completion of the `Logit` function. By default automatically provides the analyses from the standard R functions, `summary`, `confint` and `anova`, with some of the standard output modified and enhanced. The residual analysis invokes `fitted`, `resid`, `rstudent`, and `cooks.distance` functions. The option for prediction intervals calls the standard generic R function `predict`.

The default analysis provides the model's parameter estimates and corresponding hypothesis tests and confidence intervals, goodness of fit indices, the ANOVA table, analysis of residuals and influence as well as the fitted value and standard error for each observation in the model.

DATA
The name d is by default provided by the Read function included in this package for reading and displaying information about the data in preparation for analysis. If all the variables in the model are not in the same data frame, the analysis will not be complete. The data frame does not need to be attached, just specified by name with the data option if the name is not the default d.

The filter parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than. See the Examples.

GRAPHICS
For models with a single predictor variable, a scatter plot of the data is produced, which also includes the fitted values_ As with the density histogram plot of the residuals and the scatterplot of the fitted values and residuals, the scatterplot includes a colored background with grid lines. If more than a single predictor variable, then a scatter plot matrix is produced.

FORECASTS
Fitted and forecasted values are listed for all rows of data if the number of rows is less than 25 or if pred_all=TRUE. If only some of the rows are listed, sorted by the fitted value, the first and last four rows of data are listed. Also the 4 rows immediately around the fitted value of 0.5 are listed.

RESIDUAL ANALYSIS
By default the residual analysis lists the data and fitted value for each observation as well as the residual, Studentized residual, Cook's distance and dffits, with the first 20 observations listed and sorted by Cook's distance. The residual displayed is the actual difference between fitted and observed, that is, with the setting in the residuals of type="response". The res_sort option provides for sorting by the Studentized residuals or not sorting at all. The res_rows option provides for listing these rows of data and computed statistics statistics for any specified number of observations (rows). To turn off the analysis of residuals, specify res_rows=0.

INVOKED R OPTIONS
The options function turns off the stars for different significance levels (show.signif.stars=FALSE), turns off scientific notation for the output (scipen=30), and sets the width of the text output at the console to 120 characters. The later option can be re-specified with the text_width option. After Logit is finished with a normal termination, the options are re-set to their values before the Logit function began executing.

COLORS
The default color theme is "colors", but a gray scale is available with "gray", and other themes are available as explained in style, such as "red" and "green". Use the option style(sub_theme="black") for a black background and partial transparency of plotted colors.

Value

Following the standard R function glm, invisibly returns an object of class inheriting from "glm" which inherits from the class "lm". Particularly useful for comparing nested models. Assign the output of Logit for a model to an object. Then for a nested model. Then use the anova function to compare the models as shown in the examples below.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapter 13, NY: Routledge.

See Also

`formula`, `glm`, `summary.glm`, `anova`, `confint`, `fitted`, `resid`, `rstudent`, `cooks.distance`

Examples

```
# Gender has values of "M" and "F"
d <- Read("Employee", quiet=TRUE)
# logit regression, rely upon default parameter value: data=d
Logit(Gender ~ Years)

# short name
lr(Gender ~ Years)

# Modify the default settings as specified
Logit(Gender ~ Years, res_row=8, res_sort="rstudent", digits_d=8, pred=FALSE)

Logit(Gender ~ Years)

# Multiple logistic regression model with specified probability thresholds
#  for classification into the reference group
# just for employees who have worked more than 5 years at the firm
Logit(Gender ~ Years + Salary, prob_cut=c(.4, .7), filter=(Years > 3))

# Custom contrasts for categorical predictor
d$JobSat <- factor(d$JobSat, levels=c("low", "med", "high"))
contrasts(d$JobSat) <- contr.sum(n=3)
Logit(Gender ~ JobSat)


# Compare nested models
# easier and better treatment of missing data with lessR function:  Nest
full_model <- Logit(Gender ~ Years + Salary)
reduced_model <- Logit(Gender ~ Years)
anova(reduced_model, full_model)

# Save the three plots as pdf files 4 inches square, gray scale
#Logit(Gender ~ Years, pdf_file="MyModel.pdf",
#      width=4, height=4, colors="gray")

# Specify new values of the predictor variables to calculate
#  forecasted values
d <- Read("Cars93")
Logit(Source ~ HP + MidPrice, X1_new=seq(100,250,50), X2_new=c(10,60,10))
```

Merge *Merge Two Data Frames Horizontally or Vertically*

Description

Abbreviation: `mrg`

A horizontal merge combines data frames horizontally, that is, adds variables (columns) to an existing data frame, such as with a common shared ID field. Performs the horizontal merge based directly on the standard R `merge` function. The vertical merge is based on the `rbind` function in which the two data frames have the same variables but different cases (rows), so the rows build vertically, stacked on top of each other.

The advantages of this `lessR` function is that it provides a single function for merging data frames, adds text output to the standard R functions that provide feedback regarding properties of the merge, and provides more detailed and presumably more useful error messages.

Usage

```
Merge(data1, data2, by=NULL, quiet=getOption("quiet"), ...)

mrg(...)
```

Arguments

`data1`	The name of the first data frame from which to create the merged data frame.
`data2`	The name of the second data frame from which to create the merged data frame.
`by`	If a variable specified, then signals a horizontal merge with the ID field by which the data frames are merged as an inner join, that is, only rows of data are retained that both match on the ID. Specify `"rows"` to merge vertically.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	Additional arguments available in the base R `merge` function such as `all.x=TRUE` for an left outer join, which retains all rows of the first data frame even if not matched by a row in the second data table. Specify a right outer join with `all.y=TRUE` and a full outer join, in which all records from both data frames are retained, with `all=TRUE`.

Details

`Merge` creates a merged data frame from two input data frames.

If `by` is specified the merge is horizontal. That is the variables in the second input data frame are presumed different from the variables in the first input data frame. The merged data frame is the combination of variables from both input data frames, with the rows aligned by the value of `by`, an ID field common to both data frames. The result is a *natural join*, a specific instance of an *inner join* in which merging occurs according a common variable.

Invoke merge parameters all.x, all.y, and all, set to TRUE for the corresponding condition. These parameters set, respectively, a *left-outer join*, *right-outer join*, and a *outer join* in which all records from both data frames are retained regardless if a matching row in the other data frame.

Set by to "rows" for a vertical merge. The variables are presumed the same in each input data frame. The merged data frame consists of the rows of both input data frames. The rows of the first data frame are stacked upon the rows of the second data frame.

Guidance and feedback regarding the merge are provided by default. The first five lines of each of the input data frames are listed before the merge operation, followed by the first five lines of the output data frame.

Value

The merged data frame is returned, usually assigned the name of d as in the examples below. This is the default name for the data frame input into the lessR data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

merge, rbind.

Examples

```
# Horizontal
#-----------
d <- Read("Employee", quiet=TRUE)
Emp1a <- d[1:4, .(Years, Gender, Dept, Salary)]
Emp1b <- d[1:4, .(JobSat, Plan)]
# horizontal merge
d <- Merge(Emp1a, Emp1b, by="row.names")
# suppress output to console
d <- Merge(Emp1a, Emp1b, by="row.names", quiet=TRUE)

# Vertical
#---------
d <- Read("Employee", quiet=TRUE)
Emp2a <- d[1:4,]
Emp2b <- d[7:10,]
# vertical merge
d <- Merge(Emp2a, Emp2b, by="rows")
```

Model *Regression Analysis, ANOVA or t-test*

Description

Abbreviation: model, model_brief

Automatically selects and then provides an analysis of a linear model: OLS regression, Logistic regression, ANOVA, or a t-test depending on the proprieties of the data. Comprehensive regression analysis with graphics from a single, simple function call with many default settings, each of which can be re-specified. By default the data exists as a data frame with the default name of d, such as data read by the lessR rad function. Specify the model in the function call according to an R formula, that is, the response variable followed by a tilde, followed by the list of predictor variables, each pair separated by a plus sign.

Usage

```
Model(my_formula, data=d, brief=getOption("brief"), xlab=NULL, ...)

model_brief(..., brief=TRUE)

model(...)
```

Arguments

my_formula	Standard R formula for specifying a model. For example, for a response variable named Y and two predictor variables, X1 and X2, specify the corresponding linear model as Y ~ X1 + X2.
data	The default name of the data frame that contains the data for analysis is d, otherwise explicitly specify.
brief	If set to TRUE, reduced text output. Can change system default with style function.
xlab	x-axis label, defaults to variable name, or, if present, variable label.
...	Other parameter values for R functions such as lm which provide the core computations.

Details

OVERVIEW
The purpose of Model is to combine many standard R function calls into one, as well as provide ancillary analyses such as as graphics, organizing output into tables and sorting to assist interpretation of the output, all from a single function. Currently the supported models are OLS regression, ANOVA and the t-test. For more details of each of these methods, see the lessR functions Regression, Logit, ANOVA and ttest, respectively, which, in turn are based on many standard R functions.

All invocations of the model function are based on the standard R formula.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

formula, lm, glm, summary.lm, anova, confint, fitted, resid, rstudent, cooks.distance

Examples

```
# Generate random data, place in data frame d
n <- 200
X1 <- rnorm(n)
X2 <- rnorm(n)
Y <- .7*X1 + .2*X2 + .6*rnorm(n)
Ybin <- cut(Y, breaks=2, labels=FALSE)
#  instead, if read data with the Read function
#   then the result is the data frame called d
d <- round(data.frame(X1, X2, Y, Ybin),2)
rm(Y); rm(Ybin); rm(X1); rm(X2)

# One-predictor regression
# Provide all default analyses including scatterplot etc.
Model(Y ~ X1)
# alternate form
model(Y ~ X1)

# Multiple regression model
# Provide all default analyses
Model(Y ~ X1 + X2)

# Logit analysis
# Y is binary, 0 or 1
d <- recode(Ybin, old=c(1,2), new=c(0,1), quiet=TRUE)
Model(Ybin ~ X1)

# t-test
Model(breaks ~ wool, data=warpbreaks)

# ANOVA analysis
# from another data frame other than the default \code{d}
# breaks is numerical, wool and tension are categorical
Model(breaks ~ wool + tension, data=warpbreaks)
```

Nest *Nest the Values of an Integer or Factor Variable*

Description

Abbreviation: nt

A nested model has a subset of predictor variables from the corresponding full model. Compare a nested linear model with a full model to evaluate the effectiveness of the predictor variables deleted from the full model to define the nested model.

Usage

```
Nest(y, nested_model, full_model, method=c("lm", "logit"),
         data=d, digits_d=NULL, ...)

nt(...)
```

Arguments

y	Response variable.
nested_model	Predictor variables in the nested model.
full_model	Predictor variables in either the full model, or just those that added to the reduced model to derive the full model.
method	Do a least squares analysis, ls, the default, or set to logit.
data	The name of the data frame from which to create the subset, which is d by default.
digits_d	Number of decimal digits, set by default to at least 2 or the largest number of digits in the values of the response variable plus 1.
...	The specified arguments.

Details

Use the standard R function anova function to compare a nested model with a corresponding full model. By default, compare models estimated with ordinary least squares from the R function lm, or compare models estimated with logistic regression from the R function glm with family="binomial". For the logistic analysis, the anova analysis is with test="Chisq".

To insure that the same data are analyzed for both models, the fit for the full model is first obtained. Then the data frame that is returned by this analysis is input into the analysis for the nested model. This guarantees that any cases with missing data values missing for the full analysis will have been deleted for the nested analysis. Otherwise rows of data could be retained for the nested analysis that were dropped for the full analysis because of missing data values for the deleted predictor variables. This method also guarantees that cases are not deleted because data was missing on variables not included in full analysis.

Value

The output can optionally be returned and saved into an R object, otherwise it simply appears at the console. The components of this object are redesigned in lessR version 3.3 into (a) pieces of text that form the readable output and (b) a variety of statistics. The readable output are character strings such as tables amenable for viewing and interpretation. The statistics are numerical values amenable for further analysis, such as to be referenced in a subsequent R markdown document. The motivation of these three types of output is to facilitate R markdown documents, as the name of each piece, preceded by the name of the saved object followed by a dollar sign, can be inserted into the R markdown document (see `examples`).

TEXT OUTPUT
`out_models`: The specification of the two models compared
`out_anova`: Analysis of variance or, for logit, analysis of deviance

STATISTICS
`fun_call`: Function call that generated the analysis
`anova_tested`: Term that is tested
`anova_residual`: Residual df, and either ss and ms or deviance for logit
`anova_total`: For logit, total df and deviance

Although not typically needed for analysis, if the output is assigned to an object named, for example, `n`, then the complete contents of the object can be viewed directly with the `unclass` function, here as `unclass(n)`. Invoking the `class` function on the saved object reveals a class of `out_all`. The class of each of the text pieces of output is `out`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapter 12, NY: Routledge.

See Also

`anova`, `lm`, `glm`.

Examples

```
d <- Read("Reading")

# compare least-squares models
# can specify all the variables in the full model
Nest(Reading, c(Absent), c(Verbal,Absent,Income))
# or, can specify just the additional variables in the full model
Nest(Reading, c(Absent), c(Verbal,Income))

# compare logistic models, save results into an object
# define the full model by adding just the variables
#  not found in the reduced model
d <- Read("BodyMeas")
n <- Nest(Gender, c(Weight, Hips, Hand, Shoe),
      c(Height, Waist, Chest), method="logit")
# view the results
n
```

```
# see the names of the available output components
names(n)
```

PieChart *Pie Chart*

Description

Abbreviation: pc

Plots a pie chart of a categorical variable (x). The default chart is a doughnut or ring version of a pie chart, that is, a hole in the middle of the pie. Either directly enter the corresponding numerical value (y) or have the numerical variable be the tabulated counts for the frequency of occurrence for each value of the categorical variable. Also displays the frequency table for the variable with the corresponding chi-square inferential analysis. Real numbers can also be entered directly.

Usage

```
PieChart(x, y=NULL, data=d, filter=NULL,

         radius=1, hole=0.65, hole_fill=getOption("panel_fill"),

         theme=getOption("theme"),
         fill=NULL,
         color="lightgray",
         transparency=getOption("trans_bar_fill"),

         density=NULL, angle=45,
         lty="solid", lwd=1, edges=200,

         clockwise=FALSE, init_angle=ifelse (clockwise, 90, 0),

         labels=getOption("labels"),
         labels_color=getOption("labels_color"),
         labels_size=getOption("labels_size"),
         labels_digits=getOption("labels_digits"),
         labels_position=getOption("labels_position"),

         main=NULL, main_cex=getOption("main_cex")*1.2,
         labels_cex=getOption("lab_cex"), cex,

         add=NULL, x1=NULL, y1=NULL, x2=NULL, y2=NULL,

         rows=NULL,

         eval_df=NULL, quiet=getOption("quiet"),
         width=6.5, height=6, pdf_file=NULL,
         ...)
```

```
pc(...)
```

Arguments

- `x` — For each level of this categorical variable, x, display the frequencies as slices of a pie.
- `y` — Numeric variable that sets the area of each slice of the pie. If not specified, then its value is the frequency of each category of x, automatically tabulated. Applied to reading the summary table of the already aggregated data as the data for the pie chart.
- `data` — Optional data frame that contains the variable(s) of interest, default is d.
- `filter` — A logical expression that specifies a subset of rows of the data frame to analyze.
- `radius` — The pie is drawn in a box with sides that range from -1 to 1, so the maximum value of the radius without truncating the pie is 1.
- `hole` — The proportion of the radius that defines the inner hole for what is called a doughnut or hole plot. To show the full pie, set to FALSE or the value of 0.
- `hole_fill` — Fill color of the hole, which by default is the same color as panel_fill as set by the color theme or individually with the style function.
- `theme` — Selected color theme, change with style function.
- `fill` — Specified color of each slice. Default is the discrete scale with, with fixed chroma (50) and luminance (75) for unbiased comparison across colors, for all color themes except "gray" and "white", with default gray scale. Can explicitly choose "grays" or "hues" , or pre-specified R color schemes "rainbow", "terrain", and "heat". Or, set to the name of y to map the values of bar fill, specified as (count) if tabulated from the data. Can also provide pre-defined color ranges "blues", "reds" and "greens", as well as custom colors, such as generated by getColors.
- `color` — Border color of sides and the pie, can be a vector to customize the color for each slice. Default is bar_color from the lessR style function.
- `transparency` — Transparency factor of the area of each slice. Default is trans_bar_fill from the lessR style function.
- `density` — Density of shading lines, in lines per inch. Default value is NULL, that is, no shading lines.
- `angle` — Angle of shading lines in degrees.
- `lty` — Type of line that borders each slice, such as "solid", the default. Can be a vector. Acceptable values are "blank", "solid", "dashed", "dotted", "dotdash", and "longdash".
- `lwd` — Width of line that borders each slice.
- `edges` — Approximation of a circle with a polygon drawn with the number of specified edges.

clockwise
: Default value of FALSE specifies to draw the slices counter-clockwise, otherwise clockwise.

init_angle
: Starting angle (in degrees) for the slices. For counter-clockwise the default value is 0 (3 o'clock), otherwise 90 (12 o'clock).

labels
: If not the default value of "off", adds the numerical results to the plot according to "%", "prop" or "input", that is, percentages, proportions, or the value from which the slices are plotted, such as tabulated counts if y is not specified, or the value of y if the plotted values are provided. If any other labels parameter is specified, default is set to "%".

labels_color
: Color of the plotted text. Could be a vector to specify a unique color for each value. If fewer colors are specified than the number of categories, the colors are recycled.

labels_size
: Character expansion factor, the size, of the plotted text, for which the default value is 0.95.

labels_digits
: Number of decimal digits for which to display the values_ Default is 0, round to the nearest integer, for "%" and 2 for "prop".

labels_position
: Position of the plotted text. Default is inside the pie, or, if "label", as part of the label for each value outside of the pie.

main
: Title of graph. Set the color with main_color with the style function.

main_cex
: Character expansion factor of title relative to 1.

labels_cex
: Character expansion factor of labels relative to 1. No labels if set to 0.

cex
: General character expansion factor for default values of main_cex, labels_cex, and values_size. Useful for adjustment of text for larger or smaller images.

add
: **Draw one or more objects**, text or a geometric figures, on the plot. Possible values are any text to be written, the first argument, which is "text", or, to indicate a figure, "rect" (rectangle), "line", "arrow", "v.line" (vertical line), and "h.line" (horizontal line). The value "means" is short-hand for vertical and horizontal lines at the respective means. Does not apply to Trellis graphics. Customize with parameters such as fill and color from the style function.

x1
: First x coordinate to be considered for each object. All coordinates vary from -1 to 1.

y1
: First y coordinate to be considered for each object.

x2
: Second x coordinate to be considered for each object. Only used for "rect", "line" and arrow.

y2
: Second y coordinate to be considered for each object. Only used for "rect", "line" and arrow.

rows
: Deprecated old parameter name that is now called filter.

eval_df	Determines if to check for existing data frame and specified variables. By default is TRUE unless the shiny package is loaded then set to FALSE so that Shiny will run. Needs to be set to FALSE if using the pipe %\>% notation.
quiet	If set to TRUE, no text output. Can change system default with style function.
width	Width of the plot window in inches, defaults to 4.5.
height	Height of the plot window in inches, defaults to 4.5.
pdf_file	Name of the pdf file to if graphics to be redirected to a pdf file.
...	Other parameter values for graphics as defined processed by pie and par for general graphics, which includes radius of the pie, and color_main for the title of the graph.

Details

OVERVIEW
Plot a pie chart with default colors, presumably with a relatively small number of values for each variable. By default, colors are selected for the slices, background and grid lines, all of which can be customized. The basic computations of the chart are provided with the standard R functions pie and chisq.test and the lessR function chisq.test. A minor modification of the original pie code provides for the hole in the middle of the pie, the default doughnut or ring chart.

DATA
The data may either be a vector from the global environment, the user's workspace, as illustrated in the examples below, or one or more variable's in a data frame, or a complete data frame. The default input data frame is d. Can specify the source data frame name with the data option. If multiple variables are specified, only the numerical variables in the list of variables are analyzed. The variables in the data frame are referenced directly by their names, that is, no need to invoke the standard R mechanisms of the d$name notation, the with function or the attach function. If the name of the vector in the global environment and of a variable in the input data frame are the same, the vector is analyzed.

The rows parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than. See the Examples.

COLORS
Set the default color of the bars by the current color theme according to bar_fill_discrete argument of the function style, which includes the default color theme "colors" that defines a qualitative HCL color scale, or set the bar color with the fill parameter. These parameters reference a specified vector of color specifications, such as generated by the lessR getColors function.

Set fill to a single color or a color palette, of which there are many possibilities. Define a qualitative color palette with "hues" that provides HCL colors of the same chroma (saturation) and luminance (brightness). Also available are the pre-specified R color palettes "rainbow", "terrain", and "heat". Pre-defined sequential and divergent color ranges are available as implicit calls to getColors. The full list of pre-defined color ranges (defined in 30 degree increments around the HCL color wheel): "reds", "rusts", "browns", "olives", "greens", "emeralds", "turquoises", "aquas", "blues", "purples","violets", "magentas", and "grays".

Defines a *sequential color scale* with single value of `fill` for a pre-defined palette such as `"blues"`. Or, *manually specify colors*. For example, for a two-level by variable, could set `fill` to `c("coral3","seagreen3")`, where the specified colors are *not* pre-defined color ranges.

For the pre-defined color scales can obtain more control over the obtained color palettes with an explicit call to `getColors` for the argument to `fill`. Here the value of chroma (`c`) and luminance (`l`) can be explicitly manipulated in conjunction with the specification of a pre-defined color range. Or, create a custom color range for any value of hue (`h`). See `getColors` for more information.

To change the background color, set the `"panel_fill"` argument of the `style` function. The hole of the pie defaults to that color, which, of course, can also be specified to a different color_

ANNOTATIONS
Use the `add` and related parameters to annotate the plot with text and/or geometric figures_ Each object is placed according from one to four corresponding coordinates, the required coordinates to plot that object, as shown in the following table. The values of the coordinates vary from -1 to 1.

Value	Object	Required Coordinates
text	text	x1, x2
`"rect"`	rectangle	x1, y1, x2, y2
`"line"`	line segment	x1, y1, x2, y2
`"arrow"`	arrow	x1, y1, x2, y2

The value of `add` specifies the object. For a single object, enter a single value. Then specify the value of the needed corresponding coordinates, as specified in the above table. For multiple placements of that object, specify vectors of corresponding coordinates. To annotate multiple objects, specify multiple values for `add` as a vector. Then list the corresponding coordinates, for up to each of four coordinates, in the order of the objects listed in `add`. See the examples for illustrations.

Can also specify vectors of different properties, such as `add_color`. That is, different objects can be different colors, different transparency levels, etc.

STATISTICS
In addition to the pie chart, descriptive and inferential statistics are presented. First, for integer variables such as counts, the frequency table with proportions is displayed. Second, the corresponding chi-square test is also displayed. For real valued variables read from a data frame, the summary statistics such as the mean are reported.

PDF OUTPUT
Because `lessR` functions generate their own graphics calls, the standard graphic output functions such as `pdf` do not work with the `lessR` graphics functions. Instead, to obtain pdf output, use the `pdf_file` option, perhaps with the optional `width` and `height` options. These files are written to the default working directory, which can be explicitly specified with the R `setwd` function.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a `lessR` function can only be a variable name. This referenced variable must exist in either the referenced data frame, `d` by default, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

> PieChart(rnorm(10)) # does NOT work

Instead, do the following:

```
> Y <- rnorm(10)   # create vector Y in user workspace
> PieChart(Y)     # directly reference Y
```

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 3, NY: CRC Press.

See Also

pie, chisq.test.

Examples

```
# get the data from a file included with lessR
d <- rd("Employee")

# --------------------------------------------------------
# pie (doughnut) chart from the data for a single variable
# --------------------------------------------------------

# basic pie chart, actually a doughnut or ring chart
# with default hcl colors (except for themes "gray" and "white")
PieChart(Dept)
# short name
#pc(Dept)

# standard pie chart with no hole
pc(Dept, hole=0)

# specify a unique slice color for each of the two slices
# turn off borders
PieChart(Gender, fill=c("pink","lightblue"), lty="blank")

# just males with a salary larger than 75000 USD
PieChart(Dept, rows=(Gender=="M" & Salary > 75000))

# use getColors function to create the pie slice colors
# here as a separate function call
# need to set the correct number of colors to span the full range
mycolors <- getColors("aliceblue", end_pal="steelblue", n=5)
PieChart(Dept, fill=mycolors)

# specify the colors from a predefined color palette
# see ?getColors
PieChart(Dept, fill="blues")
```

```
# viridis color palette
PieChart(Dept, fill="viridis")

# display the percentage inside each slice of the pie
# provide a unique color for each displayed value
PieChart(Dept, labels="%",
         labels_color=c("yellow", "pink", "blue", "purple", "brown"))

# display the counts inside each slice of the pie
# reduce size of displayed counts to 0.75
PieChart(Dept, labels="input", labels_size=0.75,
         labels_color=getOption("window_fill"))

# add transparency and custom color for the displayed values
PieChart(Dept, transparency=.6, labels="%", labels_color=rgb(.3,.3,.3))

# map counts of each level to the fill color of the corresponding slice
PieChart(JobSat, fill=(count))


# ------------------------------
# pie chart directly from counts
# ------------------------------

# from vector
# pie chart of one variable with three levels
# enter counts as a vector with the combine function, c
# must supply the level names and variable name
# use abbreviation pc for PieChart
City <- c(206, 94, 382)
names(City) <- c("LA","Chicago","NY")
pc(City, main="Employees in Each City")

# counts from data frame
x <- c("ACCT", "ADMN", "FINC", "MKTG", "SALE")
y <- c(5, 6, 4, 6, 15)
d <- data.frame(x,y)
names(d) <- c("Dept", "Count")
PieChart(Dept, Count)

# real numbers from data frame
Dept <- c("ACCT", "ADMN", "FINC", "MKTG", "SALE")
Salary <- c(86208.42, 29808.29, 42305.52, 75855.81, 65175.51)
d <- data.frame(x,y)
pc(Dept, Salary)
rm(Dept)
rm(Salary)

# -----------
# annotations
# -----------
```

```
d <- rd("Employee")

# Place a message in the center of the pie
# Use \n to indicate a new line
PieChart(Dept, add="Employees by\nDepartment", x1=0, y1=0)

# Use style to change some parameter values
style(add_trans=.8, add_fill="gold", add_color="gold4", add_lwd=0.5)
# Add a rectangle around the message centered at <0,0>
PieChart(Dept, add=c("rect", "Employees by\nDepartment"),
                     x1=c(-.4,0), y1=c(-.2, 0), x2=.4, y2=.2)
```

---

pivot *Create a Pivot (Summary) Table*

Description

Compute one or more designated descriptive statistics (`compute` over one or more numerical variables (`variable`) either for all the data or aggregated over one or more categorical variables (`by`). Because the output is a two-dimensional table, select any two of the three possibilities: Multiple compute functions for the descriptive statistics, multiple continuous variables over which to compute, and multiple categorical variables by which to define groups for aggregation. Displays the sample size for each group. Uses the base R function `aggregate` for which to perform the aggregation.

Usage

```
pivot(data, compute, variable, by=NULL, by_cols=NULL, filter=NULL,
         show_n=TRUE, na_by_show=TRUE, na_remove=TRUE, na_group_show=TRUE,
         out_names=NULL, sort=NULL, sort_var=NULL,
         table_prop=c("none", "all", "row", "col"), table_long=FALSE,
         factors=TRUE, q_num=4, digits_d=NULL, quiet=getOption("quiet"))
```

Arguments

`data`	Data frame that contains the variables.
`compute`	One or more statistics, defined as one or more functions, to aggregate over the combinations of the values of the categorical variables.
`variable`	One or more numeric response variables for which to `compute` the specified statistics, perhaps aggregated, i.e., summarized across the groups.
`by`	Categorical variables that define the groups (cells) listed in the rows of the output long-form data frame, available to input into other data analysis routines. Ignore to compute over the variables for all the data, e.g., the grand mean.
`by_cols`	Up to two categorical variables that define the groups displayed as columns in a two dimensional table.

filter
: Subset, i.e., filter, rows of the input data frame for analysis.

show_n
: By default, display the sample size and number missing for each computed summary statistic. If FALSE, delete all variables from the output data frame that end with n_ or na_.

na_by_show
: If TRUE, the default, if all values of 'variable' are missing for a group so that the entire level of the 'by' variables is missing, show those missing cells with a reported value of computed variable n as 0. Otherwise delete the row from the output.

na_remove
: Sets base R parameter na.rm. If TRUE, the default, removes missing values from the variable(s), then reports how many values were missing. Otherwise, the aggregation statistic for a cell with any missing data returns NA.

na_group_show
: If TRUE, the default, display <NA> for missing data of a grouping variable as a level for that variable. Otherwise, do not treat a missing value of a group as a level for which to aggregate.

out_names
: Custom names for the aggregated variables. If more than one, list in the same order as specified in variable. Does not apply to the table option where the column names are the levels of the by variable(s).

sort
: Set to "+" for an ascending sort or "-" for a descending sort according to the last variable in the output data frame.

sort_var
: Either the name of the variable in the output data frame to sort, or its column number. Default is the last column.

table_prop
: Applies to a created table for the value of compute. Default value of "none" leaves frequencies. Value of "all" converts to cell proportions based on the grand total. Values of "row" and "col" provide proportions based on row and column sums.

table_long
: Applies to the value of compute of table. If set to TRUE, then the cross-tabs table is output in long form, one count per row.

factors
: For by variables of type character and integer, converted to factors in the summary table by default, except for Date variables that always retain their type. If FALSE, then the by variables retain their original character or integer type.

q_num
: For the computation of quantiles, number of intervals. Default value of 4 provides quartiles.

digits_d
: Number of significant digits for each displayed summary statistic. Trailing zeros are deleted, so, for example, integers display as integers. If not specified, defaults to 3 unless there are more than 3 decimal digits and only a single digit to the left of the decimal point. Then enough digits are displayed to capture some non-zero decimal digits to avoid rounding to 0.000. To see all digits without trailing decimal 0's, set at a large number such as 20.

quiet
: If set to TRUE, no text output. Can change system default with style function.

Details

pivot uses base R aggregate to generate a pivot table (Excel terminology). Express multiple categorical variables over which to pivot as a vector with the c function.

pivot provides two additional features than aggregate provides. First is a complete missing data analysis. If there is no missing data for the numerical variables that are aggregated, then the cell sizes are included with the aggregated data. If there is such missing data, then the amount of available data is displayed for all values to be aggregated for each cell.

The second is that the data parameter is listed first in the parameter list, which facilitates the use of the pipe operator from the magrittr package. Also, there is a different interface as the by variables are specified as a vector.

Variable ranges in the specification of by are not needed in general. Only a small number of grouping variables generally define the cells for the aggregation.

The following table lists available single summary statistics. The list is not necessarily exhaustive as the references are to functions provided by base R, including any not listed below.

Statistic	Meaning
sum	sum
mean	arithmetic mean
median	median
min	minimum
max	maximum
sd	standard deviation
var	variance
skew	skew
kurtosis	kurtosis
IQR	inter-quartile range
mad	mean absolute deviation

The functions skew() and kurtosis() are provided by this package as they have no counterparts in base R. All other functions are from base R.

The quantile and table statistical function returns multiple values.

Statistic	Meaning
quantile	min, quartiles, max
table	frequencies or proportions

The table computation applies to an aggregated variable that consists of discrete categories, such as the numbers 1 through 5 for responses to a 5-pt Likert scale. The result is a table of frequencies or proportions, a contingency table, referred to for two or more variables as a cross-tabulation table or a joint frequency distribution. Other statistical functions can be simultaneously computed with table, though only meaningful if the aggregated variable consists of a relatively small set of discrete, numeric values.

The default quantiles for quantile are quartiles. Specify a custom number of quantiles with the q_num parameter, which has the default value of 4 for quartiles.

Value

Returns a data frame of the aggregated values, unless for two by variables and table_2d is TRUE, when a table is returned.

The count of the number of elements in each group is provided as the variable n. If a combination of by variable levels that defines a group is empty, the n is set to 0 with the values of the variable set to NA.

The number of missing elements of the value variable is provided as the variable miss.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

aggregate.

Examples

```
library(knitr)  # for kable() called from pivot()
d <- Read("Employee", quiet=TRUE)

# parameter values named
pivot(data=d, compute=mean, variable=Salary, by=c(Dept, Gender))

# visualize the aggregation
# when reading a table of coordinates, a, BarChart cannot deal with
#   with missing data so do not show groups that are missing as
#   another level
a <- pivot(d, mean, Salary, c(Dept, Gender), na_group_show=FALSE)
BarChart(Dept, Salary_mean, by=Gender, data=a)

# calculate mean of Years and Salary for each combination of Dept and Gender
# parameter values by position
pivot(d, mean, c(Years, Salary), c(Dept, Gender))

# output as a 2-d cross-tabulation table
pivot(d, mean, Salary, Dept, Gender)

# cross-tabulation table
pivot(d, table, Dept, Gender)
# long form
pivot(d, table, Dept, Gender, table_long=TRUE)

# multiple functions for which to aggregate
pivot(d, c(mean,sd,median,IQR), Years, c(Gender,Dept), digits_d=2)
```

```
# A variety of statistics computed for several variables over the
#  entire data set without aggregation
pivot(d, c(mean,sd,skew,kurtosis), c(Years,Salary,Pre,Post), digits_d=2)
```

`Plot` *Scatterplots including Time Series and Violin/Box/Scatterplot*

Description

Abbreviation:
Violin Plot only: `vp`, `ViolinPlot`
Box Plot only: `bx`, `BoxPlot`
Scatter Plot only: `sp`, `ScatterPlot`

A scatterplot displays the values of a distribution, or the relationship between the two distributions in terms of their joint values, as a set of points in an *n*-dimensional coordinate system, in which the coordinates of each point are the values of *n* variables for a single observation (row of data). From the identical syntax, from any combination of continuous or categorical variables variables `x` and `y`, `Plot(x)` or `Plot(x,y)`, where `x` or `y` can be a vector, by default generates a family of related 1- or 2-variable scatterplots, possibly enhanced, as well as related statistical analyses. Define a categorical variable as an R factor. If `x` is a Date variable, then a time series is plotted.

`Plot` produces a wide variety of scatterplots as outlined in the following list.

Variable Type	**Meaning**
`x`, `y`, or `z`	single continuous variable
`xDate`	date variable, defined as a R Date type
`xCat`, `yCat`, or `zCat`	categorical variable, typically defined as an R factor
`xUnique` or `yUnique`	categorical variable with all values unique
`X` or `Y`	vector of continuous variables
`Xcat`	vector of categorical variables

Two variables
`Plot(x,y)`: traditional scatterplot of two continuous variables
`Plot(xDate,y)`: a Date variable and a continuous yields a time-series plot
`Plot(xCat,yCat)`: to solve the over-plot problem, plot a scatterplot of two categorical variables as a bubble scatterplot, the size of each bubble based on the corresponding joint frequency
`Plot(xCat,y)` or `Plot(x,yCat)`: one variable categorical and the other variable continuous, yields a scatterplot with means at each level of the categorical variable
`Plot(xCat,y, stat="mean")` or `Plot(x,yCat, stat="mean")`: one variable categorical and the other variable continuous, yields a Cleveland dot plot with a specified statistic such as the `"mean"` of the continuous variable at each level of the categorical variable
`Plot(xUnique,y)` or `Plot(x,yUnique)`: one categorical with unique (ID) values and the other variable continuous, yields a Cleveland dot (lollipop) plot, where the unique values can be variable

`row.names`

One variable
`Plot(x)`: one continuous variable generates either a violin/box/scatterplot (VBS plot), named here, or a run chart with `run=TRUE`, or `x` can be an R time series object created with `ts()` for a time series visualization
`Plot(xCat)`: one categorical variable yields a 1-dimensional bubble plot to solve the over-plot problem for a more compact replacement of the traditional bar chart

Three, four, or more variables
`Plot(x,y, size=z)`: `x` and `y` continuous yields a bubble of two continuous variables with `z` setting the size of the corresponding plotted point, i.e., bubble
`Plot(x,y, by=zCat)`: plots a different scatterplot of `x` and `y` for each level of `zCat` on the same panel
`Plot(x,y, by1=zCat)`: plots a different scatterplot of `x` and `y` for each level of `zCat` on separate panels, i.e., Trellis or facet plots
`Plot(x,y, by1=z1Cat, by2=z2Cat)`: plots a different scatterplot of `x` and `y` for each combination of levels of `zCat1` and `zCat2` on separate panels, i.e., Trellis or facet plots
`Plot(X,y)` or `Plot(x,Y)`: one vector variable of several continuous variables, paired with another single continuous variable, yields multiple scatterplots on the same graph
`Plot(Y,xUnique)`: one categorical with unique (ID) values, such as `row.names` and the other variable a vector of continuous variables yields a Cleveland dot plot of all the continuous variables, usually two

One vector
`Plot(X)`: one vector of variables, with no `y`-variable, results in a scatterplot matrix of the variables
`Plot(Xcat)`: one vector of categorical `x`-variables, with no `y`-variable, generalizes to a matrix of 1-dimensional bubble plots, here called the bubble plot frequency matrix, to replace a series of bar charts

Usage

```
Plot(

    # ------------------------------------
    # Data from which to construct the plot
    x, y=NULL, data=d, filter=NULL,


    # ------------------------------
    # Enhancements and customizations
    # ------------------------------

    # ------------------------------------------------------------------
    # Analogy of physical Marks on paper that create the bars and labels
    theme=getOption("theme"),
    fill=NULL, color=NULL,
    transparency=getOption("trans_pt_fill"),
```

```
enhance=FALSE,
size=NULL, size_cut=NULL, shape="circle", means=TRUE,
segments=FALSE, segments_y=FALSE, segments_x=FALSE,

# ----------------------
# Sort and jitter points
sort_yx=c("0", "-", "+"),
jitter_x=0, jitter_y=0,

# ----------------
# Outlier analysis
ID="row.name", ID_size=0.60,
MD_cut=0, out_cut=0, out_shape="circle", out_size=1,

# ------------------------------------------------
# Fit line, confidence interval, confidence ellipse
fit=c("off","loess", "lm", "ls", "null", "exp", "quad",
      "power", "log"),
fit_power=1, fit_se=0.95,
fit_color=getOption("fit_color"),
plot_errors=FALSE, ellipse=0,


# ---------------------------------------------------------
# Types of plots beyond default scatterplots (x, or x and y)
# ---------------------------------------------------------

# -------------------------------------------------
# Stratification: Same panel or Trellis (facet) plot [x, or x and y]
by=NULL, by1=NULL, by2=NULL,
n_row=NULL, n_col=NULL, aspect="fill",

# ----------------------------------------------------
# Time series, plot x values sequentially [xDate, y or Y]
time_unit=NULL, time_agg=c("sum","mean"), stack=FALSE, lwd=1.5,
area_fill="transparent", area_split=0,
# Run chart
run=FALSE, show_runs=FALSE,
center_line=c("off", "mean", "median", "zero"),

# ----------------------------------
# Lollipop chart from aggregated data [x and y]
stat=c("mean", "sum", "sd", "deviation", "min", "median", "max"),
stat_x=c("count", "proportion", "%"),

# ---------------------------------
# Integrated violin/box/scatter plot  [x]
```

```
vbs_plot="vbs", vbs_size=0.9, bw=NULL, bw_iter=10,
violin_fill=getOption("violin_fill"),
box_fill=getOption("box_fill"),
vbs_pt_fill="black",
vbs_mean=FALSE, fences=FALSE,
k=1.5, box_adj=FALSE, a=-4, b=3,

# -----------
# Bubble plot [xCat, or xCat and yCat]
radius=NULL, power=0.5, low_fill=NULL, hi_fill=NULL,

# -------------------------------------
# Large data sets, smoothing and binning  [x and y]
smooth=FALSE, smooth_points=100, smooth_size=1,
smooth_exp=0.25, smooth_bins=128,
n_bins=1,

# -----------------------------------------------------
# Bins for frequency polygon or text output of VBS plots
bin=FALSE, bin_start=NULL, bin_width=NULL, bin_end=NULL,
breaks="Sturges", cumulate=FALSE,


# -------------
# Miscellaneous
# -------------

# -----------------------------------------------------------------
# Labels for axes, values, and legend if x and by variables, margins
xlab=NULL, ylab=NULL, main=NULL, sub=NULL,
lab_adjust=c(0,0), margin_adjust=c(0,0,0,0),

rotate_x=getOption("rotate_x"), rotate_y=getOption("rotate_y"),
offset=getOption("offset"),

xy_ticks=TRUE, origin_x=NULL,
scale_x=NULL, scale_y=NULL,
pad_x=c(0,0), pad_y=c(0,0),
legend_title=NULL,

# ---------------------------------------------------
# Add one or more objects, text, or geometric figures
add=NULL, x1=NULL, y1=NULL, x2=NULL, y2=NULL,

# -----------------------------------------------------------------
# Output: turn off, chart to PDF file, decimal digits, markdown file
quiet=getOption("quiet"), do_plot=TRUE,
pdf_file=NULL, width=6.5, height=6,
```

```
    digits_d=NULL,

    # -----------------------------------------------------------
    # Deprecated, removed in future versions, use R factors instead
    n_cat=getOption("n_cat"), value_labels=NULL, rows=NULL,

    # -----
    # Other
    eval_df=NULL, fun_call=NULL, ...)


ScatterPlot(...)
sp(...)
BoxPlot(...)
bx(...)
ViolinPlot(...)
vp(...)
```

Arguments

`x`	By itself, or with `y`, by default, a *primary variable*, that is, plotted by its values mapped to coordinates. The **data values** can be continuous or categorical, cross-sectional or a time series. If `x` is sorted, with equal intervals separating the values, or is a time series, then by default plots the points sequentially, joined by line segments. Can specify multiple `x`-variables or multiple `y`-variables as vectors, but not both. Can be in a data frame or defined in the global environment.
`y`	An optional second *primary variable.* Variable with values to be mapped to coordinates of points in the plot on the vertical axis. Can be continuous or categorical. Can be in a data frame or defined in the global environment.
`data`	Optional data frame that contains one or both of `x` and `y`. Default data frame is `d`.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`theme`	Color theme for this analysis. Make persistent across analyses with `style`.
`fill`	Either fill color of the points or the area under a line chart. Can also set with the lessR function `getColors` to select from a variety of color palettes. For points, default is `pt_fill` and for area under a line chart, `violin_fill`. For a line chart, set to `"on"` for default color.
`color`	Border color of the points or line_color for line plot. Can be a vector to customize the color for each point or a color range such as "blues" (see `getColors`. Default is `pt_color` from the lessR `style` function.
`transparency`	Transparency factor of the fill color of each point. Default is `trans_pt_fill` from the lessR `style` function.
`enhance`	For a two-variable scatterplot, if `TRUE`, automatically add the 0.95 data ellipse, labeling of outliers beyond a Mahalanobis distance of 6 from the ellipse center,

the best-fitting least squares line of all the data, the best-fitting least squares line of the regular data without the outliers, and a horizontal and vertical line to represent the mean of each of the two variables.

`size`
: When set to a constant, the scaling factor for **standard points** (not bubbles) or a line, with default of 1.0 for points and 2.0 for a line. Set to 0 to not plot the points or lines. If `area_fill` for a line chart, then default is 0. When set to a variable, activates a bubble plot with the size of each bubble further determined by the value of `radius`. Applies to the standard two-variable scatterplot as well as to the scatterplot component of the integrated Violin-Box-Scatterplot (VBS) of a single continuous variable.

`size_cut`
: If `1` (or `TRUE`), then for a bubble plot in which the bubble sizes are defined by a `size` variable, show the value of the sizing variable for selected bubbles in the center of the bubbles, unless the bubble is too small. If `0` (or `FALSE`), no value is displayed. If a number greater than 1, then display the value only for the indicated number of values, such as just the max and min for a setting of 2, the default value when bubbles represent a size variable. Color of the displayed text set by `bubble_text` from the `style` function.

`shape`
: The plot character(s). The default value is `"circle"` with both a default exterior color and filled interior, explicitly specified with `"color"` and `"fill"`. Other possible values, with fillable interiors, are `"circle"`, `"square"`, `"diamond"`, `"triup"` (triangle up), and `"tridown"` (triangle down), all uppercase and lowercase letters, all digits, and most punctuation characters. The numbers 0 through 25 as defined by the R `points` function also apply. If plotting levels for different groups according to `by`, then list as a vector with with one shape for each level to be plotted or set to `"vary"` to have shapes selected by default across the `by` groups.

`means`
: If the one variable is categorical, expressed as a factor, and the other variable continuous, then if `TRUE`, by default, plot means with the scatterplot. Also applies to a 1-D scatterplot.

`segments`
: Designed for interaction plots of means, connects each pair of successive points with a line segment. Pass a data frame of the means, such as from `pivot`. To turn off connecting line segments for sorted, equal intervals data, set to `FALSE`. Currently, does not apply to Trellis plots.

`segments_y`
: For one `x`-variable, draw a line segment from the `y`-axis to each plotted point, such as for the Cleveland dot plot. For two `x`-variables, the line segments connect the two points.

`segments_x`
: Draw a line segment from the `x`-axis for each plotted point.

`sort_yx`
: Sort the values of `y` by the values of `x`, such as for a Cleveland dot plot, that is, a numeric `x`-variable paired with a categorical `y`-variable with unique values. If a `x` is a vector of two variables, sort by their difference.

`jitter_x`
: Randomly perturbs the plotted points of a scatterplot horizontally within the limits of the explicitly specified value, or set to `NULL` to rely upon the default value of the range of `x` divided by 50.

`jitter_y`
: Same as `jitter_x` except vertical jitter.

`ID`	Name of variable to provide the **labels for the selected plotted points for outlier identification**, row names of data frame by default. To label all the points use the `add` parameter described later.
`ID_size`	Size of the plotted labels. Modify text color of the labels with the `style` function parameter `ID_color`.
`MD_cut`	Mahalanobis distance cutoff to define an outlier in a 2-variable scatterplot.
`out_cut`	Count or proportion of plotted points to label, in order of their distance from the scatterplot center (means), counting down from the more extreme point. For two-variable plots, assess distance from the center with Mahalanobis distance. For VBS plots of a single continuous variable, refers to outliers on each side of the plot.
`out_shape`	Shape of outlier points in a 2-variable scatterplot or a VBS plot. Modify fill color from the current `theme` with the `style` function parameters `out_fill` and `out2_fill`.
`out_size`	Size of outlier points in a 2-variable scatterplot or VBS plot.
`fit`	The **best fit line**. Default value is `"off"`, with options `"loess"` for non-linear fit, `"lm"` for linear model least squares, `"null"` for the null model, `"exp"` for exponential growth or decay, `"power"` for the general power model in conjunction with `fit_power`, and `"quad"` for an increasing or decreasing function for the specific power value of 2. If potential outliers are identified according to `out_cut`, a second (dashed) fit line is displayed calculated *without* the outliers.
`fit_power`	Power that describes response Y as a power function of the predictor variable X, required for `"power"` . Optionally, and experimentally, applies to `fit` values `"exp"`.
`fit_se`	Confidence level for the error band displayed around the line of best fit. On by default at 0.95 if a fit line is specified, but turned off if `plot_errors=TRUE`. Can be a vector to display multiple ranges. Set to 0 to turn off.
`fit_color`	Color of the fit line.
`plot_errors`	Plot the line segment that joins each point to the regression line, "loess" or "lm", illustrating the size of the residuals.
`ellipse`	Confidence level of a data ellipse for a scatterplot of only a single `x`-variable and a single `y`-variable according to the contours of the corresponding bivariate normal density function. Can specify the confidence level(s) for a single or vector of numeric values from 0 to 1, to plot one or more specified ellipses. For Trellis graphics, only the maximum level applies with only one ellipse per panel. Modify fill and border colors with the `style` function parameters `ellipse_fill` and `ellipse_color`.
`by`	A categorical variable to provide a scatterplot for each level of the numeric primary variables `x` and `y` on the *same* plot, a *grouping variable*. For two-variable plots, applies to the panels of a **Trellis graphic** if `by1` is specified.
`by1`	A categorical variable called a *conditioning variable* that activates Trellis graphics, provided by Deepayan Sarkar's (2007) lattice package, to provide a *separate*

panel of numeric primary variables x and y for each level of the variable. Reorder the levels by first converting to a factor variable with factor or lessR factors.

by2
: A second *conditioning variable* to generate Trellis plots jointly conditioned on both the by1 and by2 variables, with by2 as the row variable, which yields a scatterplot (panel) for *each* cross-classification of the levels of numeric x and y variables.

n_row
: Optional specification for the number of rows and columns in the layout of a multi-panel display with Trellis graphics. Specify n_col or n_row, but not both.

n_col
: Optional specification for the number of columns in the layout of a multi-panel display with Trellis graphics. Specify n_col or n_row, but not both. If set to 1, the default, then the strip that labels each group locates to the left of each plot instead of the top.

aspect
: Lattice parameter for the aspect ratio of the panels, defined as height divided by width. The default value is "fill" to have the panels expand to occupy as much space as possible. Set to 1 for square panels. Set to "xy" to specify a ratio calculated to "bank" to 45 degrees, that is, with the line slope approximately 45 degrees.

time_unit
: Specify the time unit from which to plot a time series. Aggregation according to the time unit will occur as needed, such as a daily time series aggregated to "years", "quarters" or "months". See details below for all possible values.

time_agg
: Function by which to aggregate according to time_unit. Default is "sum".

stack
: If TRUE, multiple time plots are stacked on each other, with area set to TRUE by default.

lwd
: Width of the line segments. Set to zero to remove the line segments.

area_fill
: Specifies the area under the line segments, if present. If stack is TRUE, then default is gradation from default color range, e.g., "blues". If not specified, and fill is specified with no plotted points and area_fill is not specified,, then fill generally specifies the area under the line segments.

area_split
: [Applies only to a Trellis plot activated with parameter by1.] Value of y that defines a reference line that splits the filled area under the time series line. Values of y less than this value are below the corresponding reference line, values larger are above the line.

run
: If set to TRUE, generate a **run chart**, i.e., line chart, in which points are plotted in the sequential order of occurrence in the data table. By default, the points are connected by line. Set by default when the x-values are sorted with equal intervals or a single variable is a time series. To turn off connecting line segments for sorted, equal intervals data, set to FALSE. Customize the color of the line segments with segments_color with function style.

show_runs
: If TRUE, display the individual runs in the run analysis. Also, sets run to TRUE.

center_line
: Plots a dashed line through the middle of a run chart. Provides a center line for the "median" by default, when the values randomly vary about the mean. "mean" and "zero" specify that the center line goes through the mean or zero, respectively. Currently does not apply to Trellis plots.

`stat` Transform data for categorical variable `x` into a simple table of paired levels of `x` with numerical values of `y`. Apply specified aggregation such as `"mean"` to each of the levels of `x`, and, for a provided numerical variable `y`, transformations such as `"mean"`, `"sd"`, etc. The resulting dot plot is analogous to a bar chart.

`stat_x` If no `y` variable is specified, for constructing a frequency polygon, with access to the `bin_width` parameter. Either do the default `count` for each bin or the `proportion`, also indicated by `%`.

`vbs_plot` A character string that specifies the components of the **integrated Violin-Box-Scatterplot (VBS) of a continuous variable**. A `"v"` in the string indicates a violin plot, a `"b"` indicates a box plot with flagged outliers, and a `"s"` indicates a 1-variable scatterplot. Default value is `"vbs"`. The characters can be in any order and upper- or lower-case. Generalize to Trellis plots with the `by1` and `by2` parameters, but currently only applies to horizontal displays. Modify fill and border colors from the current `theme` with the `style` function parameters `violin_fill`, `violin_color`, `box_fill` and `box_color`.

`vbs_size` Width of the violin plot to the plot area. Make the violin (and also the accompanying box plot) larger or smaller by making the plot area and/or this value larger or smaller.

`bw` Bandwidth for the smoothness of the violin plot. Higher values for smoother plots. Default is to calculate a bandwidth that provides a relative smooth density plot.

`bw_iter` Number of iterations used to modify default R bandwidth to further smooth the obtained density estimate. When set, also displays the iterations and corresponding results.

`violin_fill` Fill color for a violin plot.

`box_fill` Fill color for a box plot.

`vbs_pt_fill` Points in a VBS scatterplot are black by default because the background is the violin, which is based on the current theme color. To use the values for `pt_fill` and `pt_color` specified by the `style` function, set to `"default"`. Or set to any desired color.

`vbs_mean` Show the mean on the box plot with a strip the color of `out_fill`, which can be changed with the `style` function.

`fences` If `TRUE`, draw the inner upper and lower fences as dotted line segments.

`k` IQR multiplier for the basis of calculating the distance of the whiskers of the box plot from the box. Default is Tukey's setting of 1.5.

`box_adj` Adjust the box and whiskers, and thus outlier detection, for skewness using the medcouple statistic as the robust measure of skewness according to Hubert and Vandervieren (2008).

`a, b` Scaling factors for the adjusted box plot to set the length of the whiskers. If explicitly set, activates `box_adj`.

`radius` Scaling factor of the bubbles in a **bubble plot**, which sets the radius of the largest displayed bubble in inches. To activate, either set the value of `size` to a third variable where the default is 0.10, or for categorical variables, either a factor or an integer variable with the number of unique values less than `n_cat`, the size of the bubbles represents frequency, with a default of 0.22.

`power` Relative size of the scaling of the bubbles to each other. Default value of 0.5 scales the bubbles so that the area of each bubble is the value of the corresponding sizing variable. Value of 1 scales so the radius of the bubble is the value of the sizing variable, increasing the discrepancy of size between the variables.

`low_fill` For a categorical variable and the resulting bubble plot, or a matrix of these plots, sets a color gradient of the fill color beginning with this color.

`hi_fill` For a categorical variable and the resulting bubble plot, or a matrix of these plots, sets a color gradient of the fill color ending with this color.

`smooth` **Smoothed density plot** for two numerical variables.

`smooth_points` Number of points superimposed on the density plot in the areas of the lowest density to help identify outliers, which controls how dark are the smoothed points.

`smooth_size` Size of points superimposed on the density plot.

`smooth_exp` Exponent of the function that maps the density scale to the color scale. Smaller than default of 0.25 yields darker plots.

`smooth_bins` Number of bins in both directions for the density estimation.

`n_bins` Specify the number of bins to bin a single numeric `x`-variable from which to compute the mean or median for a numeric `y`-variable for each bin of `x`. Points are plotted with the size dependent on the sample size for the bin, unless `size` is specified at a constant value. Default value is 1 for no binning. Available parameters include fill, color, transparency, segments, scale_x, and scale_y.

`bin` If `TRUE`, display the default frequency distribution for the text output of the Violin-Box-Scatter (VBS) Plot, or, if `values` is set to `"count"`, a frequency polygon.

`bin_start` Optional specified starting value of the bins for a **frequency polygon or for the text output of a Violin-Box-Scatter (VBS) Plot**. Also, sets `bin` to `TRUE`.

`bin_width` Optional specified bin width value. Also, sets `bin` to `TRUE`.

`bin_end` Optional specified value that is within the last bin, so the actual endpoint of the last bin may be larger than the specified value.

`breaks` The method for calculating the bins, or an explicit specification of the bins, such as with the standard R `seq` function or other options provided by the `hist` function. Also, sets `bin` to `TRUE`.

`cumulate` Specify a cumulative frequency polygon.

`xlab, ylab` **Axis label** for `x`-axis or `y`-axis_ If not specified, then the label becomes the name of the corresponding variable label if it exists, or, if not, the variable name. If `xy_ticks` is `FALSE`, no `ylab` is displayed. Customize these and related parameters with parameters such as `lab_color` from the `style` function.

`main`	Label for the title of the graph. If the corresponding variable labels exist, then the title is set by default from the corresponding variable labels.
`sub`	Sub-title of graph, below `xlab`. Not yet implemented.
`lab_adjust`	Two-element vector – x-axis label, y-axis label – adjusts the position of the axis labels in approximate inches. + values move the labels away from plot edge. Not applicable to Trellis graphics.
`margin_adjust`	Four-element vector – top, right, bottom and left – adjusts the margins of the plotted figure in approximate inches. + values move the corresponding margin away from plot edge. Not applicable to Trellis graphics.
`rotate_x`	**Rotation in degrees of the value labels** on the `x`-axis, usually to accommodate longer values, typically used in conjunction with `offset`. When equal 90 the value labels are perpendicular to the x-axis and a different algorithm places the labels so that `offset` is not needed.
`rotate_y`	Degrees that the axis values for the value labels on the `y`-axis are rotated, usually to accommodate longer values, typically used in conjunction with `offset`.
`offset`	The amount of spacing between the axis values and the axis. Default is 0.5. Larger values such as 1.0 are used to create space for the label when longer axis value names are rotated.
`xy_ticks`	Flag that indicates if tick marks and associated **value labels** on the axes are to be displayed. To rotate the axis values, use `rotate_x`, `rotate_y`, and `offset` from the `style` function.
`origin_x`	Origin of `x`-axis. Starting value of `x`, by default the minimum value of `x`, except when `stat` is set to `"count"` or related where the origin is zero by default, but can be modified.
`scale_x`	If specified, a vector of three values that define the x-axis with numerical values: starting value, ending value, and number of intervals.
`scale_y`	If specified, a vector of three values that define the y-axis with numerical values: starting value, ending value, and number of intervals.
`pad_x`	Proportion of padding added to left and right sides of the `x`-axis, respectively. Value from 0 to 1 for each of the two elements. If only one element specified, value is applied to both sides.
`pad_y`	Proportion of padding added to bottom and top sides of the `y`-axis, respectively. Value from 0 to 1 for each of the two elements. If only one element specified, value is applied to both sides.
`legend_title`	Title of the legend for a multiple-variable `x` or `y` plot.
`add`	**Overlay one or more objects**, text or a geometric figures, on the plot. Possible values are any text to be written, the first argument, which is `"text"`, or, `"labels"` to label each point with the row name, or, `"rect"` (rectangle), `"line"`, `"arrow"`, `"v_line"` (vertical line), and `"h_line"` (horizontal line). The value `"means"` is short-hand for vertical and horizontal lines at the respective means. Does not apply to Trellis graphics. Customize with parameters such as `add_fill` and `add_color` from the `style` function.

`x1`	First x-coordinate to be considered for each object, can be `"mean_x"`. Not used for `"h_line"`.
`y1`	First y-coordinate to be considered for each object, can be `"mean_y"`. Not used for`"v_line"`.
`x2`	Second x-coordinate to be considered for each object, can be `"mean_x"`. Only used for `"rect"`, `"line"` and `arrow`.
`y2`	Second y-coordinate to be considered for each object, can be `"mean_y"`. Only used for `"rect"`, `"line"` and `arrow`.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`do_plot`	If `TRUE`, the default, then generate the plot.
`pdf_file`	Indicate to direct pdf graphics to the specified name of the pdf file.
`width`	Width of the plot window in inches, defaults to 5 except in RStudio to maintain an approximate square plotting area.
`height`	Height of the plot window in inches, defaults to 4.5 except for 1-D scatterplots and when in RStudio.
`digits_d`	Number of significant digits for each of the displayed summary statistics.
`n_cat`	Number of categories, specifies the largest number of unique, equally spaced integer values of a variable for which the variable will be analyzed as categorical instead of continuous. Default is 0. Use to specify that such variables are to be analyzed as categorical, a kind of informal R factor. *[deprecated]*: Best to convert a categorical integer variable to a factor.
`value_labels`	For factors, default is the factor labels, and for character variables, default is the character values. Or, provide labels for the `x`-axis on the graph to override these values. If the variable is a factor and `value_labels` is not specified (is `NULL`), then the value_labels are set to the factor levels with each space replaced by a new line character. If `x` and `y`-axes have the same scale, they also apply to the `y`-axis. Control the plotted size with `axis_cex` and `axis_x_cex` from the lessR `style function`. *[deprecated]*: Better to convert a categorical integer variable to a factor.
`rows`	Deprecated old parameter name that is now called `filter`.
`eval_df`	Determines if to check for existing data frame and specified variables. By default is `TRUE` unless the `shiny` package is loaded then set to `FALSE` so that Shiny will run. Needs to be set to `FALSE` if using the pipe `%\>%` notation.
`fun_call`	Function call. Used with `knitr` to pass the function call when obtained from the abbreviated function call `sp`.
`...`	Other parameter values for non-Trellis graphics as defined by and processed by standard R functions `plot` and `par`, including `cex.main` for the size of the title `col.main` for the color of the title `sub` and `col.sub` for a subtitle and its color

Details

VARIABLES and TRELLIS PLOTS
There is at least one primary variable, x, which defines the coordinate system for plotting in terms of the x-axis, the horizontal axis. Plots may also specify a second primary variable, y, which defines the y-axis of the coordinate system. One of these primary variables may be a vector. The simplest plot is from the specification of only one or two primary variables, each as a single variable, which generates a single scatterplot of either one or two variables, necessarily on a single plot, called a panel, defined by a single x-axis and usually a single y-axis_

For numeric primary variables, a single panel may also contain multiple plots of two types. Form the first type from subsets of observations (rows of data) based on values of a categorical variable. Specify this plot with the by parameter, which identifies the grouping variable to generate a scatterplot of the primary variables for each of its levels. The points for each group are plotted with a different shape and/or color. By default, the colors vary, though to maintain the color scheme, if there are only two levels of the grouping variable, the points for one level are filled with the current theme color and the points for the second level are plotted with transparent interiors.

Or, obtain multiple scatterplots on the same panel with multiple numeric x-variables, or multiple y-variables. To obtain this graph, specify one of the primary variables as a vector of multiple variables.

Trellis graphics (facets), from Deepayan Sarkar's (2009) lattice package, may be implemented in which multiple panels for one numeric x-variable and one numeric y-variable are displayed according to the levels of one or two categorical variables, called conditioning variables. A variable specified with by is a conditioning variable that results in a Trellis plot, the scatterplot of x and y produced at *each* level of the by1 variable. The inclusion of a second conditioning variable, by2, results in a separate scatterplot panel for *each* combination of cross-classified values of both by1 and by2. A grouping variable according to by may also be specified, which is then applied to each panel. If there are 1000 or less unique values of x, an analysis of the maximum number of repetitions for each value of by1 is provided.

Control the panel dimensions and the overall size of the Trellis plot with the following parameters: width and height for the physical dimensions of the plot window, n_row and n_col for the number of rows and columns of panels, and aspect for the ratio of the height to the width of each panel. The plot window is the standard graphics window that displays on the screen, or it can be specified as a pdf file with the pdf_file parameter.

CATEGORICAL VARIABLES
Conceptually, there are continuous variables and categorical variables. Categorical variables have relatively few unique data values. However, categorical variables can be defined with non-numeric values, but also with numeric values, such as responses to a five-point Likert scale from Strongly Disagree to Strongly Agree, with responses coded 1 to 5. The three by –variables – by1, by2 and by – only apply to graphs created with numeric x and/or y variables, continuous or categorical.

The standard and most general way to define a categorical variable is as an R factor, such as created with the lessR factors function. lessR provides the option to define an integer variable with equally spaced values as categorical based on the value of n_cat, which can be set locally or globally with the style function. For example, for a variable with data values from 5-point Likert scale, a value of n_cat of 5 will define the define the variable as categorical. The default value is 0. To explicitly analyze the values as categorical, set n_cat to a value larger than 0, at least the size of the number of unique integer values. Can also annotate a graph of the values of an integer categorical variable with value_labels option.

A scatterplot of Likert type data is problematic because there are so few possibilities for points in the scatterplot. For example, for a scatterplot of two five-point Likert response data, there are only 26 possible paired values to plot, so most of the plotted points overlap with others. In this situation, that is, when a single variable or two variables with Likert response scales are specified, a bubble plot is automatically provided, with the size of each point relative to the joint frequency of the paired data values. To request a sunflower plot in lieu of the bubble plot, set the `shape` to `"sunflower"`.

DATA
The default input data frame is `d`. Specify another name with the `data` option. Regardless of its name, the data frame need not be attached to reference the variables directly by its name, that is, no need to invoke the `d$name` notation. The referenced variables can be in the data frame and/or the user's workspace, the global environment.

The data values themselves can be plotted, or for a single variable, counts or proportions can be plotted on the `y`-axis. For a categorical `x`-variable paired with a continuous variable, means and other statistics can be plotted at each level of the `x`-variable. If `x` is continuous, it is binned first, with the standard `Histogram` binning parameters available, such as `bin_width`, to override default values. The `stat` parameter sets the values to plot, with `data` the default. By default, the connecting line segments are provided, so a frequency polygon results. Turn off the lines by setting `lwd=0`.

The `rows` parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in `Logic` such as `&` for and, `|` for or and `!` for not, and use the standard R relational operators as described in `Comparison` such as `==` for logical equality `!=` for not equals, and `>` for greater than. See the Examples.

VALUE LABELS
[DEPRECATED. Use `factor()` instead.] The value labels for each axis can be over-ridden from their values in the data to user supplied values with the `value_labels` option. This option is particularly useful for Likert-style data coded as integers. Then, for example, a 0 in the data can be mapped into a "Strongly Disagree" on the plot. These value labels apply to integer categorical variables, and also to factor variables. To enhance the readability of the labels on the graph, any blanks in a value label translate into a new line in the resulting plot. Blanks are also transformed as such for the labels of factor variables.

However, the lessR function `factors` allows for the easy creation of factors, one variable or a vector of variables, in a single statement, and is generally recommended as the method for providing value labels for the variables.

VARIABLE LABELS
Although standard R does not provide for variable labels, `lessR` can store the labels in the data frame with the data, obtained from the `Read` function or `VariableLabels`. If variable labels exist, then the corresponding variable label is by default listed as the label for the corresponding axis and on the text output.

ONE VARIABLE PLOT
The one variable plot of one continuous variable generates either a violin/box/scatterplot (VBS plot), or a run chart with `run=TRUE`, or `x` can be an R time series variable for a time series chart. For the box plot, for gray scale output potential outliers are plotted with squares and outliers are plotted with diamonds, otherwise shades of red are used to highlight outliers. The default definition of outliers is based on the standard boxplot rule of values more than 1.5 IQR's from the box. The definition of outliers may be adjusted (Hubert and Vandervieren, 2008), such that the whiskers are computed from the medcouple index of skewness (Brys, Hubert, & Struyf, 2004).

The plot can also be obtained as a bubble plot of frequencies for a categorical variable.

TWO VARIABLE PLOT
When two variables are specified to plot, by default if the values of the first variable, x, are unsorted, or if there are unequal intervals between adjacent values, or if there is missing data for either variable, a scatterplot is produced from a call to the standard R plot function. By default, sorted values with equal intervals between adjacent values of the first of the two specified variables yields a function plot if there is no missing data for either variable, that is, a call to the standard R plot function with type="l", which connects each adjacent pair of points with a line segment.

Specifying multiple, continuous x-variables against a single y variable, or vice versa, results in multiple plots on the same graph. The color of the points of the second variable is the same as that of the first variable, but with a transparent fill. For more than two x-variables, multiple colors are displayed, one for each x-variable.

BUBBLE PLOT FREQUENCY MATRIX (BPFM)
Multiple categorical variables for x may be specified in the absence of a y variable. (A categorical variable is either a factor variable or an integer variable with n_cat set at least at the number of unique values.) A bubble plot results that illustrates the frequency of each response for each of the variables in a common figure in which the x-axis contains all of the unique labels for all of the variables plotted. Each line of information, the bubbles and counts for a single variable, replaces the standard bar chart in a more compact display. Usually the most meaningful when each variable in the matrix has the same response categories, that is, levels, such as for a set of shared Likert scales. The BPFM is considerably condensed presentation of frequencies for a set of variables than are the corresponding bar charts.

SCATTERPLOT MATRIX
A single vector of continuous variables specified as x, with no y-variable, generates a scatterplot matrix of the specified variable. A continuous variable is defined as a numeric variable with more than n_cat unique responses. To force an item with a small number of unique responses, such as from a 5-pt Likert scale, to be treated as continuous, set n_cat to a number lower than 5, such as n_cat=0 in the function call.

The scatterplot matrix is displayed according to the current color theme. Specific colors such as fill, color, etc. can also be provided. The upper triangle shows the correlation coefficient, and the lower triangle each corresponding scatterplot, with, by default, the non-linear loess best fit line. The code fit option can be used to provide the linear least squares line instead, along with the corresponding fit_color for the color of the fit line.

SIZE VARIABLE
A variable specified with size= is a numerical variable that activates a bubble plot in which the size of each bubble is determined by the value of the corresponding value of size, which can be a variable or a constant.

To explicitly vary the shapes, use shape and a list of shape values in the standard R form with the c function to combine a list of values, one specified shape for each group, as shown in the examples. To explicitly vary the colors, use fill, such as with R standard color names. If fill is specified without shape, then colors are varied, but not shapes. To vary both shapes and colors, specify values for both options, always with one shape or color specified for each level of the by variable.

Shapes beyond the standard list of named shapes, such as "circle", are also available as single characters. Any single letter, uppercase or lowercase, any single digit, and the characters "+", "*" and "#" are available, as illustrated in the examples. In the use of shape, either use standard named shapes, or individual characters, but not both in a single specification.

SCATTERPLOT ELLIPSE

For a scatterplot of two numeric variables, the `ellipse=TRUE` option draws the .95 data ellipse as computed by the `ellipse` function, written by Duncan Murdoch and E. D. Chow, from the `ellipse` package. The axes are automatically lengthened to provide space for the entire ellipse that extends beyond the maximum and minimum data values. The specific level of the ellipse can be specified with a numerical value in the form of a proportion. Multiple numerical values of `ellipse` may also be specified to obtain multiple ellipses.

BOXPLOTS
For a single variable the preferred plot is the integrated violin/box/scatter plot or VBS plot. Only the violin or box plot can be obtained with the corresponding aliases `ViolinPlot` and `BoxPlot`, or by setting `vbs_plot` to `"v"` or `"b"`. To view a box plot of a continuous variable (Y) across the levels of a categorical variable (X), either as part of the full VBS plot, or by itself, there are two possibilities:
1. Plot(Y,X) or BoxPlot(Y, X)
2. Plot(Y, by1=X) or BoxPlot(Y, by1=X)
Both styles produce the same information. What differs is the color scheme.

The first possibility places the multiple box plots on a single pane and also, for the default color scheme `"colors"`, displays the sequence of box plots with the default qualitative color palette from the lessR function `getColors`. All colors are displayed at the same level of gray-scale saturation and brightness to avoid perceptual bias. `BarChart` and `PieChart` use the same default colors as well.

The second possibility with `by1` produces the different box plots on a separate panel, that is, a Trellis chart. These box plots are displayed with a single hue, the first color, blue, in the default qualitative sequence.

TIME CHARTS
Specifying one or more `x`-variables with no `y`-variables, and `run=TRUE` plots the `x`-variables in a run chart. The values of the specified `x`-variable are plotted on the `y`-axis, with Index on the `x`-axis. Index is the ordinal position of each data value, from 1 to the number of values.

If the specified `x`-variable is of type `Date`, or is an R time series, a time series plot is generated for each specified variable. If a formal R time-series, univariate or multivariate, specify as the `x`-variable. Or, specify the `x`-variable of type `Date`, and then specify the `y`-variable as one or more time series to plot. The `y`-variable can be formatted as tidy data with all the values in a single column, or as wide-formatted data with the time-series variables in separate columns.

The parameter `time_unit` aggregates the date variable according to its specified value. The aggregation is based on two functions from the `xts` package, `endpoints()` and `period.apply()`. For example, a data variable has daily values but is plotted with aggregated quarterly values. From the `endpoints()` documentation: Valid values include: `"us"` (microseconds), `"microseconds"`, `"ms"` (milliseconds), `"milliseconds"`, `"secs"` (seconds), `"seconds"`, `"mins"` (minutes), `"minutes"`, `"hours"`, `"days"`, `"weeks"`, `"months"`, `"quarters"`, and `"years"`.

Specify the function by which to aggregate with the parameter `time_agg`. The default is `"sum"`.

2-D KERNEL DENSITY
With `smooth=TRUE`, the R function `smoothScatter` is invoked according to the current color theme. Useful for very large data sets. The `smooth_points` parameter plots points from the regions of the lowest density. The `smooth_bins` parameter specifies the number of bins in both directions for the density estimation. The `smooth_exp` parameter specifies the exponent in the function that maps the density scale to the color scale to allow customization of the intensity of the plotted gradient colors. Higher values result in less color saturation, de-emphasizing points from regions of lessor

density. These parameters are respectively passed directly to the `smoothScatter` `nrpoints`, `nbin` and `transformation` parameters. Grid lines are turned off, by default, but can be displayed by setting the `grid_color` parameter.

COLORS
A color theme for all the colors can be chosen for a specific plot with the `colors` option with the `lessR` function `style`. The default color theme is `"lightbronze"`. A gray scale is available with `"gray"`, and other themes are available as explained in `style`, such as `"sienna"` and `"darkred"`. Use the option `style(sub_theme="black")` for a black background and partial transparency of plotted colors.

Colors can also be changed for individual aspects of a scatterplot as well with the `style` function. To provide a warmer tone by slightly enhancing red, try a background color such as `panel_fill="snow"`. Obtain a very light gray with `panel_fill="gray99"`. To darken the background gray, try `panel_fill="gray97"` or lower numbers. See the `lessR` function `showColors`, which provides an example of all available named R colors with their RGB values_

For the color options, such as `violin_color`, the value of `"off"` is the same as `"transparent"`.

ANNOTATIONS
Use the `add` and related parameters to annotate the plot with text and/or geometric figures. Each object is placed according from one to four corresponding coordinates, the required coordinates to plot that object, as shown in the following table. `x`-coordinates may have the value of `"mean_x"` and `y`-coordinates may have the value of `"mean_y"`.

Value	Object	Required Coordinates
`"text"`	text	x1, y1
`"point"`	text	x1, y1
`"rect"`	rectangle	x1, y1, x2, y2
`"line"`	line segment	x1, y1, x2, y2
`"arrow"`	arrow	x1, y1, x2, y2
`"v_line"`	vertical line	x1
`"h_line"`	horizontal line	y1
`"means"`	horiz, vert lines	

The value of `add` specifies the object. For a single object, enter a single value. Then specify the value of the needed corresponding coordinates, as specified in the above table. For multiple placements of that object, specify vectors of corresponding coordinates. To annotate multiple objects, specify multiple values for `add` as a vector. Then list the corresponding coordinates, for up to each of four coordinates, in the order of the objects listed in `add`.

Can also specify vectors of different properties, such as `add_color`. That is, different objects can be different colors, different transparency levels, etc.

PDF OUTPUT
To obtain pdf output, use the `pdf_file` option, perhaps with the optional `width` and `height` options. These files are written to the default working directory, which can be explicitly specified with the R `setwd` function.

ADDITIONAL OPTIONS
Commonly used graphical parameters that are available to the standard R function `plot` are also generally available to `Plot`, such as:

cex.main, col.lab, font.sub, etc. Settings for main- and sub-title and axis annotation, see `title` and `par`.

main Title of the graph, see `title`.

xlim The limits of the plot on the x-axis, expressed as c(x1,x2), where `x1` and `x2` are the limits. Note that `x1` > `x2` is allowed and leads to a reversed axis.

ylim The limits of the plot on the y-axis.

ONLY VARIABLES ARE REFERENCED
A referenced variable in a `lessR` function can only be a variable name. This referenced variable must exist in either the referenced data frame, such as the default `d`, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

`> Plot(rnorm(50), rnorm(50)) # does NOT work`

Instead, do the following:

```
> X <- rnorm(50)   # create vector X in user workspace
> Y <- rnorm(50)   # create vector Y in user workspace
> Plot(X,Y)     # directly reference X and Y
```

Value

The output can optionally be saved into an `R` object, otherwise it simply appears in the console. The output here is just for the outlier analysis of the two-variable scatterplot with continuous variables. The outlier identification must be activated for the analysis, such as from parameter `MD_cut`.

READABLE OUTPUT
`out_stats`: Correlational analysis.
`out_outliers`: Mahalanobis Distance of each outlier.

STATISTICS
`outliers`: Row numbers that contain the outliers.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Brys, G., Hubert, M., & Struyf, A. (2004). A robust measure of skewness. Journal of Computational and Graphical Statistics, 13(4), 996-1017.

Murdoch, D, and Chow, E. D. (2013). `ellipse` function from the `ellipse` package package.

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapter 8, NY: Routledge.

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 5, NY: CRC Press.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Hubert, M. and Vandervieren, E. (2008). An adjusted boxplot for skewed distributions, Computational Statistics and Data Analysis 52, 51865201.

Sarkar, Deepayan (2008) Lattice: Multivariate Data Visualization with R, Springer. http://lmdvr.r-forge.r-project.org/

See Also

plot, stripchart, title, par, loess, Correlation, style.

Examples

```
# read the data
d <- rd("Employee", quiet=TRUE)
d <- d[.(random(0.6)),]  # less computationally intensive
dd=d

#---------------------------------------------------
# traditional scatterplot with two numeric variables
#---------------------------------------------------

Plot(Years, Salary, by=Gender, size=2, fit="lm",
     fill=c("olivedrab3", "gold1"),
     color=c("darkgreen", "gold4"))

# scatterplot with all defaults
Plot(Years, Salary)
# or use abbreviation sp in place of Plot
# or use full expression ScatterPlot in place of Plot

# maximum information, minimum input: scatterplot +
#  means, outliers, ellipse, least-squares lines with and w/o outliers
Plot(Years, Salary, enhance=TRUE)

# extend x and y axes
Plot(Years, Salary, scale_x=c(-10, 35, 10), scale_y=c(0,200000,10))

Plot(Years, Salary, add="Hi", x1=c(12, 16, 18), y1=c(80000, 100000, 60000))

Plot(Salary, row_names)

d <- factors(Gender, levels=c("M", "F"))
Plot(Years, Salary, by1=Gender)
d <- dd



# just males employed more than 5 years
Plot(Years, Salary, filter=(Gender=="M" & Years > 5))
```

```
# plot 0.95 data ellipse with the points identified that represent
#   outliers defined by a Mahalanobis Distance larger than 6
# save outliers into R object out
d[1, "Salary"] <- 200000
out <- Plot(Years, Salary, ellipse=0.95, MD_cut=6)

# new shape and point size, no grid or background color
# then put style back to default
style(panel_fill="powderblue", grid_color="off")
Plot(Years, Salary, size=2, shape="diamond")
style()

# translucent data ellipses without points or edges
#  show the idealized joint distribution for bivariate normality
style(ellipse_color="off")
Plot(Years, Salary, size=0, ellipse=seq(.1,.9,.10))
style()

# bubble plot with size determined by the value of Pre
# display the value for the bubbles with values of  min, median and max
Plot(Years, Salary, size=Pre, size_cut=3)

# variables in a data frame not the default d
# plot 0.6 and 0.9 data ellipses with partially transparent points
# change color theme to gold with black background
style("gold", sub_theme="black")
Plot(eruptions, waiting, transparency=.5, ellipse=seq(.6,.9), data=faithful)

# scatterplot with two x-variables, plotted against Salary
# define a new style, then back to default
style(window_fill=rgb(247,242,230, maxColorValue=255),
  panel_fill="off", panel_color="off", pt_fill="black", transparency=0,
  lab_color="black", axis_text_color="black",
  axis_y_color="off", grid_x_color="off", grid_y_color="black",
  grid_lty="dotted", grid_lwd=1)
Plot(c(Pre, Post), Salary)
style()

# increase span (smoothing) from default of .7 to 1.25
# span is a loess parameter, which generates a caution that can be
#   ignored that it is not a graphical parameter -- we know that
# display confidence intervals about best-fit line at
#   0.95 confidence level
Plot(Years, Salary, fit="loess", span=1.25)

# 2-D kernel density (more useful for larger sample sizes)
Plot(Years, Salary, smooth=TRUE)


#------------------------------------------------------
# scatterplot matrix from a vector of numeric variables
#------------------------------------------------------
```

```
# with least squares fit line
Plot(c(Salary, Years, Pre), fit="lm")


#---------------------------------------------------------------
# Trellis graphics and by for groups with two numeric variables
#---------------------------------------------------------------

# Trellis plot with condition on 1-variable
# optionally re-order default alphabetical R ordering by converting
#   to a factor with lessR factors (which also does multiple variables)
# always save to the full data frame with factors
d <- factors(Gender, levels=c("M", "W"))
Plot(Years, Salary, by1=Gender)
d <- Read("Employee", quiet=TRUE)



# two Trellis classification variables with a single continuous
Plot(Salary, by1=Dept, by2=Gender)

# all three by (categorical) variables
Plot(Years, Salary, by1=Dept, by2=Gender, by=Plan)

# vary both shape and color with a least-squares fit line for each group
style(color=c("darkgreen", "brown"))
Plot(Years, Salary, by1=Gender, fit="lm", shape=c("F","M"), size=.8)
style("gray")

# compare the men and women Salary according to Years worked
#   with an ellipse for each group
Plot(Years, Salary, by=Gender, ellipse=.50)


#--------------------------------------------------
# analysis of a single numeric variable (or vector)
#--------------------------------------------------

# One continuous variable
# ----------------------
# integrated Violin/Box/Scatterplot, a VBS plot
Plot(Salary)

Plot(Years, Salary, by=Gender, size=2, fit="lm",
     fill=c("olivedrab3", "gold1"),
     color=c("darkgreen", "gold4"))



# by variable, different colors for different values of the variable
# two panels
Plot(Salary, by1=Dept)
```

```
# large sample size
x <- rnorm(10000)
Plot(x)

# custom colors for outliers, which might not appear in this subset data
style(out_fill="hotpink", out2_fill="purple")
Plot(Salary)
style()

# no violin plot or scatterplot, just a boxplot
Plot(Salary, vbs_plot="b")
# or, the same with the mnemonic
BoxPlot(Salary)

# two related displays of box plots for different levels of a
#   categorical variable
BoxPlot(Salary, by1=Dept)


# binned values to plot counts
# ----------------------------
# bin the values of Salary to plot counts as a frequency polygon
# the counts are plotted as points instead of the data
Plot(Salary, stat_x="count")  # bin the values

# time charts
#------------
# run chart, with default fill area
Plot(Salary, run=TRUE, area_fill="on")

# two run charts in same plot
# or could do a multivariate time series
Plot(c(Pre, Post), run=TRUE)

# Trellis graphics run chart with custom line width, no points
Plot(Salary, run=TRUE, by1=Gender, lwd=3, size=0)

# daily time series plot
# create the daily time series from R built-in data set airquality
oz.ts <- ts(airquality$Ozone, start=c(1973, 121), frequency=365)
Plot(oz.ts)

# multiple time series plotted from dates and stacked
# black background with translucent areas, then reset theme to default
style(sub_theme="black", color="steelblue2", transparency=.55,
  window_fill="gray10", grid_color="gray25")
date <- seq(as.Date("2013/1/1"), as.Date("2016/1/1"), by="quarter")
x1 <- rnorm(13, 100, 15)
x2 <- rnorm(13, 100, 15)
x3 <- rnorm(13, 100, 15)
df <- data.frame(date, x1, x2, x3)
rm(date); rm(x1); rm(x2); rm(x3)
```

```
Plot(date, x1:x3, data=df)
style()

# aggregate monthly data to plot by quarter
n.q <- 42
month <- seq(as.Date("2013/1/1"), length=n.q, by="months")
x <- rnorm(n.q, 100, 15)
Plot(month, x, time_unit="quarters")


# trigger a time series with a Date variable specified first
# stock prices for three companies by month:  Apple, IBM, Intel
d <- rd("StockPrice")
# only plot Apple
Plot(Month, Price, filter=(Company=="Apple"))
# Trellis plots, one for each company
Plot(Month, Price, by1=Company, n_col=1)
# all three plots on the same panel, three shades of blue
Plot(Month, Price, by=Company, color="blues")


#-------------------------------------------
# analysis of a single categorical variable
#-------------------------------------------
d <- rd("Employee")

# default 1-D bubble plot
# frequency plot, replaces bar chart
Plot(Dept)



# plot of frequencies for each category (level), replaces bar chart
Plot(Dept, stat_x="count")


#-----------------------------------------------------
# scatterplot of numeric against categorical variable
#-----------------------------------------------------

# generate a chart with the plotted mean of each level
# rotate x-axis labels and then offset from the axis
style(rotate_x=45, offset=1)
Plot(Dept, Salary)
style()


#-------------------
# Cleveland dot plot
#-------------------

# row.names on the y-axis
Plot(Salary, row_names)
```

```
# standard scatterplot
Plot(Salary, row_names, sort_yx="0", segments_y=FALSE)

# Cleveland dot plot with two x-variables
Plot(c(Pre, Post), row_names)


#------------
# annotations
#------------

# add text at the one location specified by x1 and x2
Plot(Years, Salary, add="Hi There", x1=12, y1=80000)
# add text at three different specified locations
Plot(Years, Salary, add="Hi", x1=c(12, 16, 18), y1=c(80000, 100000, 60000))

# add three different text blocks at three different specified locations
Plot(Years, Salary, add=c("Hi", "Bye", "Wow"), x1=c(12, 16, 18),
  y1=c(80000, 100000, 60000))

# add an 0.95 data ellipse and horizontal and vertical lines through the
#  respective means
Plot(Years, Salary, ellipse=0.95, add=c("v_line", "h_line"),
  x1="mean_x", y1="mean_y")
# can be done also with the following short-hand
Plot(Years, Salary, ellipse=0.95, add="means")

# a rectangle requires two points, four coordinates, <x1,y1> and <x2,y2>
style(add_trans=.8, add_fill="gold", add_color="gold4", add_lwd=0.5)
Plot(Years, Salary, add="rect", x1=12, y1=80000, x2=16, y2=115000)

# the first object, a rectangle, requires all four coordinates
# the vertical line at x=2 requires only an x1 coordinate, listed 2nd
Plot(Years, Salary, add=c("rect", "v_line"), x1=c(10, 2),
  y1=80000, x2=12, y2=115000)

# two different rectangles with different locations, fill colors and translucence
style(add_fill=c("gold3", "green"), add_trans=c(.8,.4))
Plot(Years, Salary, add=c("rect", "rect"),
  x1=c(10, 2), y1=c(60000, 45000), x2=c(12, 75000), y2=c(80000, 55000))


#---------------------------------------------------
# analysis of two categorical variables (Likert data)
#---------------------------------------------------

d <- rd("Mach4", quiet=TRUE)  # Likert data, 0 to 5

# size of each plotted point (bubble) depends on its joint frequency
# triggered by default when replication of joint values and
#   less than 9 unique data values for each
# n_cat=6 means treat responses as categorical for up to 6 equally-spaced
```

```
#   integer values
Plot(m06, m07, n_cat=6)

# use value labels for the integer values, modify color options
LikertCats <- c("Strongly Disagree", "Disagree", "Slightly Disagree",
   "Slightly Agree", "Agree", "Strongly Agree")
style(fill="powderblue", color="blue", bubble_text="darkred")
d <- factors(m01:m20, 0:5, labels=LikertCats)
Plot(m01:m10)
style()  # reset theme



# get correlation analysis instead of cross-tab analysis
# rely upon the default value of n_cat=0 so that integer
#   valued variables are analyzed as numerical
Plot(m06, m07)

#-----------------------------
# Bubble Plot Frequency Matrix
#-----------------------------

#---------------
# function curve
#---------------

x <- seq(10,50,by=2)
y1 <- sqrt(x)
y2 <- x**.33
# x is sorted with equal intervals so run chart by default
Plot(x, y1)

# multiple plots from variable vectors need to have the variables
#  in a data frame
d <- data.frame(x, y1, y2)
# if variables are in the user workspace and in a data frame
#   with the same names, the user workspace versions are used,
#   which do not work with vectors of variables, so remove
rm(x); rm(y1); rm(y2)
Plot(x, c(y1, y2))


#-----------
# modern art
#-----------
clr <- colors()  # get list of color names
color0 <- clr[sample(1:length(clr), size=1)]
clr <- clr[-(153:353)]  # get rid of most of the grays

n <- sample(5:30, size=1)
x <- rnorm(n)
y <- rnorm(n)
color1 <- clr[sample(1:length(clr), size=1)]
```

```
color2 <- clr[sample(1:length(clr), size=1)]

style(window_fill=color0, color=color2)
Plot(x, y, run=TRUE, area_fill="on",
  xy_ticks=FALSE, main="Modern Art", xlab="", ylab="",
  cex.main=2, col.main="lightsteelblue", n_cat=0, center_line="off")
style() # reset style to default
```

`print.out` *Display a Portion of Output from a Saved List Object*

Description

Displays the portions of saved results of an analysis from a `lessR` function into an object, such as for later display at the console or to be integrated into a `Rmd` analysis, for example from `RStudio`. This function is usually implicitly accessed by the user simply by entering the name of an output piece into the console or in a `Rmd` file, such as, such as `r$out_coefs` that results from `r` in `r <- reg(Y ~ X)`.

Now just applies to the `lessR` `Regression` function.

Usage

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

Arguments

`x`	The piece of output to display, a character vector or a list of character vectors.
`...`	Other parameter values.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapters 9 and 10, NY: Routledge.

See Also

`Regression`

Examples

```
# read internal data set
d <- rd("Employee", quiet=TRUE)
# do the summary statistics
s <- ss_brief(Salary)
# print the piece of output, print function is implicit
s$outliers
```

`print.out_all` *Display All Text Output from a Saved List Object*

Description

Displays all the results saved as an `R` list into an object from a `lessR` analysis. An example of a saved object is `r` in `r <- reg(Y ~ X)`. The results are displayed at the console or integrated into a `knitr` analysis, for example from `RStudio`. This function is usually implicitly accessed by the user simply by entering the name of the saved object at the console or in a `knitr` file.

Usage

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

Arguments

`x`	The list of components to display.
`...`	Other parameter values.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

`Regression`

Examples

```
# read internal data set
d <- rd("Employee", quiet=TRUE)
# do the summary statistics
s <- ss_brief(Salary)
# display all the output, print function is implicit
s
```

prob_norm *Compute and Plot Normal Curve Probabilities over an Interval*

Description

Calculate the probability of an interval for a normal distribution with specified mean and standard deviation, providing both the numerical probability and a plot of the interval with the corresponding normal curve.

Usage

```
prob_norm(lo=NULL, hi=NULL, mu=0, sigma=1, nrm_color="black",
         fill_nrm="grey91", fill_int="slategray3",
         ylab="", y_axis=FALSE, z=TRUE, axis_size=.9,
         pdf_file=NULL, width=5, height=5, ...)
```

Arguments

`lo`	Lowest value in the interval for which to compute probability.
`hi`	Highest value in the interval for which to compute probability.
`mu`	Population mean of normal distribution.
`sigma`	Population standard deviation of normal distribution.
`nrm_color`	Color of the border of the normal curve.
`fill_nrm`	Fill color of the normal curve.
`fill_int`	Fill color of the interval for which the probability is computed.
`ylab`	Label for the optional vertical axis_
`y_axis`	If `TRUE`, then a vertical axis is included.
`z`	If `TRUE`, then include z-values on the horizontal-axis_ Set to `FALSE` if `mu=0` and `sigma=1`.
`axis_size`	Magnification factor for the axis labels, the value of `axis_cex`.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values for graphics.

Details

Calculate the normal curve probability for the specified interval and normal curve. If there is no upper value of the interval provided, `hi`, then the upper tail probability is provided, that is, from the specified value until positive infinity. If there is no lower value, `lo`, then the lower tail probability is provided. The probability is calculated with `pnorm`.

Value

prob: Calculated probability.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

pnorm, plot.

Examples

```
# Mu=0, Sigma=1: Standard normal prob, values between 0 and 2
prob_norm(0,2)

# Mu=0, Sigma=1: Standard normal prob, values lower than 2
prob_norm(hi=2)

# Mu=0, Sigma=1: Standard normal prob, values larger than 2
prob_norm(lo=2)

# Mu=100, Sigma=15: Change default fill color of plotted interval
prob_norm(lo=115, hi=125, mu=100, sigma=15, fill_int="plum")
```

---

prob_tcut *Plot t-distribution Curve and Specified Cutoffs with Normal Curve*

Description

Plot a specified t-distribution against the standardized normal curve with the corresponding upper and lower tail cutoffs.

Usage

```
prob_tcut(df, alpha=0.05, digits_d=3, y_axis=FALSE,
          fill="aliceblue", color_tail="palevioletred4",
          nrm_color=gray(.7), color_t=gray(.08),
          pdf_file=NULL, width=5, height=5, ...)
```

Arguments

df	Degrees of freedom for t-distribution, must be 2 or larger.
alpha	Alpha to define the tail cutoff area.
digits_d	Number of decimal digits in the output.
y_axis	If FALSE, then the y axis is not displayed.
fill	Fill color for the interior of the t-distribution curve.

`color_tail`	Color of the tail areas of the t-distribution.
`nrm_color`	Color of the normal curve.
`color_t`	Color of the t-distribution curve.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values for graphics.

Details

Replaces a t-table by providing the corresponding t-cutoff, the critical value based on the corresponding quantile, as well as a plot that illustrates the tail probabilities. Also compare to the standardized normal curve.

Value

`cutoff`: Cutoff-value, the corresponding quantile.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

`qt`, `pnorm`.

Examples

```
# t-distribution with 0.025 cutoffs for degrees of freedom of 15
prob_tcut(15)
```

`prob_znorm` *Plot a Normal Curve with Shaded Intervals by Standard Deviation*

Description

Display a normal curve with shading according to the z-score, the number of standard deviations from the mean.

Usage

```
prob_znorm(mu=0, sigma=1, color_border="gray10",
         r=.10, g=.34, b=.94, a=.20,
         xlab="", ylab="", main="",
         y_axis=FALSE, z=TRUE, axis_size=.9,
         pdf_file=NULL, width=5, height=5, ...)
```

Arguments

mu	Population mean of normal distribution.
sigma	Population standard deviation of normal distribution.
color_border	Color of the border of the normal curve.
r	Red component of fill color, from 0 to 1.
g	Green component of fill color, from 0 to 1.
b	Blue component of fill color, from 0 to 1.
a	Alpha component of fill color, that is, the transparency, from 0 to 1.
xlab	Label for the horizontal axis_
ylab	Label for the optional vertical axis_
main	Label for the graph title.
y_axis	If TRUE, then a vertical axis is included.
z	If TRUE, then include z-values on the horizontal-axis_ Set to FALSE if mu=0 and sigma=1.
axis_size	Magnification factor for the axis labels, the value of axis_cex.
pdf_file	Name of the pdf file to which graphics are redirected.
width	Width of the pdf file in inches.
height	Height of the pdf file in inches.
...	Other parameter values for graphics.

Details

Provide a normal curve with shading of each interval defined by the number of standard deviations from the mean. The layers are written with transparency, and over-written so that the middle interval is the darkest and the most extreme intervals, beyond three standard deviations from the mean, are the lightest. Specify a=0 to turn off the colors. Higher values of the alpha channel, as specified by a, yield darker colors. Specify a=1 for the same solid color for all intervals.

The normal densities are calculated with dnorm and plotted with plot.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

dnorm, plot.

Examples

```
# Mu=0, Sigma=1: Standard normal
prob_znorm()

# distribution for height of American women, mu=65.5, sigma=2.5
prob_znorm(65.5, 2.5, xlab="Height of American Women")
```

```
# do a red fill color
prob_znorm(65.5, 2.5, r=.9, xlab="Height of American Women")
```

Prop_test *Analysis of Prop_test*

Description

Abbreviation: prop

Analyze proportions, either of a single proportion against a fixed alternative, a set of proportions evaluated for equality, or a goodness-of-fit test for a single categorical variable or a test of independence for multiple variables.

Usage

```
Prop_test(variable=NULL, success=NULL, by=NULL, data=d,
          n_succ=NULL, n_fail=NULL, n_tot=NULL, n_table=NULL,
          Yates=FALSE, pi=NULL, digits_d=3, ...)

prop(...)
```

Arguments

variable	Numerical variable to analyze.
success	Value of variable considered a success.
by	Compare proportions over groups, the values of this categorical variable.
data	Data frame that contains the variable to analyze.
n_succ	Number of successes.
n_fail	Number of trials, either provide this or n.
n_tot	Number of trials, either provide this or q.
n_table	Path name of the file that contains a frequency table.
Yates	Set to TRUE to implement Yate's correction factor where applicable.
pi	Value of null hypothesized probability.
digits_d	Number of significant digits for each of the displayed summary statistics.
...	Parameter values passed to Prop_test.

Details

The analysis of proportions is of two primary types.

For one or more samples of data, focus on a single value of a categorical variable, traditionally called a success. Analyze the resulting proportion of occurrence for a single sample or compare proportions of occurrence of a success across distinct samples of data, what is called a test of homogeneity.

For a single sample, compare proportions from a contingency table. These tests are called a goodness-of-fit test for a single variable and a test of independence for multiple variables.

From standard base R functions, the lessR function `Prop_test()`, abbreviated `prop()`, provides for either type of the analysis for proportions. To use, enter either the original data from which the sample proportions are computed, or directly enter already computed sample frequencies from which the proportions are computed.

TEST OF HOMOGENEITY
When analyzing the original data, an entered value for the parameter `success` for the categorical variable of interest, indicated by parameter `variable`, triggers the test of homogeneity. For a single proportion the analysis is the exact binomial test. If the proportions are entered directly, indicate the number of successes and the total number of trials with the `n_succ` and `n_tot` parameters, each as a single value for a single sample or as vectors of multiple values for multiple samples.

TEST OF UNIFORM GOODNESS-OF-FIT
To test for goodness-of-fit from the original data, just enter the name of the categorical variable. To test from the proportions, specify the proportions as a vector with the `n_tot` parameter.

TEST OF INDEPENDENCE
Without a value for `success` or `n_succ` the analysis is of goodness-of-fit or independence. For the test of independence, to enter the joint frequency table directly, store the frequencies in a file accessible from your computer system. One possibility is to enter the numbers into a text file with file type '.csv' or '.txt'. Enter the numbers with a text editor, or with a word processor saving the file as a text file. With this file format, separate the adjacent values in each row with a comma, as indicated below. Or, enter the numbers into an MS Excel formatted file with file type '.xlsx'. Enter only the numeric frequencies, no labels. Use the parameter `n_table` to indicate the path name to the file, enclosed in quotes. Or, leave the quotes empty to browse for the joint frequency table.

To conduct the test from the data, enter the names of the two categorical variables. The variable listed first is the parameter 'variable'. The second listed variable is for the parameter 'by', the name of which must be included in the function call.

See the corresponding vignette for more detail and examples.
Enter `browseVignettes("lessR")`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

`binom.test`.

Examples

```
# generate data
Classvalues <- c("Freshman", "Sophomore", "Junior", "Senior")
Goodvalues <- c("Nice", "OK", "Mean")
Class <- sample(Classvalues, size=250, replace=TRUE)
Goodness <- sample(Goodvalues, size=250, replace=TRUE)
d <- data.frame(Class, Goodness)

# Test a single proportion
Prop_test(variable=Goodness, success="Nice")

# Test multiple proportions, one each for each level of Plan
Prop_test(Goodness, "Nice", by=Class)

# Test of independence
Prop_test(Goodness, by=Class)

# Same example as for the base R binom.test
Prop_test(n_succ=682, n_fail=243, p=.75, digits_d=2)
```

Read *Read Contents of a Data File with Optional Variable Labels and Feedback*

Description

Abbreviation: rd, rd_lbl, Read2

Reads the contents of the specified data file into an R data table, what R calls a data frame. By default the format of the file is detected from its filetype: comma or tab separated value text file from .csv, SPSS data file from .sav, SAS data from from .sas7bdat, or R data file from .rda, and Excel file from .xls, .xlsx using Alexander Walker's openxlsx package, or .ods using Gerrit-Jan Schutten and Chung-hong Chan plus other contributor's readODS package. Specify a fixed width formatted text data file to be read with the required R widths option. Identify the data file by either browsing for the file on the local computer system with Read(), or identify the file with the first argument a character string in the form of a path name or a web URL (except for .Rda files which must be on the local computer system).

Any variable labels in a native SPSS are automatically included in the data file. See the details section below for more information. Variable labels can also be added and modified individually with the lessR function label, and more comprehensively with the VariableLabels function.

The function provides feedback regarding the data that is read by invoking the lessR function details. The default brief form of this function invoked by default only lists the input files, the variable name table, and any variable labels.

The lessR function corRead reads a correlation matrix.

Usage

```
Read(from=NULL, format=NULL, var_labels=FALSE, widths=NULL,

         missing="", n_mcut=1,
         miss_show=30, miss_zero=FALSE, miss_matrix=FALSE,

         max_lines=30, sheet=1, row_names=NULL,

         brief=TRUE, quiet=getOption("quiet"),

         fun_call=NULL, ...)

rd(...)
rd_lbl(..., var_labels=TRUE)
Read2(..., sep=";", dec=",")
```

Arguments

from	File reference included in quotes, either empty to browse for the data file, a full path name or web URL, or the name of a data file included with lessR, such as "Employee". A URL begins with http://.
format	Format of the data in the file, not usually specified because set by default according to the file type of the file to read: .csv, .tsv or .txt read as a text file, .xls, .xlsx read as an Excel file, or .ods as an OpenDocument Spreadsheet file. .feather and .parquet for the arrow formats for feather and parquet dat files. .sav reads as an SPSS file, which also reads the variable labels if present, .sas7bdat reads as a SAS file, and .rda reads as a native R data file. If the data file is not identified by one of these file types, then explicitly set by setting to one of the following values: "csv", "tsv", "Excel", "feather", "parquet", "R", "SPSS", or "SAS".
var_labels	Set TRUE if reading a csv or Excel file of variable labels into the data frame l in which each row consists of a variable name in the first column and the corresponding variable label in the second column, and perhaps units in the third row if using Regression function to generate automatic markdown files of discursive text.
widths	Specifies the width of the successive columns for fixed width formatted data.
missing	Missing value code, which by default is literally a missing data value in the data table.
n_mcut	For the missing value analysis, list the row name and number of missing values if the number of missing exceeds or equals this cutoff. Requires brief=FALSE.
miss_show	For the missing value analysis, the number of rows, one row per observation, that has as many or missing values as n_mcut. Requires brief=FALSE.
miss_zero	For the missing value analysis, list the variable name or the row name even for values of 0, that is rows with no missing data. By default only variables and rows with missing data are listed. Requires brief=FALSE.

`miss_matrix`	For the missing value analysis, if there is any missing data, list a version of the complete data table with a 0 for a non-missing value and a 1 for a missing value.
`sep`	Character that separates adjacent values in a text file of data.
`dec`	Character that serves as the decimal separator in a number.
`max_lines`	Maximum number of lines to list of the data and labels.
`sheet`	For Excel files, specifies the work sheet to read. Provide either the worksheet number according to its position, or its name enclosed in quotes. The default is the first work sheet.
`row_names`	`FALSE` by default so no row names from the input data. Set to `TRUE` to convert the first column of input data to row names. For reading `.csv` files, can also set to the integer number of the column to convert to row names. For Excel and ODS files, only acceptable value is 1 for the first column.
`brief`	If `TRUE`, display only variable names table plus any variable labels.
`quiet`	If set to `TRUE`, no text output. Can change the corresponding system default with `style` function.
`fun_call`	Function call. Used with `Rmd` to pass the function call when obtained from the abbreviated function call `rd`.
`...`	Other parameter values define with the R read functions, such as the `read.table` function for text files, with row.names and header.

Details

By default `Read` reads text data files which are either comma delimited, `csv`, or tab-delimited data files, native Excel files of type `.xls` or `.xlsx`, native ODS files of type `.ods`, native R files with file type of `.rda`, native SAS files with file type `.sas7bdat`, and native SPSS files with file type `.sav`. Invoke the `widths` option to allow for the reading of fixed width formatted data. Calls the `lessR` function `details` to provide feedback regarding details of the data frame that was read. By default, variables defined by non-numeric variables are read as character strings. To read as `factors` specify `stringsAsFactors` as `FALSE`, unless all the values of a variable a non-numeric and unique, in which case the variable is classified as a character string.

CREATE csv FILE
One way to create a csv data file is to enter the data into a text editor. A more structured method is to use a worksheet application such as MS Excel, LibreOffice Calc, or Apple Numbers. Place the variable names in the first row of the worksheet. Each column of the worksheet contains the data for the corresponding variable. Each subsequent row contains the data for a specific observation, such as for a person or a company.

Call `help(read.table)` to view the other R options that can also be implemented from `Read`.

MECHANICS
Specify the file as with the `Read` function for reading the data into a data frame. If no arguments are passed to the function, then interactively browse for the file.

Given a csv data file, or tab-delimited text file, read the data into an R data frame called `d` with `Read`. Because `Read` calls the standard R function `read.csv`, which serves as a wrapper for `read.table`, the usual options that work with `read.table`, such as `row.names`, also can be passed through the call to `Read`.

SPSS DATA
Relies upon `read_spss` from the `haven` package to read data in the SPSS `.sav` or `.zsav` format. If the file has a file type of `.sav`, that is, the file specification ends in `.sav`, then the `format` is automatically set to `"SPSS"`. To invoke this option for a relevant data file of any file type, explicitly specify `format="SPSS"`. Each (usually) integer variable with value labels is converted into two R variables: the original numeric code with the original variable name, and also the corresponding factor with the variable labels named with the original name plus the suffix `_f`. The variable labels are also displayed for copying into a variable label file. See the SPSS section from `vignette("Read")`.

R DATA
Relies upon the standard R function `load`. By convention only, data files in native R format have a file type of `.rda`. To read a native R data file, if the file type is `.rda`, the `format` is automatically set to `"R"`. To invoke this option for a relevant data file of any file type, explicitly specify `format="R"`. Create a native R data file by saving the current data frame, usually `d`, with the `lessR` function `Write`.

Excel DATA
Relies upon the function `read.xlsx` from Alexander Walker's `openxlsx` package. Files with a file type of `.xlsx` are assigned a `format` of `"Excel"`. The `read.xlsx` parameter `sheet` specifies the ordinal position of the worksheet in the Excel file, with a default value of 1. The `row.names` parameter can only have a value of 1. Dates stored in Excel as an Excel date type are automatically read as an R Date type. See the help file for `read.xlsx` for additional parameters, such as `sheet` for the name or number of the worksheet to read and `startRow` for the row number for which to start reading data.

lessR DATA
`lessR` has some data sets included with the package: "BodyMeas", "Cars93", "Employee", "Jackets", "Learn", "Mach4", "Reading", and "StockPrice". `Read` reads each such data set by specifying its name, such as `Read("Employee")`. No specificaiton of `format` and no provided filetype, just enter the name of the data set.

FIXED WIDTH FORMATTED DATA
Relies upon `read.fwf`. Applies to data files in which the width of the column of data values of a variable is the same for each data value and there is no delimiter to separate adjacent data values_ An example is a data file of Likert scale responses from 1 to 5 on a 50 item survey such that the data consist of 50 columns with no spaces or other delimiter to separate adjacent data values_ To read this data set, invoke the `widths` option of `read.fwf`.

MISSING DATA
By default, `Read` provides a list of each variable and each row with the display of the number of associated missing values, indicated by the standard R missing value code NA. When reading the data, `Read` automatically sets any empty values as missing. Note that this is different from the R default in `read.table` in which an empty value for character string variables are treated as a regular data value. Any other valid value for any data type can be set to missing as well with the `missing` option. To mimic the standard R default for missing character values, set `missing=NA`.

To not list the variable name or row name of variables or rows without missing data, invoke the `miss_zero=FALSE` option, which can appreciably reduce the amount of output for large data sets. To view the entire data table in terms of 0's and 1's for non-missing and missing data, respectively, invoke the `miss_matrix=TRUE` option.

VARIABLE LABELS
Unlike standard R, `lessR` provides for variable labels, which can be provided for some or all of the

variables in a data frame. Store the variable labels in a separate data frame `l`. The variable labels file that is read by `Read` consists of one row for each variable for which a variable label is provided. Each row consists of either two columns, the variable name in the first column and the associated variable label in the second column, or three columns with the third column the variable units. Use the units in conjunction for enhanced readability with the automatic markdown generated by the `Rmd` parameter for the `Regression` function. The format of the file can be `csv` or `xlsx`. The data frame `Read` constructs from this input consists of one variable, called `label`, with the variable names as row names.

The `lessR` legacy approach is to store the variable labels directly with the data in the same data frame. The problem with this approach is that any transformations of the data with any function other than `lessR` transformation functions remove the variable labels. The option for reading the variable labels with the `labels` option of `Read` statement is retained for compatibility.

Reading the data from an SPSS file, however, retains the SPSS variable labels as part of the data file. The `lessR` data analysis functions will properly process these variable labels, but any non-`lessR` data transformations will remove the labels from the data frame. To retain the labels, copy them to the `l` data frame with the `VariableLabels` function with the name of the data frame as the sole argument.

The `lessR` functions that provide analysis, such as `Histogram` for a histogram, automatically include the variable labels in their output, such as the title of a graph. Standard R functions can also use these variable labels by invoking the `lessR` function `label`, such as setting `main=label(I4)` to put the variable label for a variable named I4 in the title of a graph.

Value

The read data frame is returned, usually assigned the name of `d` as in the examples below. This is the default name for the data frame input into the `lessR` data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 1, NY: CRC Press.

Alexander Walker (2018). openxlsx: Read, Write and Edit XLSX Files. R package version 4.1.0. https://CRAN.R-project.org/package=openxlsx

See Also

`read.csv`, `read.fwf`, `corRead`, `label`, `details`, `VariableLabels`.

Examples

```
# remove the # sign before each of the following Read statements to run

# to browse for a data file on the computer system, invoke Read with
#   the from argument empty
# d <- Read()
# abbreviated name
```

```
# d <- rd()

# read the variable labels from
#  the specified label file, here a Excel file with two columns,
#  the first column of variable names and the second column the
#  corresponding labels
# l <- Read("Employee_lbl", var_labels=TRUE)

# read a csv data file from the web
# d <- Read("http://web.pdx.edu/~gerbing/data/twogroup.csv")

# read a csv data file with -99 and XXX set to missing
# d <- Read(missing=c(-99, "XXX"))

# do not display any output
# d <- Read(quiet=TRUE)
# display full output
# d <- Read(brief=FALSE)

# read the built-in data set dataEmployee
d <- Read("Employee")

# read a data file organized by columns, with a
#   5 column ID field, 2 column Age field
#   and 75 single columns of data, no spaces between columns
#   name the variables with lessR function: to
#   the variable names are Q01, Q02, ..., Q74, Q75
# d <- Read(widths=c(5,2,rep(1,75)), col.names=c("ID", "Age", to("Q", 75)))
```

---

recode *Recode the Values of an Integer or Factor Variable*

Description

Recodes the values of one or more integer variables in a data frame. The values of the original variable may be overwritten with the recoded values, or the recoded values can be designated to be placed in a new variable, indicated by the new_name option. Valid values may be converted to missing, and missing values may be converted to valid values. Any existing variable labels are retained in the recoded data frame.

There is no provision to recode integer values to character strings because that task is best accomplished with the standard R factor function.

Usage

```
recode(old_vars, new_vars=NULL, old, new, data=d,
       quiet=getOption("quiet"), ...)
```

Arguments

`old_vars`	One or more variables to be recoded.
`new_vars`	Name of the new variable or variables that contain the recoded values, each name in quotes. If not provided, then the values of the original variable are replaced.
`old`	The values of the variables that are to be recoded. If the value is `"missing"` then any existing missing values are replaced by the value specified with `new`.
`new`	The recoded values, which match one-to-one with the values in `old`. If the value is `"missing"` then instead any values specified in `old` are converted to missing.
`data`	The name of the data frame from which to create the subset, which is `d` by default.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	Parameter values_

Details

Specify the values to be recoded with the required `old` parameter, and the corresponding recoded values with the required `new` parameter. There must be a 1-to-1 correspondence between the two sets of values, such as 0:5 recoded to 5:0, six items in the `old` set and six items in the `new` set.

Use `new_vars` to specify the name of the variable that contains the recoded values. If `new_vars` is not present, then the values of the original variable are overwritten with the recoded values.

Not all of the existing values of the variable to be recoded need be specified. Any value not specified is unchanged in the values of the recoded variable.

Unless otherwise specified, missing values are unchanged. To modify missing values, set `old="missing"` to covert missing data values to the specified value data value given in `new`. Or, set `new="missing"` to covert the one or more existing valid data values specified in `old` to missing data values.

Diagnostic checks are performed before the recode. First, it is verified that the same number of values exist in the `old` and `new` lists of values_ Second, it is verified that all of the values specified to be recoded in fact exist in the original data.

If the levels of a factor were to be recoded with `recode`, then the factor attribute would be lost as the resulting recoded variable would be character strings. Accordingly, this type of transformation is not allowed, and instead should be accomplished with the `Transform` and `factor` functions as shown in the examples.

Value

The recoded data frame is returned, usually assigned the name of `d` as in the examples below. This is the default name for the data frame input into the `lessR` data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

`transform`, `factor`.

Examples

```
# construct data frame
d <- read.table(text="Severity Description
1 Mild
4 Moderate
3 Moderate
2 Mild
1 Severe", header=TRUE, stringsAsFactors=FALSE)

# recode Severity into a new variable called SevereNew
d <- recode(Severity, new_vars="SevereNew", old=1:4, new=c(10,20,30,40))

# reverse score four Likert variables: m01, m02, m03, m10
d <- Read("Mach4")
d <- recode(c(m01:m03,m10), old=0:5, new=5:0)

# convert any 1 for Plan to missing
# use Read to put data into d data frame
# write results to newdata data frame
d <- Read("Employee")
newdata <- recode(Plan, old=1, new="missing")

# for Years and Salary convert any missing value to 99
d <- recode(c(Years, Salary), old="missing", new=99)


# ------------------------------------
# convert between factors and integers
# ------------------------------------

# recode levels of a factor that should remain a factor
#   with the Transform and factor functions
# using recode destroys the factor attribute, converting to
#   character strings instead, so Recode does not allow
d <- Read("Employee")
d <- Transform(
    Gender=factor(Gender, levels=c("F", "M"), labels=c("Female", "Male"))
)

# recode levels of a factor to convert to integer first by
#   converting to integer with Transform and as.numeric
# here Gender has values M and F in the data
# integers start with 1 through the number of levels, can use
#   recode() to change this if desired, such as to 0 and 1
d <- Transform(Gender=as.numeric(Gender))
d <- recode(Gender, old=c(1,2), new=c(0,1))

# recode integer values to levels of a factor with value labels
#   instead of recode()
# here Gender has values 0 and 1 in the data
d <- Read("Mach4")
d <- Transform(
```

```
    Gender=factor(Gender, levels=c(0,1), labels=c("Male","Female"))
    )
# -----------------------------------
```

---

regPlot *regPlot Analysis*

Description

Following a call to the lessR function Regression, in which the returned values of the function are saved into an object, allows the default plots generated by Regression to be accessed one at a time_ The specific motivation for this function is to allow custom placement of the graphs from the regression analysis from within knitr. Usually the graphics=FALSE parameter is set on the call to Regression within knitr to suppress the normal graphic output that leads to the generation of the graphs at the beginning of the knitr output.

Usage

```
regPlot(out, type, d.ancova, digits_d=NULL, pred.intervals=TRUE,
         res_sort=c("cooks","rstudent","dffits","off"),
         n_res_rows=NULL, cooks_cut=1, scatter_coef=NULL,
         pdf=FALSE, width=5, height=5, manage.gr=FALSE, ...)
```

Arguments

out	The object returned by the lessR function Regression.
type	Type of plot: 1 plots the scatter plot for a single predictor variable, or the scatter plot matrix for multiple predictors. If a single scatter plot, then the confidence and prediction intervals are included. 2 plots the density and histogram of residuals and 3 plots a scatter plot of the residuals with the fitted values_
d.ancova	If not NULL, then an ANCOVA design with 1 grouping variable and 1 covariate, which contains the original data.
digits_d	For the Basic Analysis, the number of decimal digits, set by default to at least 3 or the largest number of digits in the values of the response variable plus 1.
pred.intervals	If set to FALSE, the scatter plot for a single predictor with the response does not contain prediction and confidence intervals.
res_sort	Default is "cooks", for specifying Cook's distance as the sort criterion for the display of the rows of data and associated residuals. Other values are "rstudent" for externally Studentized residuals, "dffits" for dffits and "off" to not sort the rows of data.
n_res_rows	Default is 20, which lists the first 20 rows of data sorted by the specified sort criterion. To disable residuals, specify a value of 0. To see the output for all observations, specify a value of "all".
cooks_cut	Cutoff value of Cook's Distance at which observations with a larger value are flagged in red and labeled in the resulting scatterplot of Residuals and Fitted Values. Default value is 1.0.

`scatter_coef`	Display the correlation coefficients in the upper triangle of the scatterplot matrix.
`pdf`	If `TRUE`, then graphics are written to pdf files.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`manage.gr`	Usually leave `FALSE`. Refers to graphic management of the `lessR` system.
`...`	Other parameter values for R function `lm` which provides the core computations.

Details

OVERVIEW
The ability to separate plots is particularly useful with `knitr` to break up the output to intersperse comments between the plots. For Plot 1, for single predictor a scatter plot with the regression line and confidence and prediction intervals is produced. Otherwise a scatter plot matrix of all the variables in the models is obtained.

To help assess the validity of the model, Plot 2 is of the distribution of the residuals, histogram and density plots, both general and normal. Plot 3 plots the residuals against the fitted value and also identifies the points with the largest values of Cook's distance.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2014). R Data Analysis without Programming, Chapters 9 and 10, NY: Routledge.

See Also

`lm`, `Regression`

Examples

```
# read internal data set
d <- rd("Reading", quiet=TRUE)
# do regression analysis, save result into out
reg.out <- reg(Reading ~ Verbal)
# The full output already contains these plots, obtained by
# entering the name of the saved object
reg.out
# Particularly for knitr it is useful to obtain the plots
#   separately from the full output
# Get the scatter plot of the data with the regression line
#   and prediction and confidence intervals
regPlot(reg.out, 1, NULL)

# Can use with multiple regression for the scatter plot matrix
r <- reg(Reading ~ Verbal + Absent + Income)
regPlot(r, 1, NULL, scatter_coef=TRUE)
```

Regression *Regression Analysis*

Description

Abbreviation: reg, reg_brief

Provides a regression analysis with extensive output, including graphics, from a single, simple function call with many default settings, each of which can be re-specified. The computations are obtained from the R function lm and related R regression functions. The outputs of these functions are re-arranged and collated.

By default the data exists as a data frame with the default name of d, or specify explicitly with the data option. Specify the model in the function call as an R formula, that is, for a basic model, the response variable followed by a tilde, followed by the list of predictor variables, each pair separated by a plus sign, such as reg(Y ~ X1 + X2).

Output is generated into distinct segments by topic, organized and displayed in sequence by default. When the output is assigned to an object, such as r in r <- reg(Y ~ X), the full or partial output can be accessed for later analysis and/or viewing. A primary such analysis is with knitr for dynamic report generation, run from R directly or from within RStudio. The input instructions to knitr are written comments and interpretation with embedded R code, called R~Markdown. Doing a knitr analysis is to "knit" these comments and subsequent output together so that the R output is embedded in the resulting document – either html, pdf or Word – by default with explanation and interpretation. Generate a complete R~Markdown file with filetype (.Rmd) from the Rmd option. Simply specify the option with a file name in quotes, then run the Regression analysis to create the markdown file. Open the newly created .Rmd file in RStudio and click the knit button to create a formatted document that consists of the statistical results plus interpretative comments. See the sections arguments, value and examples for more information.

Usage

```
Regression(my_formula, data=d, filter=NULL,
         digits_d=NULL, n_cat=getOption("n_cat"),

         Rmd=NULL, Rmd_browser=TRUE,
         Rmd_format=c("html", "word", "pdf", "odt", "none"),
         Rmd_data=NULL, Rmd_custom=NULL, Rmd_dir=path.expand("~/reg"),
         Rmd_labels=FALSE,
         results=getOption("results"), explain=getOption("explain"),
         interpret=getOption("interpret"), code=getOption("code"),

         text_width=120, brief=getOption("brief"), show_R=FALSE,
         plot_errors,

         n_res_rows=NULL, res_sort=c("cooks","rstudent","dffits","off"),
         n_pred_rows=NULL, pred_sort=c("predint", "off"),
         subsets=NULL, best_sub=c("adjr2", "Cp"), cooks_cut=1,
```

```
          scatter_coef=TRUE, mod=NULL, mod_transf=c("center", "z", "none"),

          X1_new=NULL, X2_new=NULL, X3_new=NULL, X4_new=NULL,
          X5_new=NULL, X6_new=NULL,

          kfold=0, seed=NULL,
          new_scale=c("none", "z", "center", "0to1", "robust"),
          scale_response=FALSE,

          quiet=getOption("quiet"),
          graphics=TRUE, pdf=FALSE, width=6.5, height=6.5, refs=FALSE,
          fun_call=NULL, ...)

reg(...)
reg_brief(..., brief=TRUE)
```

Arguments

my_formula	Standard R formula for specifying a model. For example, for a response variable named Y and two predictor variables, X1 and X2, specify the corresponding linear model as Y ~ X1 + X2.
data	The default name of the data frame that contains the data for analysis is d, otherwise explicitly specify. If knitting and rendering the generated R~Markdown for an interpretative output as specified by the Rmd parameter, then this data frame must first be read by the lessR function Read.
filter	A logical expression that specifies a subset of rows of the data frame to analyze.
digits_d	For the Basic Analysis, it provides the number of decimal digits, set by default to at least 2 or the largest number of digits in the values of the response variable plus 1.
n_cat	Number of categories, specifies the largest number of unique, equally spaced integer values of a variable for which the variable will be analyzed as categorical instead of continuous. Default is 0. Use to specify that such variables are to be analyzed as categorical, a kind of informal R factor. *[deprecate]*: Best to convert a categorical integer variable to a factor.
Rmd	File name for the automatically generated R Markdown file, if specified. The file type is .Rmd, a simple text file that can be edited with any text editor, including RStudio to generate custom output.
Rmd_browser	If html format for Rmd rendering, then automatically open output in a browser.
Rmd_format	Format of one or more rendered R Markdown file formats, expressed in any combination of uppercase and lowercase letters. Default is "html", with a browser view automatic, or "word", "odt", "pdf" (if LaTeX is available), or "none". Requires pandoc installed, such as from RStudio.
Rmd_data	The default file reference of the data file when running the generated R Markdown file is the last data file as read by Read (with the unabbreviated version of the function name). To refer to a different file to read specify the path name or web URL of the file.

`Rmd_custom` Vector of input text sections in the Rmd file for which to convert.

`Rmd_dir` Directory where custom input text files are located for the `Rmd` option.

`Rmd_labels` Label each section of the markdown output according to the name of its input file.

`results` By default TRUE. If set to `FALSE` the results are not provided in the R Markdown document, relying upon the interpretations. Can set globally with `style(results=FALSE)`.

`explain` By default TRUE. If set to `FALSE` the explanations are not provided in the R Markdown document. Can set globally with `style(explain=FALSE)`.

`interpret` By default TRUE. If set to `FALSE` the interpretations of the results are not provided in the R Markdown document. Can set globally with `style(interpret=FALSE)`.

`code` By default TRUE. If set to `FALSE` the R code that generates the results is not provided in the R Markdown file. Can set globally with `style(code=FALSE)`.

`text_width` Width of the text output at the console.

`brief` If set to `TRUE`, reduced text output. Can change system default with `style` function.

`show_R` Display the R instructions that yielded the `lessR` output, albeit without the additional formatting of the results such as combining output of different functions into a table.

`plot_errors` For a one-predictor model, plot the line segment that joins each point to the regression line, illustrating the size of the residuals.

`n_res_rows` Default is 20, which lists the first 20 rows of data sorted by the specified sort criterion. To disable residuals, specify a value of 0. To view the output for all observations, specify a value of `"all"`.

`res_sort` Default is `"cooks"`, for specifying Cook's distance as the sort criterion for the display of the rows of data and associated residuals. Other values are `"rstudent"` for externally Studentized residuals, `"dffits"` for dffits and `"off"` to not sort the rows of data.

`n_pred_rows` Default is 3, which lists prediction intervals only for the first, middle and last 3 rows of data, unless there are 25 or less rows of data when all rows are displayed. To disable prediction intervals, specify a value of 0. To see the output for rows of data, specify a value of `"all"`.

`pred_sort` Default is `"predint"`, which sorts the rows of data and associated intervals by the lower bound of each prediction interval. Turn off this sort by specifying a value of `"off"`.

`subsets` Default is to produce the analysis of the fit based on adjusted R-squared for all possible model subsets of size 10 for each number of predictors, from the `leaps` package. Set to `FALSE` to turn off. Defaults lists a maximum of the first 50 values. Specify an integer to change the maximum.

`best_sub` Criterion for selecting best subsets of predictor variables, with default of `"adjr2"` or choose Mallow's `"Cp"` statistic.

`cooks_cut`	Cutoff value of Cook's Distance at which observations with a larger value are flagged in red and labeled in the resulting scatterplot of Residuals and Fitted Values. Default value is 1.0.
`scatter_coef`	Display the correlation coefficients in the upper triangle of the scatterplot matrix.
`mod`	Declare one continuous (numeric) predictor variable a moderator variable in a two predictor model.
`mod_transf`	Applies when `mod` specified, rescales the predictor variables, with default `"center"`, and options of `"z"` for standardize and `"none"`.
`X1_new`	Values of the first listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`X2_new`	Values of the second listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`X3_new`	Values of the third listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`X4_new`	Values of the fourth listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`X5_new`	Values of the fifth listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`X6_new`	Values of the sixth listed numeric predictor variable for which forecasted values and corresponding prediction intervals are calculated.
`kfold`	Number of K-fold cross-validations. If conducted, only the cross-validation output shown.
`seed`	Parameter `kfold` generates random partitions, folds, of data. Set the seed to an integer to recover the same random partitions on subsequent runs.
`new_scale`	Transform numeric predictor variables with more than two unique values to the specified metric before conducting the analysis, `"z"`, `"center"`, `"0to1"`, or `"robust"`. Applies to `kfold` separately to the training and testing folds as well to avoid data leakage.
`scale_response`	If doing a rescale with `new_scale`, by default do not scale the response variable, or set to `TRUE` to rescale along with the predictor variables.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`graphics`	Produce graphics. Default is `TRUE`. In `knitr` can be useful to set to `FALSE` so that `regPlot` can be used to place the graphics within the output file.
`pdf`	If `TRUE`, then graphics are written to pdf files.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`refs`	If `TRUE`, then list the references for R and the packages used from which functions were used to generate the output.
`fun_call`	Function call. Used internally with `knitr` to pass the function call when obtained from the abbreviated function call `reg`. Not usually invoked by the user.
`...`	Other parameter values for R function `lm` which provides the core computations.

Details

OVERVIEW
The purpose of Regression is to combine the following function calls into one, as well as provide ancillary analyses such as as graphics, organizing output into tables and sorting to assist interpretation of the output, as well as generate R Markdown to run through knitr, such as with RStudio, to provide extensive interpretative output.

The basic analysis successively invokes several standard R functions beginning with the standard R function for estimation of a linear model, lm. The output of the analysis of lm is stored in the object lm.out, available for further analysis in the R environment upon completion of the Regression function. By default reg automatically provides the analyses from the standard R functions, summary, confint and anova, with some of the standard output modified and enhanced. The correlation matrix of the model variables is obtained with cor function. The residual analysis invokes fitted, resid, rstudent, and cooks.distance functions. The option for prediction intervals calls the standard R function predict, once with the argument interval="confidence" and once with interval="prediction". The lessR Density function provides the histogram and density plots for the residuals and the ScatterPlot function provides the scatter plots of the residuals with the fitted values and of the data for the one-predictor model. The pairs function provides the scatterplot matrix of all the variables in the model. Thomas Lumley's leaps package contains the leaps function that provides the analysis of the fit of all possible model subsets.

INPUT DATA FRAME
The name d is by default provided by the Read function included in this package for reading and displaying information about the data in preparation for analysis. If all the variables in the model are not in the same data frame, the analysis will not complete. Specify the name of the data frame for analysis with the data option if the name is not the default d.

The filter parameter subsets rows (cases) of the input data frame according to a logical expression. Use the standard R operators for logical statements as described in Logic such as & for and, | for or and ! for not, and use the standard R relational operators as described in Comparison such as == for logical equality != for not equals, and > for greater than. See the Examples.

TEXT OUTPUT
The output is produced in pieces by topic (see values below), automatically collated by default in the final output. But the pieces are available for later reference if the output of the function is directed toward an object, such as r in r <- reg(Y ~ X). This is especially useful if the pieces are accessed within knitr or individual pieces are displayed at the console.

The text output is organized to provide the most relevant information while at the same time minimizing the total amount of output, particularly for analyses with large numbers of observations (rows of data), the display of which is by default restricted to only the most interesting or representative observations in the analyses of the residuals and predicted values. Additional economy can be obtained by invoking the brief=TRUE option, or run reg_brief, which limits the analysis to just the basic analysis of the estimated coefficients and fit, and if X1_new, etc. are requested, the relevant rows of forecasted values:w .

R MARKDOWN
An R~Markdown file ready for knitting and rendering into one of several formats can be obtained by specifying a value for Rmd. For the specified file name, the directory to which the file is written is displayed on the console text output, and the file type .Rmd is automatically appended to the specified name if it is not included in the specification.

To access the same data file for the regression analysis from running Regression from the R console, and that accomplished by knitting the generated R~Markdown, first read the data into R with the lessR Read function. That function stores the name of the last data file read so that it can be accessed via R as the markdown is knit and then rendered into the specified format. The default rendering is to HTML, but other formats can be specified with Rmd_format.

The output from Rmd is conceptually partitioned into five parts: results, explanations of the results, interpretations of the results, documentation o the code, and the code itself. By default all available output is generated but the flags results, explain, interpret, document, code can be set to FALSE to reduce the output. The options can be specified in a specific function all or set globally, such as with options(explain=FALSE). Turning off all five flags leaves just the outline of the potential output and a bare minimum of results.

Both any existing variable labels and variable units are included in the output to the R~Markdown file. Any variable units set as a dollar, are set as USD dollars and cents in the output, displayed with a dollar sign.

The default analysis provides as text output to the console the model's parameter estimates and corresponding hypothesis tests and confidence intervals, goodness of fit indices, the ANOVA table, correlation matrix of the model's variables, analysis of residuals and influence as well as the confidence and prediction intervals for each observation in the model. Also provided, for multiple regression models, collinearity analysis of the predictor variables and adjusted R-squared for the corresponding models defined by each possible subset of the predictor variables.

The Markdown is produced from input files, one for each section of the rendered document. Find the default files and their names at:\ system.file("Rmd/reg/", package="lessR")\ The Rmd_dir option specifies a location for custom input files. The Rmd_custom parameter specifies which default files should be replaced by custom files, anywhere from any one of them to all eight.

DECIMAL DIGITS
The number of decimal digits displayed on the output is, by default, the maximum number of decimal digits for all the data values of the response variable. Or, this value can be explicitly specified with the digits_d parameter.

Visualizations
Three default graphs are provided. When running R by itself, by default the graphs are written to separate graphics windows (which may overlap each other completely, in which case move the top graphics windows). Or, the pdf option may be invoked to save the graphs to a single pdf file called regOut.pdf. Within RStudio the graphs are successively written to the Plots window. Within knitr from RStudio the graphics will all appear by default at the beginning of the output. Or set to graphics=FALSE, and generate them individually with the accompanying function regPlot at the desired location within the file.

1. A histogram of the residuals includes the superimposed normal and general density plots from the Density function included in this lessR package. The overlapping density plots, which both overlap the histogram, are filled with semi-transparent colors to enhance readability.

2. A scatterplot of the residuals with the fitted values is also provided from the ScatterPlot function included in this package. The point corresponding to the largest value of Cook's distance, regardless of its size, is plotted in red and labeled and the corresponding value of Cook's distance specified in the subtitle of the plot. Also by default all points with a Cook's distance value larger than 1.0 are plotted in red, a value that can be specified to any arbitrary value with the cooks_cut option. This scatterplot also includes the lowess curve.

3. For models with a single predictor variable, a scatterplot of the data is produced, which also includes the regression line and corresponding confidence and prediction intervals. As with the density histogram plot of the residuals and the scatterplot of the fitted values and residuals, the scatterplot includes a colored background with grid lines. For multiple regression models, a scatterplot matrix of the variables in the model with the `lowess` best-fit line of each constituent scatterplot is produced. If the `scatter_coef` option is invoked, each scatterplot in the upper-diagonal of the correlation matrix is replaced with its correlation coefficient.

RESIDUAL ANALYSIS
By default the residual analysis lists the data and fitted value for each observation as well as the residual, Studentized residual, Cook's distance and dffits, with the first 20 observations listed and sorted by Cook's distance. The `res_sort` option provides for sorting by the Studentized residuals or not sorting at all. The `n_res_rows` option provides for listing these rows of data and computed statistics statistics for any specified number of observations (rows). To turn off the analysis of residuals, specify `n_res_rows=0`.

PREDICTION INTERVALS
The output for the confidence and prediction intervals includes a table with the data and fitted value for each observation, the lower and upper bounds for the confidence interval and the prediction interval, and the wide of the prediction interval. The observations are sorted by the lower bound of each prediction interval. If there are 25 or more observations then the information for only the first three, the middle three and the last three observations is displayed. To turn off the analysis of prediction intervals, specify `n_pred_rows=0`, which also removes the corresponding intervals from the scatterplot produced with a model with exactly one predictor variable, yielding just the scatterplot and the regression line.

The data for the default analysis of the prediction intervals is for the values of the predictor variables for each observation, that is, for each row of the data. New values of the predictor variables can be specified for the calculation of the prediction intervals by providing values for the options `X1_new` for the values of the first listed predictor variable in the model, `X2_new` for the second listed predictor variable, and so forth for up to five predictor variables, and all predictor variables are numeric. To provide these values, use functions such as `seq` for specifying a sequence of values and `c` for specifying a vector of values. For multiple regression models, all combinations of the specified new values for all of the predictor variables are analyzed.

RELATIONS AMONG THE VARIABLES
By default the correlation matrix of all the variables in the model is displayed, and, for multiple regression models, collinearity analysis is provided. Also provided are the first 50 models with the largest R squared adjusted from each possible model from an analysis of all possible subsets of the predictor variables. This all subsets analysis requires the `leaps` function from the `leaps` package. These contributed packages are automatically loaded if available. To turn off the all possible sets option, set `subsets=FALSE`.

RECODE PREDICTOR VARIABLES
The `new_scale` parameter provides for recoding the values of the predictor variables according to several different transformations: `"z"`, `"center"`, `"0to1"`, or `"robust"`. The later is a robust version of classic standardization in which the mean is replaced by the median and the standard deviation by the IQR. All numeric predictor variables with more than two values are standardized.

So any numeric variable with more than two values that is a categorical variable should be first converted to an R factor. If there are some numeric predictor variables that should not be standardized, such as an interaction term with centered variables that define the interaction, then the rescaling should be done separately, such as with base~R function `scale` or lessR `rescale`.

ANCOVA
If there are two predictor variables, one categorical and one continuous, an analysis of covariance is performed. The resulting scatterplot is of the continuous response variable and predictor variable, at each level of the categorical variable. To address the unbalanced ANOVA design, the Type~II sums of squares are reported for each effect. The regression model for each level of the categorical variable are displayed.

A categorical variable is defined as either an R `factor` or a non-numeric variable. If numeric and categorical, then explicitly define the categorical variable as a factor.

MODERATOR VARIABLE
For two predictor models, one of the predictor variables can be entered into the analysis as a moderator variable with the `mod` parameter. By default the two predictor variables are centered, so their means become zero. Then a third variable is entered into the model, the interaction of the two centered variables, computed by multiplication of their respective values, row by row. The potential interaction is visually displayed by plotting response Y against predictor X, at three different values of continuous W: the mean and 1 standard deviation above and below the mean.

For predictor variable, X, second predictor as a potential moderator, W, and response Y, enter the following R input.
`reg(Y ~ X + W, mod=W)`
From this, with now centered variables X and Y, the following multiple regression model is automatically defined.
`Y^ = b0 + bx(X) + bw(W) + bxw(XW)`
From that model, the functions sets the moderator variable W to each of the three constant values, `Wc`, and solves for the given value `Wc` to visually plot the potential interaction.

INVOKED R OPTIONS
The `options` function is called to turn off the stars for different significance levels (show.signif.stars=FALSE), to turn off scientific notation for the output (scipen=30), and to set the width of the text output at the console to 120 characters. The later option can be re-specified with the `text_width` option. After `Regression` is finished with a normal termination, the options are re-set to their values before the `Regression` function began executing.

COLOR THEME
A color theme for all the colors can be chosen for a specific plot with the `colors` option. Or, the color theme can be changed for all subsequent graphical analysis with the `lessR` function `style`. The default color theme is `lightbronze`, but a gray scale is available by removing the bronze background, such as with `style(window_fill="white")` or with `"gray"`. Other themes are available as explained in `style`.

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see `Read`.

Value

The output can optionally be returned and saved into an `R` object, otherwise it simply appears at the console. The components of this object are redesigned in `lessR` version 3.3 into (a) pieces of text that form the readable output and (b) a variety of statistics. The readable output are character strings such as tables amenable for viewing and interpretation. The statistics are numerical values amenable for further analysis, such as to be referenced in a subsequent `knitr` document. The motivation of these two types of output is to facilitate `knitr` documents, as the name of each piece, preceded by the name of the saved object followed by a dollar sign, can be inserted into the knitr document (see examples).

TEXT OUTPUT
out_background: variables in the model, rows of data and retained
out_estimates: estimated coefficients, hypothesis tests and confidence intervals
out_fit: fit indices; st dev of residuals; R-sq with adj and PRESS versions
out_anova: analysis of variance
out_cor: correlations among all variables in the model
out_collinear: collinearity analysis
out_subsets: R squared adjusted for all (or many) possible subsets
out_residuals: residuals
out_predict: analysis of residuals and influence
out_ref: references if selected on the Regression function call
out_Rmd: lists the name and location of the generated Rmd file
out_plots: list of plots generated if more than one
out_suggest: list of suggested other analyses

Separated from the rest of the text output are the major headings, which can be not included with custom collations of the output. out_title_bck: BACKGROUND
out_title_basic: BASIC ANALYSIS
out_title_rel: RELATIONS AMONG THE VARIABLES
out_title_res: ANALYSIS OF RESIDUALS AND INFLUENCE
out_title_pred: FORECASTING ERROR

STATISTICS
call: function call that generated the analysis
formula: model formula that specifies the model
vars: vector of variable names in the model
n.vars: number of variables in the model
n.obs: number of rows of data submitted for analysis
n.keep: number of rows of data retained in the analysis
coefficients: estimated regression coefficients
sterrs: standard errors of the estimated coefficients
tvalues: t-values of the estimated coefficients for null of 0
pvalues: p-values from the t-tests of the estimated coefficients
cilb: lower bound of 95% confidence interval of estimate
ciub: upper bound of 95% confidence interval of estimate
anova_model: model df, ss, ms, F-value and p-value
anova_residual: residual df, ss and ms
anova_total: total df, ss and ms
se: standard deviation of the residuals
resid_range: 95% range of normally distributed fitted residuals
Rsq: R-squared
Rsqadj: adjusted R-squared
PRESS: PRESS sum of squares
RsqPRESS: PRESS R-squared
m_se: K-fold average of the standard deviation of residuals. m_MSE: K-fold average of the MSE. m_Rsq: K-fold average of R-squared. cor: correlation matrix of all variables in the model

`tolerances`: tolerance of each predictor variable for collinearity analysis
`VIF`: variance inflation factor for each predictor variable
`resid.max`: five largest values of the residuals on which the output is sorted
`pred_min_max`: Rows with the smallest and largest prediction intervals
`residuals`: residuals
`fitted`: fitted values
`cooks.distance`: Cook's distance
`model`: data retained for the analysis
`terms`: terms specified for the analysis

Although not typically needed for analysis, if the regression output is assigned to an object named, for example, `r`, then the complete contents of the object can be viewed directly with the `unclass` function, here as `unclass(r)`. Invoking the `class` function on the saved object reveals a class of `out_all`. The class of each of the text pieces of output is `out`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Lumley, T., `leaps` function from the `leaps` package.

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapters 11-13, NY: Routledge.

Gerbing, D. W. (2021). Enhancement of the Command-Line Environment for use in the Introductory Statistics Course and Beyond, *Journal of Statistics and Data Science Education*, 29(3), 251-266, https://www.tandfonline.com/doi/abs/10.1080/26939169.2021.1999871.

Xie, Y. (2013). Dynamic Documents with R and knitr, Chapman & Hall/CRC The R Series.

See Also

`formula`, `lm`, `summary.lm`, `anova`, `confint`, `fitted`, `resid`, `rstudent`, `cooks.distance`, `Nest`, `regPlot`

Examples

```
# read internal data set
d <- rd("Reading", quiet=TRUE)
# do not need all this data, so take only 30% to reduce CPU time
d <- Subset(random=.3)

# one-predictor regression
# Provide all default analyses including scatterplot etc.
# Can abbreviate Regression with reg
Regression(Reading ~ Verbal)
# Provide only the brief analysis on standardized variables
#  with 3-fold cross-validations
reg_brief(Reading ~ Verbal, new_scale="z", kfold=3)
```

```
# Access the pieces of output, here in an object named \code{r}
r <- reg(Reading ~ Verbal + Absent + Income)
# Display all output at the console in the standard sequence
r
# list the names of all the saved components
names(r)
# Display just the estimated coefficients and their inferential analysis
r$out_estimates



# Generate an R markdown file with the option: Rmd
# Output file here will be read.Rmd, a simple text file that can
#   be edited with any text editor including RStudio from which it
#   can be knit to generate dynamic output to a Word document,
#   pdf file or html file, as well as automatically rendered
# Here knit into an html file, but do not display
#reg(Reading ~ Verbal + Absent, Rmd="read", Rmd_browser=FALSE)

# generate interpretative R markdown file and render Word and odt
#reg(Reading ~ Verbal + Absent, Rmd="eg", Rmd_format=c("word", "odt"))

# just for incomes > 100000 and less than 5 days absent
Regression(Reading ~ Verbal, filter=(Income > 100 & Absent < 5))

# standardize
Regression(Reading ~ Verbal, new_scale="z")


# Multiple regression model
# Save the three output plots as pdf files 4 inches square
#Regression(Reading ~ Verbal + Absent + Income, pdf=TRUE,
#   width=4, height=4)

# Compare nested models
# Reduced model:  Reading ~ Verbal
# Full model:  Reading ~ Verbal + Income + Absent
Nest(Reading, Verbal, c(Income, Absent))


# Specify new values of the predictor variables to calculate
#  forecasted values and the corresponding prediction intervals
# Specify an input data frame other than d, see help(mtcars)
Regression(mpg ~ hp + wt, data=mtcars,
  X1_new=seq(50,350,50), X2_new=c(2,3))

# Indicator (dummy) variable
#d <- Read("Employee", quiet=TRUE)
#reg(Salary ~ Dept)
```

rename — *Rename One or More Variables in a Data Frame*

Description

rename renames a single variable or a vector of variables in a data frame.

Usage

```
rename(data, from, to)
```

Arguments

data	Data frame that contains the relevant variables.
from	One or more variables to rename.
to	Corresponding list of new variable names.

Details

Assign the result to the data frame of interest, which can be the same data frame that contains the variables to rename.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

recode.

Examples

```
d <- Read("Mach4", quiet=TRUE)
names(d)

# single name change
d <- rename(d, m03, third)
names(d)

# vector of name changes
d <- rename(d, c(m01, m19), c(first, nineteen))
names(d)
```

`rescale` *Rescale a Variable*

Description

Rescale a variable to either z-scores with a mean of 0 and standard deviation of 1, normalized with a minimum of 0 and a maximum of 1, or to a variable computed like a z-score except use the median in place of the mean and the IQR in place of the standard deviation.

Usage

```
rescale(x, data=d, kind="z", digits_d=3)
```

Arguments

`x`	Variable to rescale.
`data`	Data frame that contains `x`.
`kind`	Type of rescaling.
`digits_d`	Number of significant digits.

Details

The default rescaling is standardization to z-scores, explicit with `kind` set to `"z"`, or just centering about the mean with `"center"`. For the min-max normalization to a range from 0 to 1, set `kind` to `"0to1"`. For the robust equivalent of standardization, set `kind` to `"robust"`.

If `x` is a vector in the global environment, then set `data` to `NULL`.

Value

The rescaled data.

Author(s)

David W. Gerbing (Portland State University; <`gerbing@pdx.edu`>)

See Also

`scale`.

Examples

```
# z-score for m01
d <- Read("Employee")

d[, .("Salary")]
x <- rescale(Salary)
x
```

reshape_long *Reshape a Wide-Form Data Frame to Long-Form*

Description

A simple wrapper for Base R reshape with sensible parameter names and sensible defaults, and able to specify a range of variables to transform.

Usage

```
reshape_long(data, transform, group="Group", response="Response", ID="ID",
           prefix=ID, sep="")
```

Arguments

data	Data frame that contains the variables to reshape.
transform	The wide-form column variable names to transform to a long-form single column.
group	Name of the grouping variable in the new long-form column.
response	Name of the variable of the response values in the new long-form column.
ID	Name of the newly created ID field in the new long-form column, the original row number from the wide-form. If NULL, the default value, then not created.
prefix	The prefix added to the value of ID for each row of data.
sep	Any potential separator of the ID prefix from the given value of the ID.

Details

reshape_long takes the transform variables in the wide-form from which it creates three new columns, group, response, and ID.

The correspondence between the original reshape parameter names and the reshape_long parameter names is shown in the following table.

reshape	reshape_long
varying	transform
v.names	response
timevar	group
times	transform
idvar	ID

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

reshape.

Examples

```
d <- Read("Anova_rb")

# with the default variable names in the long-form
reshape_long(d, c("sup1", "sup2", "sup3", "sup4"))

# with a variable range and custom variable names in the long-form
reshape_long(d, sup1:sup4, group="Supplement", response="Reps", ID="Person")
```

---

reshape_wide   *Reshape a Wide-Form Data Frame to Wide-Form*

Description

A simple wrapper for Base R reshape with sensible parameter names and sensible defaults, and able to specify a range of variables to transform.

Usage

```
reshape_wide(data, group, response, ID, prefix=NULL, sep="_")
```

Arguments

data	Data frame that contains the variables to analyze wide-form single column.
group	Name of the grouping variable in the input long-form column.
response	Name of the variable of the response values in the input long-form column.
ID	Name of the ID field in the long-form column.
prefix	If TRUE, prefix the column names in the wide form of each corresponding level of the group variable with the name of the response. Unless the values of group are numeric, the default is FALSE, just using the level names as the column names.
sep	If prefix is TRUE, the separator between the name of the level and the name of the response variable, with default "_".

Details

reshape_wide takes the variables in the long-form group, response, and ID and transforms to wide form. All other variables are deleted in the transformed data frame.

Here is the correspondence between the original reshape parameter names and the reshape_wide parameter names.

reshape	reshape_wide
v.names	response
timevar	group
idvar	ID

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

reshape.

Examples

```
d <- Read("Anova_rb")  # already in wide-form
dl <- reshape_long(d, sup1:sup4)  # convert to long-form

# convert back to wide form
reshape_wide(dl, group="Group", response="Response", ID="Person")

# with the name of the response prefixed to the column names
reshape_wide(dl, group="Group", response="Response", ID="Person",
             prefix=TRUE, sep=".")
```

---

see — *View the Upper and Left Corners of a Data Frame*

Description

Useful for large data frame. View the top-left corner of the specified data frame and the bottom-right corner of the data frame.

Usage

```
see(data, n_row=min(nrow(data), 5), n_col=min(ncol(data), 8))
```

Arguments

data	Name of the data frame to view.
n_row	Number of rows to view.
n_col	Number of columns to view.

Details

For the specified number of rows and columns, just view the subset of the data frame in terms of the top-left and the bottom-right.

Value

The subset data frame.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Extract.

Examples

```
d <- Read("Employee", quiet=TRUE)

# view the default top-left and bottom-right four rows and eight columns
see(d)

# view the top-left two rows and bottom-right four columns
see(d, n_row=2, n_col=2)
```

---

showColors *Display All Named R Colors and Corresponding rgb Values*

Description

For each specified color, displays the color, the name and the associated rgb definition.

Usage

```
showColors(file="colors.pdf", color=NULL)
```

Arguments

file	Name of pdf file that contains the list of colors with a default of colors.pdf.
color	NULL for all colors, otherwise specify a color and all colors which include that color as part of their name are displayed.

Details

Every color name is defined in terms of a red, a green and a blue component. This function lists the rgb definitions for the specified colors, as well as the name and a display of each color_ The output should be routed to an external pdf file for storage. The directory and file name of the output file are displayed.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# all colors
#showColors()

# all colors with 'blue' in their name
#showColors(file="theblues.pdf", color="blue")
```

`showPalettes` *Display Color Palettes*

Description

For each specified set of palettes display each in the set.

Usage

```
showPalettes(palette="hcl", n=12, border="transparent", file=NULL)
```

Arguments

`palette`	Name of the palette.
`n`	Number of colors per palette with a default of 12.
`border`	Border between intervals. By default is off.
`file`	Name of pdf file that contains the list of colors with a default of the name of the palette. Default is name of palette with a `.pdf` filetype.

Details

Available palettes are `"hcl"` for sequential palettes for each of 12 hues across the hcl color wheel in 30 degree intervals plus the qualitative scale of different hues and grayscale, `"viridis"`, and `"wesanderson"`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# all hcl palettes based on each  hue from 30 degrees of the color wheel,
#   including "colors" and "grays"
# default is 12 colors per palette
#showPalettes()

# viridis palate, simulate continuity
#showPalettes("viridis", n=500, border="off")
```

simCImean *Pedagogical Simulation for the Confidence Interval of the Mean*

Description

Show a sequence of confidence intervals, all calculated from repeated samples of simulated data from the same normal population, and show which intervals contain the true population mean.

Usage

```
simCImean(ns, n, mu=0, sigma=1, cl=0.95, seed=NULL,
         show_data=FALSE, show_title=TRUE,
         miss_only=FALSE, color_hit="gray40", color_miss="red",
         grid="grey90", ylim_bound=NULL, pause=FALSE,
         main=NULL, pdf_file=NULL, width=5, height=5, ...)
```

Arguments

`ns`	Number of samples, that is, repetitions of the experiment.
`n`	Size of each sample.
`mu`	Population mean.
`sigma`	Population standard deviation.
`cl`	Confidence level.
`seed`	Default seed is the R default. Enter a positive integer value to obtain a reproducible result, the same result for the same seed.
`show_data`	Plot the data for each sample over the confidence interval.
`show_title`	Place a title on the graph that contains the parameter values_
`miss_only`	For the text output, only display information for samples that missed the mean.
`color_hit`	Color of the confidence intervals that contains the mean.
`color_miss`	Color of the confidence intervals that miss the mean.
`grid`	Color of the grid lines.
`ylim_bound`	Specify the maximum deviation of the mean in either direction for the extent of the vertical axis_
`pause`	Build the graph and the text output, pausing after each confidence interval.
`main`	Title of graph.
`pdf_file`	Name of optional pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values.

Details

Simulate random normal data and display the resulting confidence intervals, with or without the data overlaid on each confidence interval. Highlight confidence intervals that miss the underlying population mean.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# 25 confidence intervals with a sample size each of 100
# mu=0, sigma=1, that is, sample from the standard normal
simCImean(25, 100)

# set the seed for a reproducible result with the same seed
simCImean(25, 100, seed=43)

# 25 confidence intervals with a sample size each of 100
# mu=100, sigma=15
# overlay the data over each confidence interval
simCImean(25, 100, mu=100, sigma=15, show_data=TRUE)
```

---

simCLT *Pedagogical Simulation for the Central Limit Theorem*

Description

Show the distribution of sample means and relevant summary statistics, such as the 95% range of variation. Provide a plot of both the specified population and the corresponding distribution of sample means.

Usage

```
simCLT(ns, n, p1=0, p2=1, seed=NULL,
       type=c("normal", "uniform", "lognormal", "antinormal"),
       fill="lightsteelblue3", n_display=0, digits_d=3,
       subtitle=TRUE, pop=TRUE,
       main=NULL, pdf=FALSE, width=5, height=5, ...)
```

Arguments

ns	Number of samples, that is, repetitions of the experiment.
n	Size of each sample.
p1	First parameter value for the population distribution, the mean, minimum or meanlog for the normal, uniform, and lognormal populations, respectively. Must be 0, the minimum, for the anti-normal distribution.

`p2`	Second parameter value for the population distribution, the standard deviation, maximum or sdlog for the normal, uniform and lognormal populations, respectively. Is the maximum for the anti-normal, usually left at the default value of 1.
`seed`	Default seed is the R default. Enter a positive integer value to obtain a reproducible result, the same result for the same seed.
`type`	The general population distribution.
`fill`	Fill color of the graphs.
`n_display`	Number of samples for which to display the sample mean and data values.
`digits_d`	Number of decimal digits to display on the output.
`subtitle`	If `TRUE`, then display the specific parameter values of the population or sample, depending on the graph.
`pop`	If `TRUE`, then display the graph of the population from which the data are sampled.
`main`	Title of graph.
`pdf`	Indicator as to if the graphic files should be saved as pdf files instead of directed to the standard graphics windows.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values for R function `lm` which provides the core computations.

Details

Provide a plot of both the specified population and the corresponding distribution of sample means. Include descriptive statistics including the 95% range of sampling variation in raw units and standard errors for comparison to the normal distribution. Also provide a few samples of the data and corresponding means.

Four different populations are provided: normal, uniform, lognormal for a skewed distribution, and what is called the anti-normal, the combining of two side-by-side triangular distributions so that most of the values are in the extremes and fewer values are close to the middle.

For the lognormal distribution, increase the skew by increasing the value of `p2`, which is the population standard deviation.

The anti-normal distribution requires the `triangle` package. No population mean and standard deviation are provided for the anti-normal distribution, so the 95% range of sampling variable of the sample mean in terms of standard errors is not provided. ** Not activated until the triangle package is updated. **

If the two plots, of the population and sample distributions respectively, are written to pdf files, according to `pdf=TRUE`, they are named SimPopulation.pdf and SimSample.pdf. Their names and the directory to which they are written are provided as part the console output.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# plot of the standardized normal
#  and corresponding sampling distribution with 10000 samples
#  each of size 2
simCLT(ns=1000, n=2)

# plot of the uniform dist from 0 to 4
#  and corresponding sampling distribution with 10000 samples
#  each of size 2
simCLT(ns=1000, n=2, p1=0, p2=4, type="uniform", bin_width=0.01)

# save the population and sample distributions to pdf files
# simCLT(100, 10, pdf=TRUE)
```

---

simFlips *Pedagogical Binomial Simulation, Coin flips*

Description

Simulate a sequence of coin flips.

Usage

```
simFlips(n, prob=.5, seed=NULL,
          show_title=TRUE, show_flips=TRUE,
          grid="grey90", pause=FALSE,
          main=NULL, pdf_file=NULL, width=5, height=5, ...)
```

Arguments

`n`	Size of each sample, that is, the number of trials or flips.
`prob`	Probability of a success on any one trial.
`seed`	Default seed is the R default. Enter a positive integer value to obtain a reproducible result, the same result for the same seed.
`show_title`	Place a title on the graph that contains the parameter values_
`show_flips`	Plot the outcome of each flip.
`grid`	Color of the grid lines.
`pause`	Build the graph and the text output, pausing after each confidence interval.
`main`	Title of graph.

pdf_file	Name of the pdf file to which graphics are redirected.
width	Width of the pdf file in inches.
height	Height of the pdf file in inches.
...	Other parameter values.

Details

Generate and plot successive values of a Head or a Tail using standard R rbinom function.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# 10 flips of a fair coin
simFlips(10, .5)

# set the seed for a reproducible result with the same seed
simFlips(10, .5, seed=43)
```

---

simMeans *Pedagogical Simulation of Sample Means over Repeated Samples*

Description

Show a sequence of sample means and data, all simulated from the same normal population. Useful for developing an intuition for developing an informal confidence interval, that is, specifying a likely range of values that contain the true population mean, but without a formal probability.

Usage

```
simMeans(ns, n, mu=0, sigma=1, seed=NULL,
         show_title=TRUE, show_data=TRUE, max_data=10,
         grid="grey90", ylim_bound=NULL, pause=FALSE,
         sort=NULL, set_mu=FALSE, digits_d=2,
         main=NULL, pdf_file=NULL, width=5, height=5, ...)
```

Arguments

ns	Number of samples, that is, repetitions of the experiment.
n	Size of each sample.
mu	Population mean.
sigma	Population standard deviation.

`seed`	Default seed is the R default. Enter a positive integer value to obtain a reproducible result, the same result for the same seed.
`show_title`	Place a title on the graph that contains the parameter values_
`show_data`	Show the data values on the text output.
`max_data`	Maximum number of data values per sample on the text output.
`grid`	Color of the grid lines.
`ylim_bound`	Specify the maximum deviation of the mean in either direction for the extent of the vertical axis_
`pause`	Build the graph and the text output sample by sample.
`sort`	Sort the output by the means in ascending order. By default is `TRUE` unless `se.mu` or `pause` is `TRUE`.
`set_mu`	Have the program randomly set `mu` and `sigma`, usually to guess the correct value.
`digits_d`	Sort the output by the means in ascending order.
`main`	Title of graph.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values.

Details

Simulate random normal data and display the resulting sample means, both as text output and graphic output.

If `pause=TRUE`, then the true population values are not revealed as the simulation progresses. These values are saved in the user's workspace and can be revealed by entering their names at the user prompt, `mu` and `sigma`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
# 8 samples, each with a sample size of 10
# mu=0, sigma=1, that is, sample from the standard normal
simMeans(8, 10)

# 25 sample means with a sample size each of 100
# mu=100, sigma=15
# pause after each interval and show the data
simMeans(25, 100, mu=100, sigma=15, show_data=FALSE)
```

`skew` *Skew of a variable.*

Description

The Fisher-Pearson standardized moment coefficient adjusted for sample size.

Usage

```
skew(x, na.rm=TRUE)
```

Arguments

`x`	Variable from which to compute skewness.
`na.rm`	A logical value indicating whether NA values should be removed before the computation proceeds.

Details

`G1`, the adjusted Fisher-Pearson standardized moment coefficient. The adjustment is the sample size `n` divided by the product of `n-1` and `n-2`.

The core component of the skewness expression is for each data value calculate, standardize the value, then raise the standardized value to the third power. The component is the sum of these cubics.

Value

Skew.

Author(s)

David W. Gerbing (Portland State University; `<gerbing@pdx.edu>`)

References

Doane, D. P. & Seward, L. E. (2011). Measuring Skewness: A Forgotten Statistic?,Journal of Statistics Education, 19(2), 1-18. URL: https://doi.org/10.1080/10691898.2011.11889611.

Examples

```
x <- rnorm(100)
skew(x)
```

`sort_by` *sort_by the Rows of a Data Frame*

Description

Sorts the values of a data frame according to the values of one or more variables contained in the data frame, or the row names. Variable types include numeric and factor variables. Factors are sorted by the ordering of their values, which, by default is alphabetical. Sorting by row names is also possible.

Usage

```
sort_by(data=d, by, direction=NULL, quiet=getOption("quiet"), ...)

Sort(...)
```

Arguments

`data`	The name of the data frame from which to create the subset, which is `d` by default.
`by`	One or more variables to be sorted, or just the character string `row.names` or `random`.
`direction`	Default is ascending for all variables listed in `by`. Or, specify a list of `"+"` for ascending and `"-"` for descending, one for each variable to be sorted.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	Other parameter values.

Details

`sort_by` sorts the rows of a data frame and lists the first five rows of the sorted data frame. Specify the values upon which to base the sort with the required by parameter. If not all sorted variables are sorted in ascending order, then also specify a sequence of `"+"` for ascending and `"-"` for descending, respectively, one for each variable to be sorted. If `row.names` or `random` is specified, then no other variables can be specified.

A list of consecutive variables can be specified using the colon notation, such as Years:Salary To specify a list of multiple variables, or `"+"` and `"-"` signs, or sets of variables, separate each set of variables or each sign by a comma, then invoke the R combine or `c` function. For example, if three variables are to be sorted, the first two ascending and the last descending, then specify, `direction=c("+","+","-")`.

`sort_by` is based on the standard R function `order`, though the `sort_by` function allows for the sorting of factors, whereas `order` does not.

Value

The sorted data frame is returned, usually assigned the name of `d` as in the examples below. This is the default name for the data frame input into the `lessR` data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

order.

Examples

```
# construct data frame
d <- read.table(text="Severity Description
1 Mild
4 Moderate
3 Moderate
2 Mild
1 Severe", header=TRUE)

# sort the data frame called d according to Severity
#   in ascending order
d <- sort_by(d, Severity)

# sort Description in descending order, sort Severity within
#  each level of Description in ascending order
d <- sort_by(d, c(Description, Severity), direction=c("-", "+"))

# sort by row names in ascending order
d <- sort_by(d, row.names)

# randomly re-shuffle the rows of data
d <- sort_by(d, random)
```

---

style *Set the Default Color Theme and Other System Settings*

Description

Deprecated Names: set, theme

The color and style attributes of each plot can be set as a general theme, or individually set from the following list of attributes. For convenience, groups of these attributes are specified to define color themes, plus style sub-themes that apply to any theme, with default values: theme="colors" and sub_theme="default". To reset to the default theme: style().

Usage

```
style(
  theme=c("colors", "lightbronze", "dodgerblue", "darkred", "gray",
      "gold", "darkgreen", "blue", "red", "rose", "slatered", "green",
      "purple", "sienna", "brown", "orange", "white", "light"),
```

```
sub_theme=c("default", "black", "wsj"),
set=NULL, get=FALSE, reset=TRUE,

window_fill=getOption("window_fill"),
panel_fill=getOption("panel_fill"),
panel_color=getOption("panel_color"),
panel_lwd=getOption("panel_lwd"),
panel_lty=getOption("panel_lty"),

fill=NULL,
bar_fill=getOption("bar_fill"),
bar_fill_discrete=getOption("bar_fill_discrete"),
bar_fill_cont=getOption("bar_fill_cont"),
trans=NULL,
trans_bar_fill=getOption("trans_bar_fill"),
color=NULL,
bar_color=getOption("bar_color"),
bar_color_cont=getOption("bar_color_cont"),
bar_color_discrete=getOption("bar_color_discrete"),

labels=getOption("labels"),
labels_color=getOption("labels_color"),
labels_size=getOption("labels_size"),
labels_digits=getOption("labels_digits"),
labels_position=getOption("labels_position"),

pt_fill=getOption("pt_fill"),
trans_pt_fill=getOption("trans_pt_fill"),
pt_color=getOption("pt_color"),
se_fill=getOption("se_fill"),
ellipse_fill=getOption("ellipse_fill"),
ellipse_color=getOption("ellipse_color"),
ellipse_lwd=getOption("ellipse_lwd"),
fit_color=getOption("fit_color"),
fit_lwd=getOption("fit_lwd"),
bubble_text_color=getOption("bubble_text_color"),
segment_color=getOption("segment_color"),
ID_color=getOption("ID_color"),
out_fill=getOption("out_fill"),
out_color=getOption("out_color"),
out2_fill=getOption("out2_fill"),
out2_color=getOption("out2_color"),

violin_fill=getOption("violin_fill"),
violin_color=getOption("violin_color"),
box_fill=getOption("box_fill"),
box_color=getOption("box_color"),
```

```
axis_color=getOption("axis_color"),
axis_x_color=getOption("axis_x_color"),
axis_y_color=getOption("axis_y_color"),
axis_lwd=getOption("axis_lwd"),
axis_x_lwd=getOption("axis_x_lwd"),
axis_y_lwd=getOption("axis_y_lwd"),
axis_lty=getOption("axis_lty"),
axis_x_lty=getOption("axis_x_lty"),
axis_y_lty=getOption("axis_y_lty"),
axis_cex=getOption("axis_cex"),
axis_x_cex=getOption("axis_x_cex"),
axis_y_cex=getOption("axis_y_cex"),
axis_text_color=getOption("axis_text_color"),
axis_x_text_color=getOption("axis_x_text_color"),
axis_y_text_color=getOption("axis_y_text_color"),
rotate_x=getOption("rotate_x"),
rotate_y=getOption("rotate_y"),
offset=getOption("offset"),

lab_color=getOption("lab_color"),
lab_x_color=getOption("lab_x_color"),
lab_y_color=getOption("lab_y_color"),
lab_cex=getOption("lab_cex"),
lab_x_cex=getOption("lab_x_cex"),
lab_y_cex=getOption("lab_y_cex"),
main_color=getOption("main_color"),
main_cex=getOption("main_cex"),

grid_color=getOption("grid_color"),
grid_x_color=getOption("grid_x_color"),
grid_y_color=getOption("grid_y_color"),
grid_lwd=getOption("grid_lwd"),
grid_x_lwd=getOption("grid_x_lwd"),
grid_y_lwd=getOption("grid_y_lwd"),
grid_lty=getOption("grid_lty"),
grid_x_lty=getOption("grid_x_lty"),
grid_y_lty=getOption("grid_y_lty"),

strip_fill=getOption("strip_fill"),
strip_color=getOption("strip_color"),
strip_text_color=getOption("strip_text_color"),

add_fill=getOption("add_fill"),
add_trans=getOption("add_trans"),
add_color=getOption("add_color"),
add_cex=getOption("add_cex"),
add_lwd=getOption("add_lwd"),
add_lty=getOption("add_lty"),
```

```
  n_cat=getOption("n_cat"), suggest=getOption("suggest"),
  notes=getOption("notes"),
  quiet=getOption("quiet"), brief=getOption("brief"),

  results=getOption("results"), explain=getOption("explain"),
  interpret=getOption("interpret"), document=getOption("document"),
  code=getOption("code"),

  width=120, show=FALSE, ...)

set(...)
```

Arguments

`theme` The specified color scheme. If specified, re-sets all style attributes to the values consistent with that theme.

`sub_theme` Further modification of the main themes.

`set` A list of parameter values, a theme, that was previously saved, and now is read back to become the current set of parameter values. See the examples.

`get` Save the current list of parameter values, a theme, into an R object.

`reset` Change one or more settings or the entire theme.

`window_fill` Fill color of the entire device window.

`panel_fill` Color of the plot background.

`panel_color` Color of border around the plot background, the box, that encloses the plot, with a default of `"black"`.

`panel_lwd` Line width of the box around the plot.

`panel_lty` Line type of the box around the plot. Acceptable values are `"blank"`, `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, and `"longdash"`.

`fill` Color of a filled region – bars, points and bubbles – depending on the objected plotted. Can explicitly choose `"grays"` or `"hues"` , or pre-specified R color schemes `"rainbow"`, `"terrain"`, and `"heat"`. Can also provide pre-defined color ranges `"blues"`, `"reds"` and `"greens"`, as well as custom colors, such as generated by `getColors`

`bar_fill` Color of a filled bar, bubble or box.

`bar_fill_discrete` Color of a filled bar chart bar or pie chart slice.

`bar_fill_cont` Color of a filled histogram bar.

`trans` Transparency of a filled bar, rectangular region, or points from 0 (none) to 1 (complete).

`trans_bar_fill` The transparency of a filled bar or rectangular region, such as a histogram bar or the box in a box plot. Value from 0 to 1, opaque to transparent.

`color` Color of a line segment such as the border of bar or point. Can explicitly choose "grays" or "hues", or pre-specified R color schemes "rainbow", "terrain", and "heat". Can also provide pre-defined color ranges "blues", "reds" and "greens", as well as custom colors, such as generated by getColors

`bar_color` Color of the border of a filled region such as a histogram bar.

`bar_color_discrete` Color of the border of a filled region for values on a qualitative scale.

`bar_color_cont` Color of the border of a filled region for values on a quantitative scale, such as a histogram bar.

`labels` If not the default value of "off", adds the numerical results to the plot according to "%", "prop" or "input", that is, percentages, proportions, or the value from which the slices are plotted, such as tabulated counts if y is not specified, or the value of y if the plotted values are provided. If any other labels parameter is specified, default is set to "%".

`labels_color` Color of the plotted text. Could be a vector to specify a unique color for each value. If fewer colors are specified than the number of categories, the colors are recycled.

`labels_size` Character expansion factor, the size, of the plotted text, for which the default value is 0.95.

`labels_digits` Number of decimal digits for which to display the labels. Default is 0, round to the nearest integer, for "%" and 2 for "prop".

`labels_position` Position of the plotted text. Default is inside the pie, or, if "label", as part of the label for each value outside of the pie.

`pt_fill` Color of a filled plotted point.

`trans_pt_fill` The transparency of the inner region of a plotted point. Value from 0 to 1, opaque to transparent.

`pt_color` Color of a line or outline of a filled region, such as the border of a plotted point.

`se_fill` Color of the fill for the standard error plot about a fit line in a scatter plot.

`ellipse_fill` Color of the fill for an ellipse in a scatter plot.

`ellipse_color` Color of the border for an ellipse in a scatter plot.

`ellipse_lwd` Line width of the border for an ellipse in a scatter plot.

`fit_color` Color of the fit line in a scatter plot.

`fit_lwd` Width of fit line. By default is 2 for Windows and 1.5 for Mac.

`bubble_text_color` Color of the displayed text regarding the size of a bubble, either a tabulated frequency for categorical variables, or the value of a third variable according to size.

`segment_color` Color of connecting line segments when there are also plotted points, such as in a frequency polygon. Default color is color.

`ID_color` Color of the text to display the ID labels.

`out_fill`	For a scatterplot, color of the border of potential outliers, which, for the unadjusted boxplot, are default values 1.5 IQR's beyond the lower or upper quartile.
`out_color`	For a scatterplot, color of potential outliers.
`out2_fill`	For a scatterplot, color of extreme outliers, which, for the unadjusted boxplot, are default values 3 IQR's beyond the lower or upper quartile.
`out2_color`	For a scatterplot, color of the border of extreme outliers.
`violin_fill`	Fill color for a **violin plot.**
`violin_color`	Border color for the violin in a violin plot.
`box_fill`	Fill color for a box plot.
`box_color`	Border color of a box in a box plot.
`axis_color`	Color of the axes.
`axis_x_color`	Color of the `x`-axis_
`axis_y_color`	Color of the `y`-axis_
`axis_lwd`	Line width of axes.
`axis_x_lwd`	Line width of horizontal axis_
`axis_y_lwd`	Line width of vertical axis_
`axis_lty`	Line type of axes, either "solid", "dashed", "dotted", "dotdash", "longdash", "twodash", or "blank".
`axis_x_lty`	Line type of horizontal axis_
`axis_y_lty`	Line type of vertical axis_
`axis_cex`	Scale magnification factor, which by defaults displays the axis values to be smaller than the axis labels. Provides the functionality of, and can be replaced by, the standard R `cex.axis`.
`axis_x_cex`	Scale magnification factor for the `x`-axis_
`axis_y_cex`	Scale magnification factor for the `y`-axis_
`axis_text_color`	Color of the font used to label the axis values.
`axis_x_text_color`	Color of the font used to label the `x`-axis values.
`axis_y_text_color`	Color of the font used to label the `y`-axis values.
`rotate_x`	Degrees that the `x`-axis values are rotated, usually to accommodate longer values, typically used in conjunction with `offset`.
`rotate_y`	Degrees that the `y`-axis values are rotated.
`offset`	The spacing between the axis values and the axis_ Default is 0.5. Larger values such as 1.0 are used to create space for the label when longer axis value names are rotated.
`lab_color`	Color of the axis labels.

lab_x_color Color of the axis labels on the horizontal axis_

lab_y_color Color of the axis labels on the vertical axis_

lab_cex Size of labels for x and y axes.

lab_x_cex Size of labels for x.

lab_y_cex Size of labels for y.

main_color Color of the title.

main_cex Size of the title font.

grid_color Color of the grid lines.

grid_x_color Color of the grid lines for the x-axis_

grid_y_color Color of the grid lines for the y-axis_

grid_lwd Width of grid lines.

grid_x_lwd Width of vertical grid lines, inherits from grid_lwd.

grid_y_lwd Width of horizontal grid lines, inherits from grid_lwd.

grid_lty Line type for grid lines: "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash", or "blank".

grid_x_lty Line-type of vertical grid lines, inherits from grid_lty.

grid_y_lty Line-type of horizontal grid lines, inherits from grid_lty.

strip_fill Fill color for the **strip that labels each panel** in a Trellis plot.

strip_color Border color for the strip that labels each panel in a Trellis plot.

strip_text_color
Color of the label in each strip of a Trellis plot.

add_fill Interior fill color of added object. Can explicitly choose "grays" or "hues", or pre-specified R color schemes "rainbow", "terrain", and "heat". Can also provide pre-defined color ranges "blues", "reds" and "greens", as well as custom colors, such as generated by getColors

add_trans Transparency level of color or fill, which ever is applicable from 0 (opaque) to 1 (transparent).

add_color Color of borders and lines of added object.

add_cex Text expansion factor, relative to 1. As with the following properties, can be a vector for multiple placement or objects.

add_lwd Line width of added object.

add_lty Line type of added object. See panel_lty for types.

n_cat Number of categories that specifies the largest number of unique equally-spaced values of variable of a numeric data type for which the variable will be analyzed as categorical. Default value is 0. *[deprecated]*: Best to convert a categorical integer variable to a factor.

suggest If TRUE, then provide suggestions for alternative analyses.

`notes`	If `TRUE`, then provide notes.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`brief`	If set to `TRUE`, reduced text output.
`results`	For the R markdown file generated by the Rmd option, show the results.
`explain`	For the R markdown file generated by the Rmd option, explain the results.
`interpret`	For the R markdown file generated by the Rmd option, interpret the results.
`document`	For the R markdown file generated by the Rmd option, documents the code that generated the results.
`code`	For the R markdown file generated by the Rmd option, shows the code that generated the results.
`width`	Maximum width of each line displayed at the console, just accesses the standard R options function for width.
`show`	Option for showing all settings.
`...`	Parameter values.

Details

OVERVIEW
Sets the default color palette via the R `options` statement, as well as the transparency of plotted bars and points and other non-color characteristics such as the color of the grid lines. For convenience, groups of attributes are organized into themes and sub-themes. When the `theme` is specified, *all* options are reset to their default values. All other modifications, with individual parameters or grouped parameters as a sub-theme, are cumulative. For example, one sub-theme can be followed by another, as well as the specifications of individual attributes. Calling the function with no arguments sets to the default style.

Available themes:
"lightbronze" [default]
"dodgerblue" [default `lessR` 3.6.0 and earlier]
"darkred"
"gray"
"gold"
"darkgreen"
"blue"
"red"
"rose"
"green"
"purple"
"sienna"
"brown"
"orange"
"white"
"light"

The "gray" color theme is based on the colors used in Hadley Wickham's ggplot2 package. The "lightbronze" theme, especially with the wsj sub-theme, is based on Jeffrey Arnold's wsj theme from his ggthemes package.

SUB-THEMES
"black": Black background of the entire device window with translucent fill colors from the current theme. "wsj": Similar to the wsj theme from the ggthemes package, especially with the theme of "lightbronze". The y-axis is removed with though the value labels retained, the vertical grid is removed, and the horizontal grid is dotted and thicker than the default.

Value

The current settings can optionally be saved into an R object, and then read back at a later time with the set function.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Arnold, Jeffrey B., (2017), ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'. R package version 3.4.0. https://CRAN.R-project.org/package=ggthemes

Gerbing, D. W. (2020). R Visualizations: Derive Meaning from Data, Chapter 10, NY: CRC Press.

Wickham, Hadley, (2009), ggplot2: Elegant Graphics for Data Analysis, 2nd edition, Springer.

See Also

options.

Examples

```
# some data
d <- rd("Employee", quiet=TRUE)

# gold colors embedded in a black background
style("gold", sub_theme="black")
Plot(Years, Salary, size=0, ellipse=seq(.1,.9,.1))

# three ways to do gray scale
style(window_fill="white")
# 1. gray scale with a light gray background
style("gray")
# 2. gray scale with a dark, almost black, background
style("gray", sub_theme="black")
# 3. mostly black and white
style("white")

# reset style to the default "colors"
style()
```

```
# set bar fill to qualitative hcl colors
# here also turn off bar borders and set to a mild transparency
Histogram(Salary, fill="greens", color="off")
# same as
# style(bar_fill_cont="greens", bar_color="off")
# Histogram(Salary)

# set bar fill to 6 blue colors
# for continuous band explicitly call getColors and specify n
#   to obtain the full spectrum, such as for analysis of Likert
#   scale responses with six possible responses per item
style(bar_fill=getColors("blues", n=6))

# adjust Trellis strip to a dark background
style(strip_fill="gray60", strip_color="gray20",
      strip_text_color=rgb(247,242,230, maxColorValue=255))
Plot(Years, Salary, by1=Gender)

# define a custom style beyond just colors
style(panel_fill="off", panel_color="off",
      window_fill=rgb(247,242,230, maxColorValue=255),
      pt_fill="black", trans=0,
      lab_color="black", axis_text_color="black",
      axis_y_color="off",
      grid_x_color="off", grid_y_color="black",  grid_lty="dotted", grid_lwd=1)
hs(Salary)

# save the current theme settings into an R object without changes
# unless set to FALSE, get is always TRUE, for all calls to style
mystyle <- style(get=TRUE)
# ...  bunch of changes
# then recall older settings to current theme setting
style(set=mystyle)

# create a gray-scale with a sub-theme of wsj
# save, and then at a later session read back in
grayWSJ <- style("gray", sub_theme="wsj")
# Write(grayWSJ, "grayWSJ", format="R")
# ...
#mystyle <- Read("grayWSJ.rda")  # read grayWSJ.rda
#style(set=mystyle)



# all numeric variables with 8 or less unique values and equally spaced
#  intervals are analyzed as categorical variables
style(n_cat=8)
```

Subset *Subset the Values of One or More Variables*

Description

Abbreviation: `subs`

Deprecated, use `.` instead in conjunction with base R `link{Extract}`.

Based directly on the standard R `subset` function to only include or exclude specified rows or data, and for specified columns of data. Output provides feedback and guidance regarding the specified subset operations. Rows of data may be randomly extracted, and also with the code provided to generate a hold out validation sample created. The hold out sample is created from the original data frame, usually named d, so the subset data frame must be directed to a data frame with a new name or the data re-read to construct the holdout sample. Any existing variable labels are retained in the subset data frame.

Usage

```
Subset(rows, columns, data=d, holdout=FALSE,
       random=0, quiet=getOption("quiet"), ...)
```

Arguments

`rows`	Specify the rows, i.e., observations, to be included or deleted, such as with a logical expression or by direct specification of the numbers of the corresponding rows of data.
`columns`	Specify the columns, i.e., variables, to be included or deleted.
`data`	The name of the data frame from which to create the subset, which is `d` by default.
`holdout`	Create a hold out sample for validation if rows is a proportion or an integer to indicate random extraction of rows of data.
`random`	If an integer or proportion, specifies number of rows to data to randomly extract.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	The list of variables, each of the form, `variable = equation`. Each `variable` can be the name of an existing variable in the data frame or a newly created variable.

Details

`Subset` creates a subset data frame based on one or more rows of data and one or more variables in the input data frame, and lists the first five rows of the revised data frame. Guidance and feedback regarding the subsets are provided by default. The first five lines of the input data frame are listed before the subset operation, followed by the first five lines of the output data frame.

The argument `rows` can be a logical expression based on values of the variables, or it can be an integer or proportion to indicate random extraction of rows. An integer specifies the number of rows to retain, and a proportion specifies the corresponding proportion, which is then rounded to an integer. If `holdout=TRUE`, then the code to create a hold out data frame with a subsequent `Subset` analysis is also created. Copy and run this code on the original data frame to create the hold out sample.

To indicate retaining an observation, specify at least one variable name and the value of the variable for which to retain the corresponding observations, using two equal signs to indicate the logical equality. If no rows are specified, all rows are retained. Use the base R row.names function to identify rows by their row names, as illustrated in the examples below.

To indicate retaining a variable, specify at least one variable name. To specify multiple variables, separate adjacent variables by a comma, and enclose the list within the standard R combine function, c. A single variable may be replaced by a range of consecutive variables indicated by a colon, which separates the first and last variables of the range. To delete a variable or variables, put a minus sign, -, in front of the c.

Value

The subset of the data frame is returned, usually assigned the name of d as in the examples below. This is the default name for the data frame input into the lessR data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

subset, factor.

Examples

```
# construct data frame
d <- read.table(text="Severity Description
1 Mild
4 Moderate
3 Moderate
2 Mild
1 Severe", header=TRUE)

# only include those with a value of Moderate for Description
d <- Subset(rows=Description=="Moderate")

# locate, that is, display only, the 2nd and 4th rows of data
Subset(row.names(d)=="2" | row.names(d)=="4")

# retain only the first and fourth rows of data, store in myd
myd <- Subset(c(1,4))

# delete only the first and fourth rows of data, store in myd
myd <- Subset(-c(1,4))

# built-in data table warpbreaks has several levels of wool
#   and breaks plus continuous measure tension
# retain only the A level of wool and the L level of tension,
#   and the one variable breaks
d <- Subset(wool=="A" & tension=="L", columns=breaks, data=warpbreaks)

# delete Years and Salary
d <- Read("Employee", quiet=TRUE)
```

```
d <- Subset(columns=-c(Years, Salary))

# locate, display only, a specified row by its row.name
d <- Read("Employee", quiet=TRUE)
Subset(row.names(d)=="Fulton, Scott")

# randomly extract 60% of the data
# generate code to create the hold out sample of the rest
d <- Read("Employee", quiet=TRUE)
mysubset <- Subset(random=.6, holdout=TRUE)
```

SummaryStats *Summary Statistics for One or Two Variables*

Description

Abbreviation: ss

The summary statistics aspect for continuous variables is deprecated. Use pivot instead.

Descriptive or summary statistics for a numeric variable or a factor, one at a time or for all numeric and factor variables in the data frame. For a single variable, there is also an option for summary statistics at each level of a second, usually categorical variable or factor, with a relatively few number of levels. For a numeric variable, output includes the sample mean, standard deviation, skewness, kurtosis, minimum, 1st quartile, median, third quartile and maximum, as well as the number of non-missing and missing values_ For a categorical variable, the output includes the table of counts for each value of a factor, the total sample size, and the corresponding proportions.

If the provided object to analyze is a set of multiple variables, including an entire data frame, then each non-numeric variable in the data frame is analyzed and the results written to a pdf file in the current working directory. The name of each output pdf file that contains a bar chart and its path are specified in the output.

When output is assigned into an object, such as s in s <- ss(Y), the pieces of output can be accessed for later analysis. A primary such analysis is knitr for dynamic report generation in which R output embedded in documents See value below.

Usage

```
SummaryStats(x=NULL, by=NULL, data=d, rows=NULL, n_cat=getOption("n_cat"),
         digits_d=NULL, brief=getOption("brief"), label_max=20, ...)

ss_brief(..., brief=TRUE)

ss(...)
```

Arguments

x Variable(s) to analyze. Can be a single variable, either within a data frame or as a vector in the user's workspace, or multiple variables in a data frame such as designated with the c function, or an entire data frame. If not specified, then defaults to all variables in the specified data frame, d by default.

by	Applies to an analysis of a numeric variable, which is then analyzed at each level of the by variable. The variable is coerced to a factor.
data	Optional data frame that contains the variable of interest, default is d.
rows	A logical expression that specifies a subset of rows of the data frame to analyze.
n_cat	Specifies the largest number of unique values of variable of a numeric data type for which the variable will be analyzed as a categorical. Default is off, set to 0. *[deprecated]*: Best to convert a categorical integer variable to a factor.
digits_d	Specifies the number of decimal digits to display in the output.
brief	If set to TRUE, reduced text output. Can change system default with style function.
label_max	Maximum size of labels for the values of a variable. Not a literal maximum as preserving unique values may require a larger number of characters than specified.
...	Further arguments to be passed to or from methods.

Details

OVERVIEW
The by option specifies a categorical variable or factor, with a relatively few number of values called levels. The variable of interest is analyzed at each level of the factor.

The digits_d parameter specifies the number of decimal digits in the output. It must follow the formula specification when used with the formula version. By default the number of decimal digits displayed for the analysis of a variable is one more than the largest number of decimal digits in the data for that variable.

Reported outliers are based on the boxplot criterion. The determination of an outlier is based on the length of the box, which corresponds, but may not equal exactly, the interquartile range. A value is reported as an outlier if it is more than 1.5 box lengths away from the box.

Skewness is computed with the usual adjusted Fisher-Pearson standardized moment skewness coefficient, the version found in many commercial packages.

The lessR function Read reads the data from an external csv file into the data frame called d. To describe all of the variables in a data frame, invoke SummaryStats(d), or just SummaryStats(), which then defaults to the former.

In the analysis of a categorical variable, if there are more than 10 levels then an abbreviated analysis is performed, only reporting the values and the associated frequencies. If all the values are unique, then the user is prompted with a note that perhaps this is actually an ID field which should be specified using the row.names option when reading the data.

DATA
If the variable is in a data frame, the input data frame has the assumed name of d. If this data frame is named something different, then specify the name with the data option. Regardless of its name, the data frame need not be attached to reference the variable directly by its name, that is, no need to invoke the d$name notation.

To analyze each variable in the d data frame, use SummaryStats(). Or, for a data frame with a different name, insert the name between the parentheses. To analyze a subset of the variables in a data frame, specify the list with either a : or the c function, such as m01:m03 or c(m01,m02,m03).

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see Read.

ONLY VARIABLES ARE REFERENCED
The referenced variable in a lessR function can only be a variable name. This referenced variable must exist in either the referenced data frame, such as the default d, or in the user's workspace, more formally called the global environment. That is, expressions cannot be directly evaluated. For example:

> SummaryStats(rnorm(50)) # does NOT work

Instead, do the following:

```
> Y <- rnorm(50)   # create vector Y in user workspace
> SummaryStats(Y)     # directly reference Y
```

Value

The output can optionally be saved into an R object, otherwise it simply appears in the console. Redesigned in lessR version 3.3 to provide two different types of components: the pieces of readable output in character format, and a variety of statistics. The readable output are character strings such as tables amenable for reading. The statistics are numerical values amenable for further analysis. A primary motivation of these two types of output is to facilitate knitr documents, as the name of each piece can be inserted into the knitr document.

If the analysis is of a single numeric variable, the full analysis returns the following statistics: n, miss, mean, sd, skew, kurtosis, min, quartile1, median, quartile3, max, IQR. The brief analysis returns the corresponding subset of the summary statistics. If the anlaysis is conditioned on a by variable, then nothing is returned except the text output. The pieces of readable output are out_stats and out_outliers.

If the analysis is of a single categorical variable, a list is invisibly returned with two tables, the frequencies and the proportions, respectively named freq and prop. The pieces of readable output are out_title and out_stats.

If two categorical variables are analyzed, then for the full analysis four tables are returned as readable output, but no numerical statistics. The pieces are out_title, out_freq, out_prop, out_colsum, out_rowsum.

Although not typically needed, if the output is assigned to an object named, for example, s, as in s <- ss(Y), then the contents of the object can be viewed directly with the unclass function, here as unclass(s).

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

summary, formula, boxplot.

Examples

```
# -------------------------------------------
# one or two numeric or categorical variables
# -------------------------------------------

# create data frame, d, to mimic reading data with rad function
# d contains both numeric and non-numeric data
# X has two character values, Y is numeric
n <- 15
X <- sample(c("Group1","Group2"), size=n, replace=TRUE)
Y <- round(rnorm(n=n, mean=50, sd=10),3)
d <- data.frame(X,Y)
rm(X); rm(Y)

# Analyze the values of numerical Y
# Calculate n, mean, sd, skew, kurtosis, min, max, quartiles
SummaryStats(Y)
# short name
ss(Y)
# output saved for later analysis
s <- ss(Y)
# view full text output
s
# view just the outlier analysis
s$out_outliers
# list the names of all the components
names(s)

# Analyze the values of categorical X
# Calculate frequencies and proportions, totals, chi-square
SummaryStats(X)

# Only a subset of available summary statistics
ss_brief(Y)
ss_brief(X, label_max=3)

# Reference the summary stats in the object: stats
stats <- ss(Y)
my.mean <- stats$mean

# Get the summary statistics for Y at each level of X
# Specify 2 decimal digits for each statistic displayed
SummaryStats(Y, by=X, digits_d=2)


# ----------
# data frame
# ----------

# Analyze all variables in data frame d at once
# Any variables with a numeric data type and 4 or less
#  unique values will be analyzed as a categorical variable
```

```
SummaryStats()

# Analyze all variables in data frame d at once
# Any variables with a numeric data type and 7 or less
#  unique values will be analyzed as a categorical variable
SummaryStats(n_cat=7)

# analyze just a subset of a data frame
d <- Read("Employee", quiet=TRUE)
SummaryStats(c(Salary,Years))


# ----------------------------------------
# data frame different from default d
# ----------------------------------------

# variables in a data frame which is not the default d
# access the breaks variable in the R provided warpbreaks data set
# although data not attached, access the variable directly by its name
data(warpbreaks)
SummaryStats(breaks, by=wool, data=warpbreaks)

# Analyze all variables in data frame warpbreaks at once
SummaryStats(warpbreaks)


# ----------------------------
# can enter many types of data
# ----------------------------

# generate and enter integer data
X1 <- sample(1:4, size=100, replace=TRUE)
X2 <- sample(1:4, size=100, replace=TRUE)
SummaryStats(X1)
SummaryStats(X1,X2)

# generate and enter type double data
X1 <- sample(c(1,2,3,4), size=100, replace=TRUE)
X2 <- sample(c(1,2,3,4), size=100, replace=TRUE)
SummaryStats(X1)
SummaryStats(X1, by=X2)

# generate and enter character string data
# that is, without first converting to a factor
Travel <- sample(c("Bike", "Bus", "Car", "Motorcycle"), size=25, replace=TRUE)
SummaryStats(Travel)
```

`to` *Create a Sequence of Numbered Variable Names with a Common Prefix and Width*

Description

Generates sequentially numbered variable names, all starting with the same prefix, usually in conjunction with reading data values into R. The advantage over the standard R function `paste0` is that `to` maintains equal widths of the names, such as m08 instead of m8 if some values are m10 or larger up to m99.

Usage

```
to(prefix, until, from=1, same_size=TRUE, ...)
```

Arguments

`prefix`	Character string that begins each variable name.
`until`	Last name in the sequence, the one with the last number.
`from`	First name in the sequence, the one with the initial number.
`same_size`	If `TRUE`, pads the beginning of each number for the variable name with leading zeros so that all names are of the same width.
`...`	Other parameter values.

Details

Some data sets, particularly those from surveys, have sequentially numbered variable names, each beginning with the same prefix, such as the first later of the name of a set of related attitude items. This function generates the string of such variable names, generally intended for use in a `read` statement for reading the data and then naming the variables, or for a subsequent assignment of the names with a `names`. Relies upon the R `paste` function.

Author(s)

David W. Gerbing (Portland State University; <`gerbing@pdx.edu`>)

See Also

`paste`.

Examples

```
# generate: "m01" "m02" "m03" "m04" "m05" "m06" "m07" "m08" "m09" "m10"
to("m", 10)

# generate: "m1"  "m2"  "m3"  "m4"  "m5"  "m6"  "m7"  "m8"  "m9"  "m10"
to("m",10, same_size=FALSE)
# equivalent to standard R function
paste0("m", 1:10)

# generate a 10 x 10 data frame
d <- data.frame(matrix(rnorm(100), nrow=10))
# name the variables in the data frame
names(d) <- to("m", 10)
```

train_test — *Create Training and Testing Data*

Description

Given a data frame, create a list of either two components, `train` and `test`, or four components, for training and testing data: `train_x`, `train_y`, `test_x`, and `test_y`.

Usage

```
train_test(data, response=NULL, p_train=0.75, seed=NULL, matrix_out=FALSE)
```

Arguments

`data`	Data frame that contains the variables.
`response`	Optional name of the response variable of the response values.
`p_train`	Percentage of the input data frame to be retained for training.
`seed`	Set to a usually odd value to reproduce results.
`matrix_out`	If `TRUE` then output data structures as matrices instead of data frames.

Details

From the input data frame create training and testing data frames. If the `response` is specified, create four component data frames with x and y variables separated. Otherwise create two component data frames, `train` and `test`.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
d <- Read("Employee")

# create four component data frames that separate the response variable, y,
#   from predictor variables, X: train_x, train_y, test_x, and test_y
out <- train_test(d, response=Salary)
names(out)
# then can copy to regular data frames apart from the list output structure
X_train <- out$train_x
y_train <- out$train_y
X_test <- out$test_x
y_test <- out$test_y

# create two component data frames, train and test, which retain all
#   variables for the model in the same data frame
out <- train_test(d)
names(out)
```

```
# then can copy to regular data frames apart from the list output structure
d_train <- out$train
d_test <- out$test
```

Transform *Deprecated: Transform the Values of an Integer or Factor Variable*

Description

This function is deprecated. Instead use base R transform() function or just enter the transformation formula directly. Example, `d$Xsq <- d$X^2` to create a squared version of Variable X in the d data frame.

A wrapper for the base R `transform` function that defaults to the d data frame and provides output regarding the specified transformation(s).

Usage

```
Transform(data=d, quiet=getOption("quiet"), ...)
```

Arguments

`data`	The name of the data frame from which to create the subset, which is `d` by default.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	The list of transformations, each of the form, `variable = equation`. Each `variable` can be the name of an existing variable in the data frame or a newly created variable.

Details

The first five rows of the data frame are listed before the transformation, and the first five values of the transformed variables are listed after the transformation. The default input data frame is `d`.

Guidance and feedback regarding the transformations are provided by default. The first five lines of the input data frame are listed before the transformation, then the specified transformations are listed, followed by the first five lines of the transformed data frame.

Multiple transformations can be defined with a single statement. Note that a newly created transformed variable cannot then be used to define another transformed variable in the same `Transform()` function call. Instead, the transformed variable that depends on an earlier created transformed variable must be defined in its own `Transform()` function call.

Value

The transformed data frame is returned, usually assigned the name of `d` as in the examples below. This is the default name for the data frame input into the `lessR` data analysis functions.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

transform, factor.

Examples

```
# construct data frame
d <- read.table(text="Status Severity
1 Mild
4 Moderate
3 Moderate
2 Mild
1 Severe", header=TRUE)

# replace Status with a transformed version
d <- Transform(Status=Status-1)

# replace Status with a transformed version
# leave input d unmodified
# save transformed data frame to the created data frame called newdata
newdata <- Transform(Status=Status-1)

# construct data frame
# recode Status into a factor
d <- Transform(Status=factor(Status, labels=c("OK","Hurts","Painful","Yikes")))

# read lessR data set dataEmployee into data frame d
d <- Read("Employee")
# multiple transformations in one statement
#  Months is a new variable
#  Salary is a new version of the old Salary
#  JobSat was read as non-numeric, so as a factor, but is also ordinal
#  Plan was read as numeric values 0,1,2, now converted to a factor
d <- Transform(
  Months=Years*12,
  Salary=Salary/1000,
  Plan=factor(Plan,
     levels=c(0,1,2), labels=c("GoodHealth", "YellowCross", "BestCare"))
)
# new variable Months now exists
# if relevant, supply a corresponding variable label
# d <- label(Months, "Months Employed in the Company")
# confirm
db()


# ----------------------------
# transformations with factors
# ----------------------------
```

```
# transform a nominal variable to ordinal, re-order the categories
d <- Transform(JobSat=
  factor(JobSat, levels=c("low", "med", "high"), ordered=TRUE))

# recode levels of a factor that should remain a factor
#   with the Transform and factor functions
# using Recode destroys the factor attribute, converting to
#   character strings instead, so Recode does not allow
d <- Read("Employee")
d <- Transform(
    Gender=factor(Gender, levels=c("F", "M"), labels=c("Female", "Male"))
)

# recode levels of a factor to convert to integer first by
#   converting to integer with Transform and as.numeric
# here Gender has values M and F in the data
# integers start with 1 through the number of levels, can use
#    Recode to change this if desired, such as to 0 and 1
# Gender is now a factor to illustrate
d <- Transform(Gender=as.numeric(Gender))
d <- recode(Gender, old=c(1,2), new=c(0,1))

# recode integer values to levels of a factor with value labels
#   with the Transform function instead of Recode
# here Gender has values 0 and 1 in the data
d <- Read("Mach4")
d <- Transform(
      Gender=factor(Gender, levels=c(0,1), labels=c("Male","Female"))
      )
# ----------------------------
```

`ttest` *Generic Method for t-test and Standardized Mean Difference with Enhanced Graphics*

Description

Abbreviation: `tt`, `tt_brief`

Provides enhanced output from the standard `t.test` function applied to the analysis of the mean of a single variable, or the independent groups analysis of the mean difference, from either data or summary statistics. Includes the analysis of a dependent-groups analysis from the data. The data can be in the form of a data frame or separate vectors of data, one for each group. This output includes the basic descriptive statistics, analysis of assumptions and the hypothesis test and confidence interval. For two groups the output also includes the analysis for both with and without the assumption of homogeneous variances, the pooled or within-group standard deviation, and the standardized mean difference or Cohen's *d* and its confidence interval.

The output from data for two groups introduces the ODDSMD plot, which displays the Overlapping Density Distributions of the two groups as well as the means, mean difference and Standardized

Mean Difference. The plot also includes the results of the descriptive and inferential analyses. For the dependent-groups analysis, a scatter plot of the two groups of data also is produced, which includes the diagonal line through the scatter plot that represents equality, and a line segment for each point in the scatter plot which is the vertical distance from the point to the diagonal line to display the amount of change.

Can also be called from the more general `model` function.

Usage

```
ttest(x=NULL, y=NULL, data=d, filter=NULL, paired=FALSE,

         n=NULL, m=NULL, s=NULL, mu=NULL,
         n1=NULL, n2=NULL, m1=NULL, m2=NULL, s1=NULL, s2=NULL,

         Ynm="Y", Xnm="X", X1nm="Group1", X2nm="Group2", xlab=NULL,

         brief=getOption("brief"), digits_d=NULL, conf_level=0.95,
         alternative=c("two_sided", "less", "greater"),
         mmd=NULL, msmd=NULL, Edesired=NULL,

         show_title=TRUE, bw1="bcv", bw2="bcv",

         graph=TRUE, line_chart=FALSE, quiet=getOption("quiet"),
         width=5, height=5, pdf_file=NULL, ...)

tt_brief(...)

tt(...)
```

Arguments

`x`	A `formula` of the form Y ~ X, where Y is the numeric response variable compared across the two groups, and X is a grouping variable with two levels that define the corresponding groups, or, if the data are submitted in the form of two vectors, the responses for the first group.
`y`	If `x` is not a formula, the responses for the second group, otherwise `NULL`.
`data`	Data frame that contains the variable of interest, default is `d`.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`paired`	Set to `TRUE` for a dependent-samples t-test with two data vectors or variables from a data frame, with the difference computed from subtracting the first vector from the second.
`n`	Sample size for one group.
`m`	Sample size for one group.
`s`	Sample size for one group.
`mu`	Hypothesized mean for one group. If not present, then confidence interval only.
`n1`	Sample size for first of two groups.

n2	Sample size for second of two groups.
m1	Sample mean for first of two groups.
m2	Sample mean for second of two groups.
s1	Sample standard deviation for first of two groups.
s2	Sample standard deviation for second of two groups.
Ynm	Name of response variable.
Xnm	Name of predictor variable, the grouping variable or factor with exactly two levels.
X1nm	Value of grouping variable, the level that defines the first group.
X2nm	Value of grouping variable, the level that defines the second group.
xlab	x-axis label, defaults to variable name, or, if present, variable label.
brief	If set to `TRUE`, reduced text output. Can change system default with `style` function.
digits_d	Number of decimal places for which to display numeric values_ Suggestion only.
conf_level	Confidence level of the interval, expressed as a proportion.
alternative	Default is `"two_sided"`. Other values are `"less"` and `"greater"`.
mmd	Minimum Mean Difference of practical importance, the difference of the response variable between two group means. The concept is optional, and only one of mmd and msmd is provided.
msmd	For the Standardized Mean Difference, Cohen's d, the Minimum value of practical importance. The concept is optional, and only one of mmd and msmd is provided.
Edesired	The desired margin of error for the needed sample size calculation for a 95% confidence interval, based on Kupper and Hafner (1989).
show_title	Show the title on the graph of the density functions for two groups.
bw1	Bandwidth for the computation of the densities for the first group.
bw2	Bandwidth for the computation of the densities for the second group.
graph	If `TRUE`, then display the graph of the overlapping density distributions.
line_chart	Plot the run chart for the response variable for each group in the analysis.
quiet	If set to `TRUE`, no text output. Can change system default with `style` function.
width	Width of the pdf file in inches.
height	Height of the pdf file in inches.
pdf_file	Name of the pdf file to which the density graph is redirected. Also specifies to save the line charts with pre-assigned names if they are computed.
...	Further arguments to be passed to or from methods.

Details

OVERVIEW
If `n` or `n1` are set to numeric values, then the analysis proceeds from the summary statistics, the sample size and mean and standard deviation of each group. Missing data are counted and then removed for further analysis of the non-missing data values_ Otherwise the analysis proceeds from data, which can be in a data frame, by default named `d`, with a grouping variable and response variable, or in two data vectors, one for each group.

Following the format and syntax of the standard `t.test` function, to specify the two-group test with a formula, `formula`, the data must include a variable that has exactly two values, a grouping variable or factor generically referred to as X, and a numerical response variable, generically referred to as Y. The formula is of the form Y ~ X, with the names Y and X replaced by the actual variable names specific to a particular analysis. The formula method automatically retrieves the names of the variables and data values for display on the resulting output.

The values of the response variable Y can be organized into two vectors, the values of Y for each group in its corresponding vector. When submitting data in this form, the output is enhanced if the actual names of the variables referred to generically as X and Y, as well as the names of the levels of the factor X, are explicitly provided.

For the output, when computed from the data the two groups are automatically arranged so that the group with the larger mean is listed as the first group. The result is that the resulting mean difference, as well as the standardized mean difference, is always non-negative.

The inferential analysis in the full version provides both homogeneity of variance and the Welch test which does not assume homogeneity of variance. Only a two-sided test is provided. The null hypothesis is a population mean difference of 0.

If computed from the data, the bandwidth parameter controls the smoothness of the estimated density curve. To obtain a smoother curve, increase the bandwidth from the default value.

DATA
If the input data frame is named something different than `d`, then specify the name with the `data` option. Regardless of its name, the data frame need not be attached to reference the variable directly by its name without having to invoke the d$name notation.

PRACTICAL IMPORTANCE
The practical importance of the size of the mean difference is addressed when one of two parameter values are supplied, the minimum mean difference of practical importance, mmd, or the corresponding standardized version, msmd. The remaining value is calculated and both values are added to the graph and the console output.

DECIMAL DIGITS
The number of decimal digits is determined by default from the largest number of decimal digits of the entered descriptive statistics. The number of decimal digits is then set at that value, plus one more with a minimum of two decimal digits by default. Or, override the default with the `digits_d` parameter.

VARIABLE LABELS
If variable labels exist, then the corresponding variable label is by default listed as the label for the horizontal axis and on the text output. For more information, see `Read`.

PDF OUTPUT
To obtain pdf output, use the `pdf_file` option, perhaps with the optional `width` and `height` options.

These files are written to the default working directory, which can be explicitly specified with the R setwd function.

Value

Returned value is NULL except for a two-group analysis from a formula. Then the values for the response variable of the two groups are separated and returned invisibly as a list for further analysis as indicated in the examples below. The first group of data values is the group with the largest sample mean.

value1	Value of the grouping variable for the first group.
group1	Data values for the first group.
value2	Value of the grouping variable for the second group.
group2	Data values for the second group.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

References

Gerbing, D. W. (2023). *R Data Analysis without Programming: Explanation and Interpretation*, 2nd edition, Chapters 6 and 7, NY: Routledge.

Kupper and Hafner (1989). The American Statistician, 43(2):101-105.

See Also

t.test, density, plot.density, ttestPower, formula.

Examples

```
# ---------------------------------------------------------
# tt for two groups, from a formula
# ---------------------------------------------------------

d <- Read("Employee", quiet=TRUE)


# analyze data with formula version
# variable names and levels of X are automatically obtained from data
# although data frame not attached, reference variable names directly
ttest(Salary ~ Gender)

# short form
#tt(Salary ~ Gender)

# brief version of results
tt_brief(Salary ~ Gender)

# return the vectors group1 and group2 into the object t.out
```

```
# separate the data values for the two groups and analyze separately
Y <- rnorm(100)
ttest(Y)
t.out <- ttest(Salary ~ Gender)
Histogram(group1, data=t.out)
Histogram(group2, data=t.out)

# compare to standard R function t.test
t.test(d$Salary ~ d$Gender, var.equal=TRUE)

# consider the practical importance of the difference
ttest(Salary ~ Gender, msmd=.5)

# obtain the line chart of the response variable for each group
ttest(Salary ~ Gender, line_chart=TRUE)

# variable of interest is in a data frame which is not the default d
# access the data frame in the lessR dataLearn data set
# although data not attached, access the variables directly by their name
data(dataLearn)
ttest(Score ~ StudyType, data=dataLearn)


# ------------------------------------------------------------
# tt for a single group, from data
# ------------------------------------------------------------

# summary statistics, confidence interval only, from data
ttest(Salary)

# confidence interval and hypothesis test, from data
ttest(Salary, mu=52000)

# just with employees with salaries less than $100,000
ttest(Salary, mu=52000, filter=(Salary < 100000))


# ---------------------------------------------------------
# tt for two groups from data stored in two vectors
# ---------------------------------------------------------

# create two separate vectors of response variable Y
# the vectors exist are not in a data frame
#   their lengths need not be equal
Y1 <- round(rnorm(n=10, mean=50, sd=10),2)
Y2 <- round(rnorm(n=10, mean=60, sd=10),2)

# analyze the two vectors directly
# usually explicitly specify variable names and levels of X
#   to enhance the readability of the output
ttest(Y1, Y2, Ynm="MyY", Xnm="MyX", X1nm="Group1", X2nm="Group2")

# dependent groups t-test from vectors in global environment
```

```
ttest(Y1, Y2, paired=TRUE)

# dependent groups t-test from variables in data frame d
d <- data.frame(Y1,Y2)
rm(Y1);  rm(Y2)
ttest(Y1, Y2, paired=TRUE)
# independent groups t-test from variables (vectors) in a data frame
ttest(Y1, Y2)


# --------------------------------------------------------
# tt from summary statistics
# --------------------------------------------------------

# one group: sample size, mean and sd
# optional variable name added
tt(n=34, m=8.92, s=1.67, Ynm="Time")

# confidence interval and hypothesis test, from descriptive stats
# get rid of the data frame, analysis should still proceed
rm(d)
tt_brief(n=34, m=8.92, s=1.67, mu=9, conf_level=0.90)

# two groups: sample size, mean and sd for each group
# specify the briefer form of the output
tt_brief(n1=19, m1=9.57, s1=1.45, n2=15, m2=8.09, s2=1.59)
```

`ttestPower` *Compute a Power Curve for a One or Two Group t-test*

Description

Abbreviation: `ttp`

From one or two sample sizes, and either the within-cell (pooled) standard deviation, or one or two separate group standard deviations, generate and calibrate a power curve for either the one-sample t-test or the independent-groups t-test, as well as ancillary statistics. Uses the standard R function `power.t.test` to calculate power and then the `ScatterPlot` function in this package to automatically display the annotated power curve with colors.

For both the one and two-group t-tests, power is calculated from a single sample size and single standard deviation. For the two-sample test, the within-group standard deviation is automatically calculated from the two separate group standard deviations if not provided directly. Similarly, the harmonic mean of two separate sample sizes is calculated if two separate sample sizes are provided.

Usage

```
ttestPower(n=NULL, s=NULL, n1=NULL, n2=NULL, s1=NULL, s2=NULL,
         mmd=NULL, msmd=NULL, mdp=.8, mu=NULL,
         pdf_file=NULL, width=5, height=5, ...)

ttp(...)
```

Arguments

`n`	Sample size for each of the two groups.
`s`	Within-group, or pooled, standard deviation.
`n1`	Sample size for Group 1.
`n2`	Sample size for Group 2.
`s1`	Sample standard deviation for Group 1.
`s2`	Sample standard deviation for Group 2.
`mmd`	Minimum Mean Difference of practical importance, the difference of the response variable between two group means. The concept is optional, and only one of `mmd` and `msmd` is provided.
`msmd`	For the Standardized Mean Difference, Cohen's `d`, the Minimum value of practical importance. The concept is optional, and only one of `mmd` and `msmd` is provided.
`mdp`	Minimum Desired Power, the smallest value of power considered to provide sufficient power. Default is 0.8. If changed to 0 then the concept is dropped from the analysis.
`mu`	Hypothesized mean, of which a provided value triggers a one-sample analysis.
`pdf_file`	Name of the pdf file to which graphics are redirected.
`width`	Width of the pdf file in inches.
`height`	Height of the pdf file in inches.
`...`	Other parameter values, such as `lwd` and `lab_cex` from `plot` and `col.line` and `col.bg` from `ScatterPlot`.

Details

This function relies upon the standard `power.t.test` function to calibrate and then calculate the power curve according to the relevant non-central t-distribution. The `Plot` function from this package, which in turn relies upon the standard `plot` function, plots the power curve. As such, parameters in `Plot` for controlling the different colors and other aspects of the display are also available, as are many of the more basic parameters in the usual `plot` function.

Also plotted, if provided, is the minimal meaningful difference, `mmd`, as well as the minimal desired power, `mdp`, provided by default. Relevant calculations regarding these values are also displayed at the console. One or both concepts can be deleted from the analysis. Not providing a value `mmd` implies that the concept will not be considered, and similarly for setting `mdp` to 0.

Invoke the function with the either the within-group (pooled) standard deviation, `s`, or the two separate group standard deviations, `s1` and `s2`, from which `s` is computed. If the separate standard deviations are provided, then also provide the sample sizes, either as a single value of `n` or as two separate sample sizes, `n1` and `n2`. If separate sample sizes `n1` and `n2` are entered, their harmonic mean serves as the value of `n`.

For power analysis of the two-sample t-test, the null hypothesis is a zero population mean difference. For a one-sample test, the null hypothesis is specified, and it is this non-null specification of mu that triggers the one-sample analysis. Only non-directional or two-tailed tests are analyzed.

The effect size that achieves a power of 0.8 is displayed. If a minimal meaningful difference, mmd, is provided, then the associated power is also displayed, as well as the needed sample size to achieve a power of 0.8.

If the function is called with no parameter values, that is, as ttp(), then the values of n1, n2 and sw must already exist before the function call. If they do, these values are used in the power computations.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Plot, plot, power.t.test.

Examples

```
# default power curve and colors
ttestPower(n=20, s=5)
# short name
ttp(n=20, s=5)

# default power curve and colors
# plus optional smallest meaningful effect to enhance the analysis
ttestPower(n=20, s=5, mmd=2)

# power curve from both group standard deviations and sample sizes
# also provide the minimum standardized mean difference of
#   practical importance to obtain corresponding power
ttestPower(n1=14, n2=27, s1=4, s2=6, msmd=.5)

# power curve for one sample t-test, triggered by non-null mu
ttestPower(n=20, s=5, mu=30, mmd=2)
```

---

values *List the Values of a Variable*

Description

List the values of a variable from the global environment or a data frame.

Usage

```
values(x, data=d, ...)
```

Arguments

x	Variable for which to construct the histogram and density plots.
data	Data frame that contains the variable of interest, default is d.
...	Other parameter values for as defined processed by print, including digits.

Details

Provided for listing the values of a variable in an unattached data frame. All lessR functions that access data for analysis from a data frame, such as the default d provided by the Read function that reads the data frame from an external data file, do not require the data frame to be attached. Attaching a data frame can lead to some confusing issues, but one negative of not attaching is that simply listing the name of a variable within the data frame leads to an 'object not found' error. The values function provides access to that variable within a data frame just as is true for any other lessR function that accesses data.

The function displays the values of the specified variable with the standard R print function, so parameter values for print can also be passed to values.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

print

Examples

```
# generate 10 random normal data values
Y <- rnorm(10)
d <- data.frame(Y)
rm(Y)

# list the values of Y
values(Y)

# variable of interest is in a data frame which is not the default d
# access the breaks variable in the R provided warpbreaks data set
# although data not attached, access the variable directly by its name
data(warpbreaks)
values(breaks, data=warpbreaks)
```

VariableLabels *Create or Display Variable Labels*

Description

Assign and/or display variable labels stored in the data frame l. Variable labels enhance output of analyses either as text output at the console or as graphics, such as an axis label on a graph. The variable labels can be assigned individually, or for some or all variables.

NOTE: Mostly deprecated. Can just set var_labels=TRUE on for a call to Read to read a file of variable labels, and assign the output to l. Still needed to pull labels out of data frame from an SPSS read, or to read units to generate Rmd files from Regression .

Usage

```
VariableLabels(x, value=NULL, quiet=getOption("quiet"))

vl(...)
```

Arguments

`x`	The file reference or character string variable (see examples) from which to obtain the variable labels, or a variable name for which to assign or obtain the corresponding variable label in conjunction with the `value` parameter. Can also be a data frame from which to extract any existing variable labels.
`value`	The variable label assigned to a specific variable, otherwise `NULL`.
`quiet`	If set to `TRUE`, no text output. Can change system default with `style` function.
`...`	Other parameter values.

Details

Unlike standard R, `lessR` provides for variable labels, here stored in the data frame `l`. To read the labels from an external file, specify a file reference as the first argument of the function call. Or create a character string of variable names and labels and specify the character string as the first argument to the function call. To assign an individual variable label with this function specify the variable name as the first argument followed by the label in quotes. Not all variables need have a label, and the variables with their corresponding labels can be listed or assigned in any order. If the `l` data frame is created or modified, the output of the function must be assigned to `l`, as shown in the following examples.

When all or some of the labels are read, either from the console or an external `csv` or `Excel` file, each line of the file contains the variable name and then the associated variable label. The file types of `.csv` and `.xlsx` in the file reference listed in the first position of the function call are what trigger the interpretation of the argument as a file reference.

For a file that contains only labels, each row of the file, including the first row, consists of the variable name, a comma if a `csv` file, and then the label. For the `csv` form of the file, this is the standard `csv` format such as obtained with the `csv` option from a standard worksheet application such as Microsoft Excel or LibreOffice Calc. Not all variables in the data frame that contains the data, usually `d`, need have a label, and the variables with their corresponding labels can be listed in any order. An example of this file follows for four variables, I1 through I4, and their associated labels.

I2,This instructor presents material in a clear and organized manner.
I4,Overall, this instructor was highly effective in this class.
I1,This instructor has command of the subject.
I3,This instructor relates course materials to real world situations.

If there is a comma in the variable label, then the label needs to be enclosed in quotes.

The `lessR` functions that provide analysis, such as `Histogram` for a histogram, automatically include the variable labels in their output, such as the title of a graph. Standard R functions can also use these variable labels by invoking the `lessR` function `label`, such as setting `main=label(I4)` to put the variable label for a variable named I4 in the title of a graph.

Variable units may also be added to the third column of a variable label file. These are used for generating a better natural language text in the generation of R~Markdown files with the Rmd option on supporting functions such as Regression. For currency (USD), indicate with unit: dollar. a

Value

The data frame with the variable labels is returned.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Read.

Examples

```
# read file and then variable labels from csv files
# l <- Read("http://lessRstats.com/data/employee.csv")
# l <- VariableLabels("http://lessRstats.com/data/employee_lbl.csv")

# construct and read variable labels from console
lbl <- "
Years, Years of Company Employment
Gender, Male or Female
Dept, Department Employed
Salary, Annual Salary (USD)
JobSat, JobSat with Work Environment
Plan, 1=GoodHealth 2=YellowCross 3=BestCare
"
l <- VariableLabels(lbl)
l

# add/modify a single variable label
l <- VariableLabels(Salary, "Annual Salaries in USD")
l

# list the contents of a single variable label
VariableLabels(Salary)

# display all variable labels
VariableLabels()
```

Write *Write the Contents of a Data Frame to an External File*

Description

Abbreviation: wrt, wrt_r, wrt_x

Writes the contents of the specified data frame, such as with the default d, to the current working directory as either the default csv data file, an Excel data table, an OpenDocument Spreadsheet file, an arrow feather or parquet file, or a native R data file of the specified data frame. If the write is for a csv file, then any variable labels are written to a second csv file with "_lbl" appended to the file name. Any variable labels and variable units are automatically included in a native R data file.

Usage

```
Write(data=d, to=NULL,

         format=c("csv", "R", "Excel", "ODS", "SPSS", "feather", "parquet"),

         rowNames=NULL,

         ExcelTable=FALSE, ExcelColWidth=TRUE,

         quiet=getOption("quiet"), ...)

wrt(...)

wrt_r(..., format="R")
wrt_x(..., format="Excel")
```

Arguments

to	Name of the output file as a character string, that is, with quotes. If not included in the name, the file type is automatically added to the name, either .csv or .rda, depending of the value of format.
data	Data frame to be written as an object, that is, no quotes.
format	Format of file to be written with .csv as the default.
rowNames	Format of file to be written with .csv as the default. Set to TRUE by default unless writing to Excel or csv file and row names are just the integers from 1 to the number of rows.
ExcelTable	If TRUE, write the Excel file as an Excel table.
ExcelColWidth	TRUE by default but calculation of column widths for large files takes more time, so option to turn off.
quiet	If set to TRUE, no text output. Can change system default with style function.

... Other parameter values for csv files consistent with the usual write.table, including na="" to write missing data to a csv file as blanks instead of NA.

Details

Can specify the file name without the file type, which Write adds automatically, .csv for a comma separated values data file and .rda for a native R data file. The default file name is the name of the data frame to be written. The name of the file that is written, as well as the name of the working directory into which the file was written, are displayed at the console.

An Excel file is written using functions from Alexander Walker's openxlsx package.

Write is designed to work in conjunction with the function Read from this package, which reads a csv, fixed width format, or native SPSS or R data files into the data frame d. Write relies upon the R functions write.csv and save.

When writing the data frame in native R format, the specified name of the resulting .rda file is distinct from the name of the data frame as stored within R.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

See Also

Read, write.csv, save.

Examples

```
# create data frame called d
#n <- 12
#X <- sample(c("Group1","Group2"), size=n, replace=TRUE)
#Y <- rnorm(n=n, mean=50, sd=10)
#d <- data.frame(X,Y)

# write the current contents of default data frame d to GoodData.csv
# Write(d, "GoodData")
# short name
# write the default data frame d to the R data file d.rda
#  wrt_r()

# write the data as an Excel data table in an Excel file
# Write(d, "GoodData", format="Excel")
# with abbreviation
# wrt_x(d, "GoodData")

# access the R data frame warpbreaks
#   then, write the file warpbreaks.rda
# data(warpbreaks)
# wrt_r(warpbreaks)
```

xAnd — *Text Processing: Insert and Into a List*

Description

Inserts the word and into a vector of words, each a separate character string. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xAnd(x)
```

Arguments

`x`	The set of character strings for which to insert and.

Details

Input is a vector of character strings, output is a single character string with and inserted if needed.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
xAnd(c("sky", "land", "mountains"))
```

xNum — *Text Processing: Convert a Number to a Word*

Description

Converts a number to a word. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xNum(x)
```

Arguments

`x`	The integer to convert.

Details

Input is an integer, or coerced to integer after rounding. For integers from 0 to 12, output is the single English word. For values larger than 12, or negative, the integer is just converted to character format.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
xNum(5)
```

`xP` *Text Processing: Print Formatted Numbers*

Description

Prints numbers nicely formatted, with optional units. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xP(x, d_d=NULL, unit=NULL, semi=FALSE)
```

Arguments

`x`	The variable.
`d_d`	The digits.
`unit`	Unit of measurement for the variable.
`semi`	Add a semicolon before the unit to add some horizontal spacing in math mode.

Details

Input is numeric, output is formatted text. A special unit is `"$"`, which is added to the front of the number instead of as a trailing descriptor.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
xP(12345678.9, d_d=2, unit="$")

xP(12345678.9, d_d=2, unit="lbs")
```

`xRow` *Text Processing: Add the Word Row to Case Labels that Could be Numeric*

Description

For a vector of row names, if the names can be represented as integers the word Row is added to the beginning of each name in the vector. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xRow(x)
```

Arguments

`x`	Vector with names for each value.

Details

Input is a vector of values, output is vector of associated row labels, perhaps with the added word Row.

Author(s)

David W. Gerbing (Portland State University; `<gerbing@pdx.edu>`)

Examples

```
# The word Row gets added
v <- c(2, 4, 6)
names(v) <- c("1", "2", "3")
xRow(v)

# The word Row does not get added
v <- c(2, 4, 6)
names(v) <- c("Bill", "Tulane", "Hanna")
xRow(v)
```

`xU` *Text Processing: Capitalize First Letter of a Word*

Description

Capitalize the first letter of a word. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xU(x)
```

Arguments

`x`	The character string (word) for which to capitalize the first letter.

Details

Input is a single word. Output is the word with its first letter capitalized.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
xU("the")
```

`xW` *Text Processing: Wrap Words to Create New Lines From a Specified Line*

Description

Split a larger line into multiple lines by wrapping words with inserted line feeds. Primarily for internal use in text processing of `knitr` output. Not usually referenced by the user.

Usage

```
xW(x, w=90, indent=5)
```

Arguments

`x`	The character string to split into separate lines.
`w`	Maximum width of each line.
`indent`	Amount of spaces to indent lines after the first line.

Details

Input is a sentence. Output is the sentence word wrapped into multiple lines, each line up to the maximum width.

Author(s)

David W. Gerbing (Portland State University; <gerbing@pdx.edu>)

Examples

```
xW("The quick brown fox jumped over the lazy dog's back.", w=30)
```

Index