Package 'lessR'

March 5, 2026

Version 4.5.2

Title Less Code with More Comprehensive Results

Description Each function replaces multiple standard R functions. For example, two function calls, Read() and CountAll(), generate summary statistics for all variables in the data frame, plus histograms and bar charts. Other functions provide data aggregation via pivot tables; comprehensive regression, ANOVA, and t-test; visualizations including integrated 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 call; variable labels; time series with aggregation and forecasting; color themes; and Trellis (facet) graphics. Also includes a confirmatory factor analysis of multiple-indicator measurement models, pedagogical routines for data simulation (e.g., Central Limit Theorem), generation and rendering of regression instructions for interpretative output, and both interactive construction of visualizations and interactive visualizations with plotly.

Depends R (>= 3.6.0)

Imports graphics, grDevices, stats, utils, methods, lattice, latticeExtra, robustbase, ellipse, leaps, openxlsx, colorspace, shiny, knitr, kableExtra, xts, zoo, MASS, conflicted, plotly (>= 4.11.0), htmlwidgets, htmltools

Suggests KernSmooth, haven, readODS, triangle, arrow, rmarkdown, wesanderson, tsibble, fable, fabletools, distributional, rstudioapi

VignetteBuilder knitr

License GPL (>= 2)

Encoding UTF-8

LazyLoad yes

NeedsCompilation no

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

Author David W. Gerbing [aut, cre] (ORCID: <https://orcid.org/0000-0001-6998-8350>, Affiliation: School of Business, Portland State University)

Repository CRAN

Date/Publication 2026-03-05 06:10:29 UTC

Contents

. 4
ANOVA 5
Chart 10
corCFA 28
corEFA 35
corPrint 37
corProp 38
corRead 40
corReflect 42
Correlation 43
corReorder 46
corScree 49
CountAll 51
dataAnova_1way 52
dataAnova_2way 53
dataAnova_rb 53
dataAnova_rbf 54
dataAnova_sp 55
dataBodyMeas 56
dataCars93 57
dataEmployee 58
dataEmployee_lbl 58
dataFreqTable99 59
dataJackets 60
dataLearn 60
dataMach4 61
dataMach4_lbl 62
dataReading 63
dataStockPrice 63
dataWeightLoss 64
details 64
factors 66
Flows 68
getColors 70
interact 76
kurtosis 77
label 78
Logit 79
Merge 84
Model 86
Nest 87
order_by 90

pivot . . . 91
print.out . . . 95
print.out_all . . . 96
prob_norm . . . 97
prob_tcut . . . 98
prob_znorm . . . 100
Prop_test . . . 101
Read . . . 103
recode . . . 109
regPlot . . . 111
Regression . . . 113
rename . . . 124
rescale . . . 125
reshape_long . . . 126
reshape_wide . . . 128
savePlotly . . . 129
see . . . 131
showColors . . . 132
showPalettes . . . 133
simCImean . . . 134
simCLT . . . 135
simFlips . . . 137
simMeans . . . 138
skew . . . 140
STL . . . 141
style . . . 144
Subset . . . 153
SummaryStats . . . 155
to . . . 160
train_test . . . 161
Transform . . . 162
ttest . . . 165
ttestPower . . . 171
values . . . 173
VariableLabels . . . 174
Write . . . 176
X . . . 179
xAnd . . . 191
xNum . . . 192
xP . . . 193
xRow . . . 193
xU . . . 194
xW . . . 195
XY . . . 196

Index **222**

. *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"),
         quiet=getOption("quiet"),
         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.
quiet	If set to TRUE, no text output. Can change system default with style function.

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)
```

Chart *Charts for One or Two Categorical Variables*

Description

lessR introduces the concept of a *data view* visualization function, in which the choice of visualization function directly reflects the structure of the data and the analyst's intent for understanding the data. The function Chart() visualizes the distribution of a categorical variable along with related statistics from aggregated data of a numerical variable, either counts or a statistic such as the mean of another numerical variable. Choose the type of visualization according to the value of the parameter type.

- Bar chart with type = "bar", the default value
- Radar chart with type = "radar"
- Bubble chart with type = "bubble"
- Dot chart with type = "dot"
- Pie chart with type = "pie"
- Starburst chart with type = "pie" and another categorical variable(s) with by
- Treemap chart with type = "treemap"
- Icicle chart with type = "icicle"

Stratify, that is, divide the distribution into groups with each group plotted separately, with parameters by, which plots the groups on the same panel, or facet, which plots the groups on different panels (not applicable to bubble charts). With this conceptualization, a starburst chart is a pie chart with nested layers, For the hierarchical charts – pie and starburst, treemap, and icicle – the by stratification parameter can be a vector, defining multiple levels.

When using RStudio, plots are directed to the Plots window for the bar chart, and also the Viewer window for Plotly interactive plots. The remaining plots are all Plotly visualizations.

Unless by is a vector of at least length two, the chart is constructed from the one- or two-dimensional table that pairs each level or joint 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. A one-dimensional example is the average salary of the employees in each department. Corresponding two-dimensional example is the average salary of men and women in each department. Enter the original, raw data from which Chart calculates the summary table, or enter the summary table directly as the input data.

Chart 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 facet variable in place of by for the second categorical variable.

For bar charts, 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. For the default bar chart, a standard bar chart is presented simultaneously with the interactive version because there are some features in the standard chart not yet available in the interactive version.

Usage

```
Chart(

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

        # ----------------------------------
        # Chart type, defaults to a bar chart
        type=c("bar", "radar", "bubble", "dot", "pie", "icicle", "treemap"),
        hole=0.65,  # pie chart
        radius=0.35, power=0.5,  # bubble chart

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

        # ------------------------------------------------------------
        # Facet plot, stratify on different panels
        facet=NULL, n_row=NULL, n_col=NULL, aspect="fill",

        # ----------------------------------------------------
        # Bar chart parameters: 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, fill_scaled=FALSE, fill_chroma=75,

        labels=c("%", "input", "prop", "off"),
        labels_position=c("in", "out"),
        labels_color="white",
        labels_size=1.00,
        labels_decimals=NULL,
        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"), rotate_y=getOption("rotate_y"),
break_x=NULL, offset=getOption("offset"),
axis_fmt=c("K", ",", ".", ""), axis_x_pre="", axis_y_pre="",
label_max=100,

legend_title=NULL, legend_position="right_margin",
legend_labels=NULL, legend_horiz=FALSE,
legend_size=NULL, legend_abbrev=10, legend_adjust=0,

# ------------------------------------------------
# Draw one or more objects, text, or geometric figures
# Only applies to the standard bar chart
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,
use_plotly=getOption("lessR.use_plotly"),
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, facet1=NULL,

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

Arguments

x	Primary categorical **variable** to analyze. For bar charts, x can be a single variable (in a data frame or as a vector in the user's workspace), a vector of variables specified with c, or an entire data frame. If not specified, defaults to all non-numeric variables in the data frame given by data (or d by default). To improve label legibility, category labels are automatically wrapped: unless break_x = FALSE, spaces in labels are replaced with line breaks. To keep two short words on the same line, replace the intervening space with a tilde; the tilde is displayed as a blank in the axis label.
by	Optional second categorical variable for stratification. Creates a two-way display (e.g., stacked or grouped bars, nested pies, multi-variable bubbles), with subgroups shown within each level of x. The same stratification applies within panels when facet is used.
y	Numeric variable whose values determine bar heights, bubble sizes, or other aggregated measures across categories. If y is supplied for raw data, a summary statistic must be specified via stat. If y is omitted, counts (or proportions) are

computed from the data and used as the default response.

`data` Optional data frame that contains the variables of interest. May be raw data, from which summaries are computed, or a pre-aggregated summary table with one categorical column and one numeric column giving the heights/sizes of the plotted objects.

`filter` Logical expression or vector of row indices that defines a subset of rows in `data` to analyze. Use logical operators such as `\&`, `|`, and `!` and relational operators such as `==`, `!=`, and `>`.

`type` Chart family to produce. Default is `"bar"`. Alternatives include `"pie"`, `"treemap"`, `"bubble"`, and `"radar"`. A hierarchical pie chart (with a `by` vector) is rendered as a sunburst chart.

`hole` For pie and sunburst charts, proportion of the radius occupied by the inner hole (a doughnut chart). Set to `0` or `FALSE` for a full pie.

`radius` For bubble charts, scaling factor for bubble radius (in pixels) controlling the size of the largest displayed bubble.

`power` For bubble charts, controls the relative scaling of bubbles. The default `0.5` scales radii so that bubble *areas* are proportional to the underlying values. A value of `1` scales radii directly to the values, increasing visual differences in size.

`stat` Summary statistic applied to `y` within groups defined by `x` and optional `by`. Typical values include `"sum"`, `"mean"`, `"sd"`, `"dev"` (mean deviations), `"min"`, `"median"`, and `"max"`. The resulting summary table (pivot table) defines the plotted heights or sizes.

`stat_x` When `y` is not supplied, specifies whether to plot the `"count"` (default) of each group or its `"proportion"`.

`facet` Optional categorical variable that activates **Trellis graphics** (facets) using the `lattice` framework. A separate chart is drawn for each level of `facet`, in contrast to `by`, which overlays subgroups on the same panel.

`n_row` Number of rows in the facet layout. If specified, `n_col` is determined automatically and should not be set simultaneously.

`n_col` Number of columns in the facet layout. If specified, `n_row` is determined automatically and should not be set simultaneously. When `n_col = 1`, facet strips are placed to the left of panels instead of above.

`aspect` Lattice aspect ratio for facet panels, defined as height/width. Default `"fill"` expands panels to occupy available space. Set to `1` for square panels or `"xy"` to bank lines to an effective slope of 45 degrees.

`horiz` Orientation of bars in a bar chart. Defaults to `FALSE` (vertical bars) unless `one_plot = TRUE`, in which case horizontal bars are often more readable.

`sort` Sorting strategy for bar categories. Default `"0"` retains the original order. Use `"-"` for descending and `"+"` for ascending order of frequencies (for one-way charts) or column sums (with `by`). Not applicable to facet plots. When `one_plot = TRUE`, the default is `"+"`.

`beside` For a two-way bar chart, if `TRUE` plots the levels of the second variable as adjacent bars (grouped bars) rather than stacked segments.

`stack100` Produces a 100% stacked bar chart when a `by` variable is present, equivalent to setting `stat_x = "proportion"` with `by`.

`gap` Controls the spacing between bars; passed to the `space` argument of `barplot`. Default is `0.2`, except for two-variable plots with `beside = TRUE`, where the default is `c(0.1, 1)`.

`scale_y` Optional numeric vector of length three defining the y-axis (numeric axis) scale: minimum, maximum, and number of intervals. Applies to bar and similar charts.

`one_plot` For multiple `x` variables, selects whether to draw a separate bar chart for each variable or combine all variables into a single multi-item chart. By default, if variables share a common response scale (e.g., Likert items), `one_plot` is set to `TRUE`; otherwise it defaults to `FALSE`.

`theme` Color **theme** for this analysis. Use `style` to set persistent defaults across analyses.

`fill` Fill color(s) for bars, pie slices, tiles, or bubbles. Default is the qualitative `"hues"` palette under the `"colors"` theme, or an ordered sequential palette (e.g., `"blues"`) for ordinal categories. For other themes, default fill is taken from the corresponding gradient (e.g., `"reds"` for `"darkred"`). May also be any vector of colors (e.g., from `getColors`) or predefined palettes including color-blind–safe options such as `"viridis"`. When `fill` is set to the name of `y` (or `(count)` for tabulated counts), values of `y` are mapped to a color scale. Not used when `fill_split` is active.

`color` Border color of plotted objects (bars, slices, bubbles, tiles). May be a vector to vary borders by category. Default is `bar_color_discrete` from `style`.

`transparency` Transparency of filled areas, from `0` (opaque) to `1` (fully transparent). Default is `trans_bar_fill` from `style`.

`fill_split` For bar charts, splits bars into two fill colors relative to a numeric threshold. Bars with `y <= fill_split` are drawn in the first fill color; larger values use the second. Alternatively, supply a length-2 vector of colors.

`fill_scaled` For bar charts without a `by` variable, scales the lightness of the fill color according to height (the value of `y`). Larger values yield darker bars. When `fill` is a single color, a sequential scale is generated; when `fill` is two colors, a diverging scale is used.

`fill_chroma` Chroma (saturation) for `fill_scaled` bars. Full saturation is 100; lower values approach grayscale. Has no effect for the `"gray"` theme, which is already achromatic.

`labels` Adds numeric labels to bars or pie slices. Default `"%"` displays percentages, `"prop"` shows proportions, and `"input"` shows the underlying numeric values (counts or supplied `y`). If `y` is omitted, the input values are the tabulated counts.

`labels_position` Position of labels for pies/sunbursts. Default is `"in"` (inside slices); use `"out"` to place labels outside.

`labels_color` Color(s) of the plotted labels. May be a vector; if fewer colors are given than categories, colors are recycled.

`labels_size` Character expansion factor for label text. Default is `0.95`, or `0.9` of that value when `beside = TRUE` and `labels_position = "in"` (to account for narrower bars).

`labels_decimals`
: Number of decimal places displayed in labels. Defaults to 0 for integer-valued `y` and 2 for `"prop"`.

`labels_cut`
: Minimum relative size required to show a label. When `labels_position = "out"`, the default is `0.028` for simple charts, and `0.040` when a `by` variable is present or multiple `x` variables are combined.

`xlab`
: **Axis label** for the x-axis. If omitted, the label is taken from the variable label (if present) or the variable name. If `xy_ticks = FALSE`, no x-axis label is drawn. When no `y` is specified, `xlab` defaults to `"Index"` unless explicitly set.

`ylab`
: Axis label for the y-axis. If omitted, the label is taken from the variable label (if present) or the variable name. If `xy_ticks = FALSE`, no y-axis label is drawn.

`main`
: Title of the chart. Size and color may be controlled via `main_cex` and `main_color` in `style`.

`sub`
: Subtitle placed below `xlab`. Not yet implemented.

`lab_adjust`
: Two-element numeric vector (x-label, y-label) giving approximate inch offsets for axis labels. Positive values move labels away from the plotting region. Not applicable to facet (Trellis) plots.

`margin_adjust`
: Four-element numeric vector (top, right, bottom, left) that adjusts plot margins in inches. Positive values expand the corresponding margin. Not applicable to facet plots.

`pad_y_min`
: Proportion of padding added at the lower end of the y-axis (0–1).

`pad_y_max`
: Proportion of padding added at the upper end of the y-axis (0–1).

`rotate_x`
: For bar charts, rotation (in degrees) of category labels on the x-axis, typically used to accommodate long labels in combination with `offset`. When `rotate_x = 90`, labels are vertical and an alternative placement algorithm is used, so `offset` is usually unnecessary.

`rotate_y`
: Applies to BPFM (bubble plot frequency matrix), a sequence of stacked bubble charts. Controls rotation of labels along the vertical axis.

`break_x`
: For bar charts, controls automatic line-breaking of category labels. When `TRUE`, spaces are converted to new lines and tildes to blanks (keeping words joined by a tilde on the same line). Defaults to `TRUE` for vertical bars with `rotate_x = 0`, and `FALSE` otherwise.

`offset`
: For bar charts, controls the spacing between axis labels and the axis itself. Default is `0.5`. Larger values (e.g., `1.0`) create additional room for rotated or long labels.

`axis_fmt`
: Numeric format for axis labels. Default `"K"` shows thousands as `"K"` (e.g., `100000` as `100K`). Alternatives include `","` (comma separators with decimal point), `"."` (period separators), or `""` to disable formatting.

`axis_x_pre`
: Prefix for labels on the x-axis, such as `"$"`.

`axis_y_pre`
: Prefix for labels on the y-axis, such as `"$"`.

`label_max`
: For bar charts, improves console readability of text output by setting a target maximum label length. Longer labels are abbreviated in the printed frequency distribution. The limit is not strict when necessary to preserve uniqueness.

legend_title	Title of the **legend**. Usually set automatically from variable names, but must be supplied explicitly when plotting raw count matrices without variable metadata.
legend_position	Legend placement when plotting two variables. Default is in the right margin. Standard positions such as "topleft", "top", and "topright" are also available; see legend.
legend_labels	Legend labels when plotting two variables. Defaults to the levels of the second (or by) variable.
legend_horiz	If TRUE, draws the legend horizontally; default is vertical.
legend_size	Character expansion factor for legend text.
legend_abbrev	If specified, truncates legend title and labels to at most the given number of characters (subject to preserving uniqueness).
legend_adjust	Horizontal shift of the legend in two-way bar charts. Positive values move the legend to the right from its default position.
add	For bar charts, **overlays** additional objects (text or geometric figures) on the plot. The first argument "text" writes arbitrary text; geometric options include "rect", "line", "arrow", "v_line" (vertical line), and "h_line" (horizontal line). The value "means" is shorthand for vertical and horizontal lines at the respective means. Does not apply to facet plots. Use style parameters such as add_fill and add_color to control appearance.
x1	First x-coordinate (in standardized -1 to 1 units) for each added object.
y1	First y-coordinate for each added object.
x2	Second x-coordinate for each added object. Used for "rect", "line", and "arrow".
y2	Second y-coordinate for each added object. Used for "rect", "line", and "arrow".
quiet	If TRUE, suppresses text output to the console. The default can be changed via style.
do_plot	If TRUE (default), produces the chart. Set to FALSE to compute and return results without plotting.
use_plotly	If TRUE (default), produces a Plotly-based interactive chart in the RStudio Viewer window in addition to the static plot in the Plots window. Some advanced options apply only to the static chart.
pdf_file	If specified, directs graphics output to a PDF file with this name.
width	Width of the plot window (or PDF device) in inches. Default is 4.5.
height	Height of the plot window (or PDF device) in inches. Default is 4.5.
digits_d	Number of decimal digits used for displayed numeric summaries. Defaults to at least 2 or one more than the maximum number of digits in the response variable, whichever is larger.
out_size	Target maximum line width (in characters) for console frequency tables of a single variable. Longer lines trigger a vertical layout for improved readability.

n_cat	For analyses of all variables in a data frame, sets the maximum number of unique values for a numeric variable to be treated as categorical rather than continuous. Default is 0. **Deprecated**: It is preferable to convert such variables explicitly to factors.
value_labels	For factors, defaults to factor levels; for character variables, defaults to the character values. May be used to override axis labels on the x-axis. If the variable is a factor and value_labels is NULL, levels are used with embedded spaces replaced by line breaks. If x and y share the same scale, labels may also be used on the y-axis. Label size is controlled via axis_cex and axis_x_cex in style.
rows	**Deprecated**. Old name for filter.
facet1	**Deprecated**. Old parameter name, replaced by facet.
eval_df	Controls whether the function checks for existence of data and referenced variables. Defaults to TRUE, except when **shiny** is loaded, in which case it is set to FALSE so that Shiny applications run without conflict. Set to FALSE when using the pipe operator %>%.
fun_call	Function call object used internally (e.g., by **knitr**) to reconstruct the original call.
...	Additional graphical parameters passed to base barplot, legend, and par. Common options include cex.main (title size), col.main (title color), line types such as "dotted" or "dotdash", and subtitle options sub and col.sub. Axis label orientation can be adjusted with las = 3, and bar spacing with space in one-variable bar charts.

Details

OVERVIEW

Chart() visualizes numerical values associated with one or two categorical variables, each with a relatively small number of levels. By default, colors for bars, background, and grid lines are taken from the active style theme, but all can be customized. Base computations use standard R functions such as barplot, chisq.test, and, for two variables, legend. For horizontal bar charts (horiz = TRUE), category labels are drawn horizontally and the left margin is automatically extended to accommodate both the labels and the axis title.

DATA

Conceptually, the chart is built from a summary table in which each row consists of a level of the categorical variable x paired with a numerical value y, with as many rows as there are levels of x. You may:

- supply x and y directly as a pre-aggregated summary table, or
- supply x (and optionally y) at the observation level and let Chart() aggregate over the levels of x (and by) using stat.

A second categorical variable by can be used to form a two-way table.

The filter parameter subsets rows (cases) of the input data frame according to a logical expression or a set of integers that specify the row numbers to retain. Use the standard R logical operators described in Logic, such as \& (and), | (or), and ! (not), and the standard relational operators

described in Comparison, such as == (equality), != (not equal), and > (greater than). Alternatively, specify a vector of integers that correspond to row numbers. See the Examples.

The input can be factors, numeric values, characters, or a matrix. You can:

- enter raw data and let Chart() compute frequencies or summaries, or
- enter a pre-tabulated summary table of counts or statistics.

When y is not supplied, the numerical values are simply the counts of each level of x (and of each combination of x and by).

TWO DATA MODES FOR PLOTLY OUTPUTS

Chart() supports two conceptual modes for aggregated values used in the plots and tables:

- **Count mode** (default): if y is NULL, the chart uses counts of x (and by, when supplied).
- **Summary mode**: if y is numeric, the chart aggregates y over the categories of x (and by) using stat.

From full data with repeated values of x (and by), you can reduce to a summary table using one of the following transformations:

Transformation	Meaning
"sum"	sum
"mean"	mean
"sd"	standard deviation
"dev"	mean deviation
"min"	minimum
"median"	median
"max"	maximum

All numeric values (both in console tables and Plotly hovers) are formatted according to digits_d.

Before plotting, Chart() constructs a 1-D table (x) or 2-D table (by × x) of either counts (when y = NULL) or aggregated y (when y is supplied). For count mode, Chart() prints the frequency table and a chi-square test of equal probabilities. For summary mode, it prints the aggregated table (no chi-square test is computed for numeric summaries). These tables serve as a concise audit of the data supplied to the visualization.

NON-HIERARCHICAL PLOTLY CHARTS

For non-hierarchical charts (type = "bar", "radar", or "bubble"):

- with y = NULL, the geometry encodes counts;
- with y supplied, the geometry encodes the chosen stat of y per category (and per group when by is supplied).

In bubble charts, bubble size is proportional to the aggregated value. Use `radius` (in pixels) and `power` to control size mapping (area proportional to the value when `power = 0.5`).

HIERARCHICAL PLOTLY CHARTS

Hierarchical charts include pies with a `by` variable (sunburst charts) and `type = "treemap"` or `"icicle"`. For these charts, `Chart()` constructs a path table from `x` and `by`, where `by` may be:

- a single factor (one additional level), or
- a data frame of multiple factors, where each column represents a deeper level in the hierarchy.

It then aggregates `y` (or counts, if `y = NULL`) along the path:

- Node values are computed by applying `stat` within each node to its child records.
- For **additive** statistics (`"sum"` and counts), parent node values equal the sum of their children. Children are sized proportionally to their parent (`branchvalues = "total"`), so hover percentages of parent/root are well defined.
- For **non-additive** statistics (`"mean"`, `"median"`, `"min"`, `"max"`, `"sd"`), parent values are computed at the parent level using the same `stat` and are not sums of children. In this case, "% of parent/root" is not shown in hovers because these proportions are not meaningful for non-additive summaries.

All numeric values shown in hovers are formatted using `digits_d`. For hierarchical charts:

- additive modes show the aggregated value and, when appropriate, the % of parent and % of root;
- non-additive modes show the aggregated value only.

Titles for console output and interactive plots reflect the mode:

- Count mode: e.g., "Count of `x`" (optionally "by `by`").
- Summary mode: e.g., "`stat` of `y` by `x`" (and "by `by`" when grouped).
- Hierarchical: analogous titles using the same `stat` and variable names.

In all cases, `Chart()` preserves factor level order when building 1-D and 2-D tables and prints tables with informative dimnames. The console chi-square test is computed only for count mode (1-D or 2-D).

VECTOR OF x-VALUES

A vector of categorical `x`-variables (character or factor) generalizes to a matrix of one-dimensional plots, depending on the value of `type`:

- for `type = "bar"`, a stacked bar chart (stack of one-dimensional bar plots),
- for `type = "bubble"`, a stacked bubble chart, referred to as a *bubble plot frequency matrix* (BPFM).

COLORS

For a one-variable plot, the default bar colors are taken from the current theme via the `bar_fill_discrete` argument of `style`, which by default uses the qualitative HCL palette `"hues"`. Alternatively, set the bar colors explicitly with the `fill` parameter, using:

- a single color,
- a palette name, or
- a vector of colors, e.g., from `getColors`.

Pre-defined sequential and divergent HCL ranges are available through `getColors`. The qualitative sequence `"hues"` provides equally spaced HCL colors (same chroma and luminance). Sequential and divergent ranges are available at 30-degree increments around the HCL color wheel, including `"reds"`, `"rusts"`, `"browns"`, `"olives"`, `"greens"`, `"emeralds"`, `"turquoises"`, `"aquas"`, `"blues"`, `"purples"`, `"violets"`, `"magentas"`, and `"grays"`.

Define a *divergent color scale* by providing a vector of two such ranges to `fill`, e.g., `c("purples", "rusts")`. These are especially useful for multiple bar charts with a common response scale (e.g., Likert items). Alternatively, specify colors manually, such as `c("coral3", "seagreen3")` for a two-level by variable.

For finer control, call `getColors` explicitly and pass its result to `fill`, adjusting chroma (`c`) and luminance (`l`), or defining a custom hue (`h`). See `getColors` for details.

The values of another variable can be mapped to bar fill by setting `fill` equal to that variable's name, typically `y` when supplied. When `y` is tabulated, refer to it as `(count)`. Larger values produce darker bars.

Additional pre-specified palettes include `"rainbow"`, `"terrain"`, and `"heat"`. Distinct palettes include `"distinct"` (maximally separated hues), the viridis family (`"viridis"`, `"cividis"`, `"magma"`, `"inferno"`, `"plasma"`), and color-blind friendly options such as `"Okabe-Ito"`. Wes Anderson–inspired palettes such as `"Moonrise1"`, `"Royal1"`, `"GrandBudapest1"`, `"Darjeeling1"`, and `"BottleRocket1"` are also available (with variants using `2` or `3` in the name where defined).

LEGEND

When two variables are plotted, a legend is produced with entries for each level of the second or `by` variable. By default, the legend is placed in the right margin. This position can be changed with `legend_position`, which accepts `"right_margin"` and any valid position accepted by the standard R `legend` function.

The legend title can be abbreviated with `legend_abbrev`, which specifies the maximum number of characters. The legend is vertical by default, but can be drawn horizontally with `legend_horiz`.

LONG CATEGORY NAMES

Category labels are often long. Adjust their display with `rotate_x` and `rotate_y`, in conjunction with `offset`, which moves labels away from the axis to compensate for rotation. These settings can be made persistent with `style`. To reset to defaults, call `style()` again.

Spacing codes for category names:

1. Any space in a category name is converted to a new line in the plotted label.
2. To keep words on the same line, replace the space with a tilde `~`; the tilde is rendered as a space without a line break.

For console output, you can limit label length with `label_max`. Longer names are abbreviated to the specified number of characters, with a mapping table provided to show the correspondence between abbreviated and full names. For one-variable frequency distributions, `out_size` controls the maximum line width before the distribution is printed vertically instead of horizontally.

MULTIPLE BAR CHARTS ON THE SAME PANEL (PLOT)

For multiple `x`-variables, set `one_plot = TRUE` to overlay individual bar charts on a single panel. This is especially useful when all items share a common response scale (e.g., Likert items). By default, `Chart()` produces a single-panel display when a common response scale is detected.

The algorithm for detecting a common response scale identifies the variable with the largest set of responses, then checks that all other variables' responses are contained within that set. Some items may not exhibit all possible responses (e.g., no one chooses "Strongly Disagree"), but as long as at least one variable contains the full response set, the scales are treated as common.

Regardless of this automatic detection, you can explicitly set `one_plot` to either `TRUE` or `FALSE`. Explicitly setting `one_plot` bypasses the commonality check and saves computation.

ENTER NUMERIC VARIABLE DIRECTLY

Instead of computing counts from raw data, you can enter a numeric variable directly as `y`, together with a categorical `x` (and possibly a categorical `by`). In this case, the chart uses the supplied numeric values as-is (or aggregates them according to `stat`). Alternatively, you can read a pre-tabulated table of counts into R as a matrix or data frame and pass it to `Chart()`.

STATISTICS

In addition to the Plotly and static charts, descriptive and optional inferential statistics are reported. For count mode, a frequency table (one variable) or joint frequency table (two variables) is displayed, followed by Cramér's V and the chi-square test of independence (or equal probabilities) by default. For summary mode, the aggregated table is printed without a chi-square test, as the test is not appropriate for numeric summaries.

VARIABLE LABELS

If variable labels are stored in the data frame (e.g., via `Read` or `VariableLabels`), they are used by default as axis labels and in text output. For a single variable, the `x`-axis label defaults to the variable label unless `xlab` is explicitly supplied. For two variables, the plot title is derived from both variable labels unless overridden by `main`. Variable labels are also shown in the printed tables.

PDF OUTPUT

To write graphics to a PDF file, use `pdf_file`, optionally with `width` and `height`. Files are written to the current working directory, which you can explicitly set with `setwd`.

ONLY VARIABLES ARE REFERENCED

Arguments that denote variables in `Chart()` (and other `lessR` functions) must be names of existing variables, either in the referenced data frame (e.g., the default `d`) or in the user's workspace (global environment). Expressions are not evaluated directly. For example:

```
> Chart(cut(rnorm(50))) # does NOT work
```

Instead, assign the expression to a variable and reference that variable:

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

Value

For interactive visualizations, `Chart()` returns a Plotly `htmlwidget` object (class `plotly`) that can be printed for interactive viewing or saved as a self-contained HTML file.

For standard (non-interactive) charts, the output can optionally be saved as an R object. Otherwise, it appears only in the console (unless `quiet = TRUE`). Two types of components are provided: *readable text output* and *numerical statistics*.

The readable output consists of character strings such as frequency or summary tables suitable for display. The numerical components are statistics amenable to further analysis. This design supports reproducible reporting in R Markdown documents by referencing the name of each output component directly, using the syntax `object$component`.

Each component appears only when relevant to the current analysis. For example, cell proportions (`out_prop`) are included only for two-way tables.

Example: save the output of a standard chart to an object with any valid R name, such as `b <- Chart(Dept)`. View the available output elements with `names(b)`, and access a specific component by prefixing with the object name, such as `b$out_chi` to display the chi-square test results. These objects can be displayed directly in the console or within R Markdown for integrated text and analysis.

Bar charts only: tabulated numerical variable `y`

When `Chart()` is used as a bar chart with a tabulated numerical variable (counts or proportions), the object may contain:

Readable output:

`out_title` Title of the analysis.

`out_lbl` Variable label.

`out_counts` Frequency or two-way frequency distribution.

`out_chi` Chi-square test of equal probabilities (one variable) or independence (two variables).

One variable `out_miss` Number of missing values.

Two variables `out_prop` Cell proportions.

Two variables `out_row` Row-wise cell proportions.

Two variables `out_col` Column-wise cell proportions.

Statistics:

`n_dim` Number of dimensions, 1 or 2.

`p_value` p-value for the null hypothesis of equal proportions (one variable) or independence (two variables).

`freq` Data frame of the 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

When Chart() reads a numeric variable y directly from the data and summarizes it across one or two categorical variables, the returned object can include:

out_y Values of y used in the analysis.

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.

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC. URL: https://plotly.com/r/

See Also

X, XY, getColors, barplot, table, legend, savePlotly.

Examples

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

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

# for each level of Dept, display the frequencies
# -----------------------------------------------
# bar chart, standard and plotly
  Chart(Dept)  # bar chart by default

  # radar chart, plotly only
  Chart(Dept, type="radar")

  # bubble chart, plotly only
  Chart(Dept, type="bubble")

  # pie chart, plotly only
  Chart(Dept, type="pie")

  # treemap chart, plotly only
```

```
  Chart(Dept, type="treemap")

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

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

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

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

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

# change the theme just for this analysis, as opposed to style()
Chart(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
Chart(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
Chart(Gender, fill=c("palegreen3","tan"))

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

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

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

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

```
Chart(Dept, y=Salary, stat="dev", sort="+", fill_split=0)

# scale the luminosity of the bars with the sequential scale
Chart(Dept, y=Salary, stat="deviation", sort="+",
      fill_scaled=TRUE, fill="green")

# scale the luminosity of the bars with a divergent scale
Chart(Dept, y=Salary, stat="deviation", sort="+", fill_scaled=TRUE,
         fill=c("red", "blue"))

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

# at each level of Dept, show the frequencies of the Gender levels
  # bar chart, standard and plotly
  Chart(Dept, by=Gender)  # bar chart by default

  # radar chart, plotly only
  Chart(Dept, by=Gender, type="radar")

  # bubble chart, plotly only
  Chart(Dept, by=Gender, type="bubble")

  # pie chart, plotly only
  Chart(Dept, by=Gender, type="pie")

  # treemap chart, plotly only
  Chart(Dept, by=Gender, type="treemap")
# -------------------------------------

# Trellis (facet) plot, bar chart only
Chart(Dept, facet=Gender)

# at each level of Dept, show the row proportions of the Gender levels
#   i.e., 100% stacked bar graph
Chart(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
Chart(Gender, by=JobSat, fill="reds", labels="off")

# specify two fill colors for Gender
Chart(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
Chart(Gender, by=Dept, beside=TRUE, horiz=TRUE, labels_position="out")

# horizontal bar chart of two variables, put legend on the top
Chart(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")
Chart(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()


# ---------------------------------------------------------------------
# bar chart from a statistic aggregated across 1 or 2 categorical variables
# ---------------------------------------------------------------------
Chart(Dept, y=Salary, stat="mean")

Chart(Dept, by=Gender, y=Salary, stat="mean")

# ------------------------------------------------------------
# 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)

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

# stacked bubble charts for 20 6-pt Likert scale items
Chart(m01:m20, type="bubble")




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

# custom divergent scale with pre-defined color palettes
#  with implicit call to getColors
Chart(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)
Chart(X1)
Chart(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)
Chart(X1)
Chart(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)
Chart(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)
Chart(Dept, y=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)
Chart(Dept, y=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?
Chart(Dept, y=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
Chart(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>
Chart(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 including and after 4.4.3, the correlation matrix computed by Correlation is now the object returned if show is equal to "cor", the default, or a missing data analysis if equal to "missing".

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
# 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 including and after 4.4.3, the correlation matrix computed by Correlation is now the object returned if show is equal to "cor", the default, or a missing data analysis if equal to "missing".

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)
```

---

corPrint *Print a Correlation Matrix with Special Formatting*

Description

Display a correlation matrix in readable, compact form.

Usage

```
corPrint(R, min_value=0)
```

Arguments

R	Square, input correlation matrix.
min_value	The minimal value in magnitude of the displayed correlations.

Details

Drop the 0. characters of each displayed correlation coefficient. For example, display 0.42 as 42. Have an option to only display correlations larger in magnitude than a minimum threshold, so that all correlations between -min_value and +min_value display as blank spaces.

Value

The output correlation matrix.

Author(s)

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

See Also

Correlation, corReorder.

Examples

```
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)

# display all the correlations
corPrint(mycor)

# only display correlations in magnitude of 0.2 or larger
corPrint(mycor, min_value=.2)
```

---

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 subset 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.

Usage

```
Correlation(x, y, data=d,
         miss=c("pairwise", "listwise", "everything"),
         show=c("cor", "missing"),
         fill_low=NULL, fill_hi=NULL,
```

```
          brief=FALSE, digits_d=NULL, heat_map=TRUE,
          main=NULL, bottom=3, right=3, quiet=getOption("quiet"),
          pdf_file=NULL, 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.
show	Default is to compute and show correlations, or specify to compute and show missing data by setting to "missing".
fill_low	Starting color for a custom sequential palette.
fill_hi	Ending color for a custom sequential palette.
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 in the bottom margin of heat map.
right	Number of lines in the right margin of heat map.
quiet	If set to TRUE, no text output. Can change system default with style function.
pdf_file	Indicate to direct pdf graphics to the specified name of the pdf file.
width	Width of the pdf file in inches.
height	Height of the pdf file in inches.
...	Additional arguments passed to cor and cor.test, e.g., method="spearman" or method="kendall", and alternative="less" or alternative="greater".

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 device. Set pdf_file 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
`r`: Estimated correlation coefficient
`tvalue`: t statistic for testing $H_0 : r = 0$
`df`: Degrees of freedom for the t test
`pvalue`: P-value for the t test
`lb`: Lower bound of the confidence interval for `r`
`ub`: Upper bound of the confidence interval for `r`
`n`: Number of non-missing paired observations
`cov`: Sample covariance

matrix
`out_cor`: Correlations or `out_missing`: Missing values analysis

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

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 d assigned to R
R <- Correlation()

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

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

# analysis of entire data frame that is not R
data(attitude)
R <- 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, quiet=getOption("quiet"),
          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.
quiet	If set to TRUE, no text output. Can change system default with style function.
pdf	if TRUE, create possibly two PDF files: the hierarchical cluster analysis dendrogram if requested and the heat map of the reordered correlation matrix.
width	Width of a pdf file in inches.
height	Height of a 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

`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 dataEmployee
# 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)
```

`Flows` *Sankey Flow Diagram (2- or 3-Stage) with Plotly*

Description

Create a Sankey flow diagram between source and target categories with weighted links that visualizes movement of volume between categorical stages. Supports either two stages (A -> B) or three stages (A -> B -> C). Flows (links) are colored by the leftmost source category and retain that color across all legs. Node fills can be neutral gray or (by default) the left/source nodes can inherit the flow colors while intermediate and right nodes remain gray. In RStudio, the visualization renders in the `Viewer` pane.

Usage

```
Flows(
  value,
  stage1,
  stage2,
  stage3 = NULL,
  data = d,
  # appearance
  title = NULL,
  fill = NULL,
  link_alpha = 0.55,
  nodes_gray = FALSE,
  neutral_gray = "gray80",
  border_gray  = "gray60",
  lift_y = 0,
  labels_size = 1.00,
  digits_d = 0
)
```

Arguments

`stage1`	Column name for the left (source) stage. May be given unquoted (recommended) or as a character string.
`stage2`	Column name for the middle (or right, in 2-stage) stage. Unquoted or character string.
`stage3`	Optional column name for the right stage (3-stage). If `NULL`, a 2-stage diagram is created. Unquoted or character string.
`value`	Column name with non-negative weights (e.g., counts, volume, tonnage) used to determine link thickness. Unquoted or character string.

data	A data frame containing the stages and the weight column. If NULL, the function will look for the columns in the calling environment.
title	Optional chart title. If NULL, a title is constructed from the provided stage names.
fill	Optional color range name or vector used for *source* categories. If NULL (default), the range is inherited from the current theme via .color_range(.get_fill(getOption("theme"), n.x), where n.x is the number of unique stage1 categories.
link_alpha	Numeric from 0 to 1. Transparency for flows (links). Default 0.55.
nodes_gray	Logical. If FALSE (default), left/source nodes are colored (matching their flows) and all other nodes are neutral gray. If TRUE, all nodes are neutral gray.
neutral_gray	Color for gray node fills. Defaults to "gray80".
border_gray	Color for node borders. Defaults to "gray60".
lift_y	Proportion of vertical shift of the displayed chart. Positive values lift the chart upward.
labels_size	Size of displayed labels, with 0.90 the default.
digits_d	Integer number of decimals for link values shown in the hover. Default 0.

Details

Flow semantics

Flows is intended for directional flows where the width of each band represents the amount moving from one category to the next (e.g., supply chain, traffic, energy, user funnels). It is *not* a general association plot for arbitrary cross-tabulations without directional meaning. For hierarchical composition without directional flow, consider Chart with parameter type set to "sunburst", "treemap", "radar", or "pie" instead.

Color mapping

Flows are colored by the *leftmost* source category (stage1). In a 3-stage diagram, colors are carried from stage1 through stage2 to stage3 so each link retains its origin color. When nodes_gray = FALSE (default), only source nodes inherit the flow colors; middle and right nodes are neutral gray. When nodes_gray = TRUE, all nodes are gray. If fill is NULL, colors are drawn from the current lessR theme (getOption("theme")) via .get_fill() and .color_range().

Layout

Nodes are arranged in fixed columns: left = stage1, middle = stage2, right = stage3 (if provided). Vertical positions are spaced evenly within each column. The diagram prints automatically in interactive sessions and returns a Plotly htmlwidget (invisibly) for programmatic use.

Value

An object of class htmlwidget (Plotly). The widget prints automatically in interactive sessions; otherwise it is returned (invisibly) for further composition or saving.

Author(s)

David W. Gerbing

See Also

Chart for other Plotly-based charts in **lessR**, including type="bar", type="sunburst", "treemap", "pie", "radar", and "bubble". Histogram and Plot also create Plotly-based charts.

Examples

```
# Example data: Source -> Port -> Destination with volumes
SupplyChain <- data.frame(
  Source      = c("China","China","China","Germany","Germany",
                  "Brazil","Brazil","USA","USA"),
  Port        = c("Los Angeles","Seattle","Houston","New York","Baltimore",
                  "Miami","Houston","Los Angeles","Miami"),
  Destination = c("Portland","Chicago","Dallas","Boston","Philadelphia",
                  "Atlanta","Dallas","San Francisco","Atlanta"),
  Volume      = c(320,180,150,220,130,200,100,170,140)
)

# 3-stage: Source -> Port -> Destination
Flows(Volume, Source, Port, Destination, data=SupplyChain)

# 2-stage: Source -> Destination, ignore Port
Flows(Volume, Source, Desination, data=SupplyChain)

# All nodes neutral gray
Flows(Volume, Source, Port, Destination, data=SupplyChain,
       nodes_gray=TRUE, link_alpha=0.6)

# Supply a specific range (overrides theme), show one decimal in hover
Flows(Volume, Source, Port, Destination, data=SupplyChain,
       fill="blues", digits_d=1)
```

---

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
Chart(Dept, fill=getColors())
# or with implicit call to getColors
Chart(Dept, fill="colors")
# or an implicit call with the blues
Chart(Dept, fill="blues")
# or explicit call
Chart(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
Chart(Dept, fill="viridis")
```

`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", "Histogram", "ScatterPlot". If missing, then the valid names are listed.

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)
```

`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, the 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,
     pt_size=0.9, transparency=0.8,
     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,

     xlab=NULL, ylab=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.
`pt_size`	Size of the plotted points of the scatterplot for a one-predictor model.
`transparency`	Transparency level of the plotted points of the scatterplot, from 0 to 1 for full transparency.
`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, `"dffits"`, 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.
`xlab`	x-axis label, name of the x-variable by default.
`ylab`	y-axis label, name of the y-variable by default.
`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 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 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 latter 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_rows=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)
```

---

order_by *order_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

```
order_by(data=d, by, direction=NULL, quiet=getOption("quiet"), ...)
```

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

order_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("+","+","-").

order_by is based on the standard R function order, though the order_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 <- order_by(d, Severity)

# sort Description in descending order, sort Severity within
#  each level of Description in ascending order
d <- order_by(d, c(Description, Severity), direction=c("-", "+"))

# sort by row names in ascending order
d <- order_by(d, row.names)

# randomly re-shuffle the rows of data
d <- order_by(d, random)
```

---

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_group_show=TRUE, na_remove=TRUE,
         out_names=NULL, sort=NULL, sort_var=NULL,
```

```
table_prop=c("none", "all", "row", "col"), table_long=TRUE,
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. If no numerical variable, then ignore.
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_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, deleting the level from the analysis.
na_remove	Sets base R parameter na.rm. If TRUE, the default, removes missing values from the variable(s), which are aggregated, then reports how many values were missing. Otherwise, the aggregation statistic for a cell with any missing data returns NA.
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. When TRUE, 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, by=c(Dept, Gender))
# short form
pivot(d, table, by=c(Dept, Gender), table_long=TRUE)

# multiple functions for which to aggregate
pivot(d, c(mean,sd,median,IQR), Years, by=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)
```

`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 .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 contributors' 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 data file 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 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, header=TRUE,

         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:// or https://.
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 data 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 column 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 specifies one or more missing data values in the data table. Can combine numerical and character codes, such as missing=c(-99, "xxxx").
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 more 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, the only acceptable value is 1 for the first column.
`header`	If `TRUE`, the default, then the first row of the data table contains the variable names.
`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 defined with the R read functions, such as the `read.table` function for text files, with row.names and header.

Details

The following table lists various file formats along with the associated R packages and functions for reading them.

Extension	Format	Package	Function
.csv	Text, comma-separated values	utils	`read.csv()`
.tsv	Text, tab-separated values	utils	`read.delim()`
.prn	Text, space-separated values	utils	`read.table()`
.txt	Text, comma or tab-separated	utils	`read.table()`
.xls	Excel	openxlsx	`read.xlsx()`
.xlsx	Excel	openxlsx	`read.xlsx()`
.ods	ODS	readODS	`read_ods()`
.feather	Feather	arrow	`read_feather()`
.parquet	Parquet	arrow	`read_parquet()`
.rda	R data	base	`load()`
.sav	SPSS	haven	`read_sav()`
.zsav	SPSS (compressed)	haven	`read_zsav()`
.dta	Stata	haven	`read_dta()`

.sas7bdat	SAS	haven	`read_sas()`

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 `?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 specification 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,

          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"), bubble_plot=NULL,
          graphics=TRUE, size=NULL, pdf=FALSE, width=6.5, height=6.5,
          refs=FALSE,

          n_cat=getOption("n_cat"),

          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.
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	By default is FALSE. 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 all 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.
size	Size of plotted points.
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.
bubble_plot	For a single predictor variable, by default, plot as bubbles only when the single predictive variable and the target variable have less than or equal to 12 unique values. Otherwise, set TRUE or FALSE.
refs	If TRUE, then list the references for R and the packages used from which functions were used to generate the output.
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.
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 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 Histogram function with stat="density" 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. .

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 four parts: results, explanations of the results, interpretations of the results, documentation of 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 call or set globally, such as with `options(explain=FALSE)`. Turning off all the 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 `Histogram` function with `stat="density"` 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 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 width 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 latter 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 W, the following multiple regression model is automatically defined.
$\hat{Y} = b_0 + b_x X + b_w W + b_{xw} XW$
From that model, the function 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 changed globally with style. 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 cannot be 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="", shape=c("rect", "square"))
```

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, 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.
shape	In general, the input data is a rectangular, wide-form data frame. However, it can also be a symmetric, square correlation or covariance matrix.

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

For a correlation or related symmetric, square input matrix, the applicable parameters are the data parameter to specify the input matrix, group to name the two columns that contain the variable names, and response to specify the name of the variable with data values that are the input correlations or related.

The name entered for group is repeat it for the first two columns output with a suffix of 1 and 2, respectively. For example, group="Item" becomes "Item1" and "Item2". If not name for group is entered, the default variable names are Col and Row.

Author(s)

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

See Also

reshape.

Examples

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

# 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")

# input correlation matrix
R <- matrix(nrow=3, ncol=3, byrow=TRUE,
c(1.000,0.480,0.320,
  0.480,1.000,0.240,
  0.320,0.240,1.000))
colnames(R) <- c("X1", "X2", "X3")
```

```
rownames(R) <- colnames(R)
# reshape to long
dl <- reshape_long(R, shape="square", group="Item", response="Cor")
dl
```

---

reshape_wide *Reshape a Long-Form Data Frame to Wide-Form*

Description

Takes the variables in a long-form data frame, `widen`, `response`, and `ID`, and transforms to a wide form data frame. All other variables are deleted in the transformed data frame. A simple wrapper for Base R `reshape` with sensible parameter names and sensible defaults, and able to specify a range of variables to transform. Conversion currently limited to converting based on a single grouping variable.

Usage

```
reshape_wide(data, widen, response, ID, prefix=NULL, sep="_", ...)
```

Arguments

`data`	Data frame that contains the variables to analyze as a wide-form single column.
`widen`	Name of the (single) grouping variable in the input long-form column to have its individual values listed as columns in the corresponding wide form version.
`response`	Name of the variable of the response values in the input long-form column that becomes the data values in the wide-form version.
`ID`	Name of the ID field in the long-form column used to identify each row in the wide-form version.
`prefix`	If `TRUE`, prefix the column names in the wide form of each corresponding level of the `widen` variable with the name of the `response`. Unless the values of `widen` 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 `"_"`.
`...`	Older parameter values to be converted internally.

Details

Here is the correspondence between the original `reshape` parameter names and the `reshape_wide` parameter names.

reshape	reshape_wide

```
v.names   response
timevar   widen
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, widen=Group, response=Response, ID=Person)

# with the name of the response prefixed to the column names
reshape_wide(dl, widen=Group, response=Response, ID=Person,
             prefix=TRUE, sep=".")
```

---

savePlotly *Save the last Plotly chart to a standalone HTML file*

Description

Save a Plotly/htmlwidget chart to an HTML file. If no object is supplied, `savePlotly()` uses the most recent Plotly chart generated by **lessR** functions (cached in `options("lessR.last_plotly")`). A sensible filename is auto-generated from chart metadata (e.g., "pie", variable names) or from the chart title if metadata are unavailable. Optionally opens the saved file in your default web browser.

Usage

```
savePlotly(obj = NULL, file = NULL, open = TRUE,
           selfcontained = TRUE, libdir = NULL)
```

Arguments

`obj`	A Plotly/htmlwidget object. If `NULL` (default), the function attempts to retrieve the last chart from `options("lessR.last_plotly")`.
`file`	Output path for the HTML file. If `NULL`, a filename is constructed in the current working directory using the chart kind and variables, e.g., `plotly_pie_Dept.html` or `plotly_pie_DeptGender.html`.

open	Logical. If TRUE (default), open the saved file in the system's default web browser.
selfcontained	Logical. Passed to htmlwidgets::saveWidget(). If TRUE, bundle all dependencies into a single HTML file.
libdir	Optional directory to copy widget dependencies when selfcontained = FALSE. Passed to htmlwidgets::saveWidget().

Details

savePlotly() is designed for seamless export of interactive outputs created by **lessR** Plotly functions When the obj parameter is NULL, the function retrieves the most-recent Plotly widget stored by lessR in options("lessR.last_plotly").

If file is not provided, a filename is automatically derived from attributes embedded in the widget (when available), specifically:

- lessR_kind (e.g., "pie")
- lessR_xname (primary variable name)
- lessR_byname (grouping variable name, if any)

If those attributes are missing, the function falls back to parsing the plain chart title (e.g., "Gender by Dept"). Characters unsuitable for filenames are removed.

The function prints the absolute path of the saved file to the console and (invisibly) returns that path. Setting open = TRUE will launch the file in your default browser after saving.

Value

(Invisibly) returns the normalized file path of the saved HTML file (a character scalar).

Side Effects

- Prints a confirmation message with the save location.
- Optionally opens the saved HTML file in the default browser when open = TRUE.

Note

Requires the **htmlwidgets** package to be installed. If no recent Plotly chart is available and obj is NULL, the function will error and prompt you to pass a chart or render one first.

Author(s)

lessR Development Team

See Also

saveWidget,

Examples

```
## Not run:

d <- Read("Employee")

if (interactive()) {
  # Basic usage with the most recent lessR Plotly chart
  PieChart(Dept, data=d)
  savePlotly()  # -> e.g., "plotly_pie_Dept.html"

  # Grouped example; filename reflects both variables
  PieChart(Dept, by=Gender, data=d)
  savePlotly()  # -> e.g., "plotly_pie_DeptGender.html"

  # Explicit object and custom path
  plt <- PieChart(Dept, data=d)
  savePlotly(plt, file=file.path(tempdir(), "my_dept_pie.html"), open=FALSE)

  # Non-selfcontained (assets in a side folder)
  savePlotly(plt, file="dept_pie.html", selfcontained=FALSE,
             libdir="dept_pie_libs")
  ## End(Not run)
  }
```

---

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,
       dist=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.
dist	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, dist="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)
```

STL *Seasonal and Trend Decomposition of a Time Series via Loess*

Description

Decompose a time series into seasonal, trend and irregular components using loess, a wrapper to provide additional information to the Base R `stl` function and accept more general input beyond the `stl` required time series object input. This function accepts a Base R time series from the global environment as input, but also accepts data in the traditional `x,y` format where `x` is a variable of type `Date`. Moreover, the `Date` variable can be inferred from digital character string inputs. The time unit of the input dates can also be aggregated, such as changing monthly dates to quarterly dates.

Usage

```
STL(x, y=NULL, data=d, filter=NULL,
    ts_format=NULL, ts_unit=NULL, ts_agg=c("sum","mean"),
    show_range=FALSE, robust=FALSE, quiet=FALSE, do_plot=TRUE,
    pdf_file=NULL, width=6.5, height=6)
```

Arguments

`x`	Dates for the time series as a variable within a data frame, or a time series object created with the Base R function `ts`. If the dates are of type `character`, they will be automatically converted to type `Date`.
`y`	Numerical variable that is planted in the time series, not used if `x` is a time series object.
`data`	Data frame that contains the `x` and `y` variables. Default data frame is `d`.
`filter`	A logical expression that specifies a subset of rows of the data frame to analyze.
`ts_format`	A specified format (for R function `as.Date()`) that describes the values of the date variable on the x-axis, needed if the function cannot identify the correct date format to properly decode the given date values. For example, describe a character string date such as `"09/01/2024"` by the format `"%m/%d/%Y"`. See `details` for more information.
`ts_unit`	Specify the time unit from which to plot a time series, plotted when the `x`-variable is of type `Date`. Default value is the time unit that describes the time intervals as they occur in the data. Aggregation according to the time unit will occur as specified, such as a daily time series aggregated to `"months"`. Dates are currently stored as variable type `Date()` which stores information as calendar dates without times of the day. Valid values include: `"days"`, `"weeks"`, `"months"`, `"quarters"`, and `"years"`, as well as `"days7"` to provide seasonality for daily data on a weekly instead of annual basis. Otherwise, for forecasting, the time unit for detecting seasonality will usually be ‘"months"‘ or ‘"quarters"‘.
`ts_agg`	Function by which to aggregate over time according to `ts_unit`. Default is `"sum"` with an option for `"means"`.
`show_range`	Display the range for each component.

robust	stl() parameter for a more robust solution.
quiet	If TRUE, no text output to the console.
do_plot	If FALSE, no 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.

Details

PURPOSE
Obtain and plot the seasonal, trend, and the irregular (remainder or residual) components of a time series using the Base R stl function. The corresponding plot is of four panels, one for the data and one each for the seasonal, trend, remainder components. Provide additional information comparing the relative sizes of the components in the form of the percent of variance of each component accounted for and the range of values of each component.

Seasonality is detected over a year, such as four quarters in a year or 12 months in a year. The exception is for daily data, for which seasonality can be indicated by the time unit of "days7", which will evaluate seasonality over the seven days of a week.

RANGE BARS
By definition, the data shows the most variability compared to the three components. If the four panels were scaled on the same y-axis, then the relative magnitude of the variations in each of the components, such as assessed by the ranges of each of their values, would be more directly observable. For example, if seasonality has no practical presence in the data, then the amplitude of the seasonal plot, the range of the seasonal component values, would be a small fraction the amplitude of the data plot, only reflecting random noise. Plotted on the same panel, the comparison would be direct.

Instead, however, the plots of the data and each of the three components are drawn such that each component is plotted on its own panel with its own scale with the most detail possible. The purpose of the range bars is to show a relative scale for comparison across the panels. Each range bar is a magnification indicator. The larger the bar, the more expanded is the corresponding panel, which means the smaller the variation of the component relative to the range of the data. Shrinking the size of a range bar along with the corresponding panel to the same size as the range bar for the data, the smallest range bar, would show the comparison directly.

DATE FORMAT
STL() makes reasonable attempt to decode a character string date value as the x-axis variable as read from a text data file such as a csv file. Some date formats are not available for conversion by default, such as date values that include the name of the month instead of its number. And, in general, there is no guarantee that a date format is not miss-inferred as they can be inherently ambiguous. If the default date conversion is not working, then manually supply the date format following one of the format examples in the following table according to the parameter ts_format.

Date	Format
"2022-09-01"	"%Y-%m-%d"
"2022/9/1"	"%Y/%m/%d"

"2022.09.01"	"%Y.%m.%d"
"09/01/2022"	"%m/%d/%Y"
"9/1/15"	"%m/%d/%y"
"September 1, 2022" "	%B %d, %Y"
"Sep 1, 2022"	"%b %d, %Y"
"20220901"	"%Y%m%d"

For emphasis, each range bar is displayed in a pale yellow color.

Value

An `stl()` object and text to the console.

Here is an example of saving the output to an R object with any valid R name, such as `s`: `s <- STL(Price)`. To see the names of the output objects for that specific analysis, enter `names(s)`. To display any of the objects, precede the name with `s$`, such as to view the saved frequency distribution with `s$out_freq`. Or, only list the name of the output object to get the four output components displayed as a single data frame. View the output at the R console or within a markdown document that displays your results.

`x.name`: Name of the date variable on the horizontal axis. `trend`: Value of the `trend` component for each x-value. `season`: Value of the `season` component for each x-value. `error`: Value of the `error` component for each x-value.

Author(s)

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

See Also

stl

Examples

```
# read the built-in data set dataStockPrice
d <- Read("StockPrice")
# extract just the data for Apple, the first 473 rows of data
d <- d[1:473,]

# do the time series decomposition
STL(Month, Price)

# enter a time series, here one that comes with Base R
# monthly average air temperatures in Nottingham, UK from 1920 to 1939
# get the time series into the global environment
my.ts <- nottem
STL(my.ts)
```

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"),
  lessR.use_plotly=getOption("lessR.use_plotly"),

  results=getOption("results"), explain=getOption("explain"),
  interpret=getOption("interpret"), document=getOption("document"),
  code=getOption("code"),

  width=120, show=FALSE, ...)
```

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.

lessR.use_plotly — If set to TRUE, an use_plotly plot is created in the RStudio Viewer window in addition to the usual static plot in the plot window for most plotting routines.

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")
XY(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
X(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))
XY(Years, Salary, facet=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)
X(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 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

# 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
```

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", pt_size=0.8,
```

```
          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.
pt_size	Size of plotted points for the paired t-test Cleveland dot plot of differences.
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 and also tab limited and space unlimited text files, 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`, or `.tsv`, or `.prn` text 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", "txt", "tsv", "prn",
                  "Excel", "ODS", "R", "SPSS", "feather", "parquet"),
```

```
          row_names=NULL, quote="if_needed", missing=NULL, dec=".", sep=NULL,

          ExcelTable=FALSE, ExcelColWidth=TRUE,

          quiet=getOption("quiet"), ...)

wrt(...)

wrt_r(..., format="R")
wrt_x(..., format="Excel")
```

Arguments

`data`	Data frame of which the contents are to be written to an external data file, that is, no quotes.
`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 according to the specified `format`. Or, specify the file name with the file extension from which the `format` is derived. If omitted, then the file name is the data frame name.
`format`	Format of file to be written with `.csv` as the default.
`row_names`	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.
`quote`	Specifies how character data values are to be quoted. The default is `"if_needed"`, which puts quotes around character data values that include spaces, commas, tabs, and other white spaces. The value of `TRUE` quotes character data values, needed or not.
`missing`	The data value indicates missing for text files. Defaults to a blank space except for `prn` files, which is then `NA`.
`dec`	The character that represents the decimal point for text files.
`sep`	The character that separates adjacent data values for text files. Defaults to `","`.
`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

The default file name is the name of the data frame to be written, otherwise use `to` to specify the name. To specify the file type of the output data file, do so with any available file type provided as part of the file name for the output file, or by the value of the `format` parameter. Can specify the

file name without the file type, which, if no `format` is provided, `Write` adds automatically the `.csv` file extension. 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.

The following table lists various file formats along with the associated R packages and functions for writing them. The default text file format, `.txt`, defaults to `dec="."` and `sep=","`, that is, North American `csv` format, but can be customize as needed.

Extension	Format	Package	Function
.txt	Text, customize dec and sep	R utils	`write.table()`
.csv	Text, comma-separated values	R utils	`write.table()`
.tsv	Text, tab-separated values	R utils	`write.table()`
.prn	Text, space-separated values	R utils	`write.table()`
.xlsx	Excel	openxlsx	`writeData()` or `writeDataTable()`
.ods	ODS	readODS	`write_ods()`
.feather	Feather	arrow	`write_feather()`
.parquet	Parquet	arrow	`write_parquet()`
.rda	R data	R base	`save()`
.sav	SPSS	haven	`write_sav()`

`Write` is designed to work in conjunction with the function `Read` from this package, which reads a csv or other text file, 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.table`, `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(d)

# 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)
```

X *Univariate and Grouped Displays for a Continuous Variable*

Description

lessR introduces the concept of a *data view* visualization function, in which the choice of visualization function directly reflects the structure of the data and the analyst's intent for understanding the data. The function X() visualizes the distribution of a continuous variable along with associated summary statistics. The specific representation is selected with the type argument:

- Histogram with type = "histogram" (default)
- Density plot with type = "density"
- Box plot with type = "box"
- Violin plot with type = "violin"
- Strip (one-dimensional scatter) plot with type = "strip"
- Integrated violin-box-strip (VBS) plot with type = "vbs"

Stratification divides the distribution into groups. Use by to overlay the groups within a single panel, or facet to display each group, or combination of groups, in separate panels.

When running in RStudio, static displays appear in the Plots window, while histograms and density plots also appear as interactive Plotly visualizations in the Viewer window.

Usage

```
X(

    # ------------------------------------------------
    # Data from which to construct the visualization
    x=NULL, by=NULL, facet=NULL, data=d, filter=NULL,

    # ----------------------------------------
    # Type of plot for the continuous variable x
    type = c("histogram", "freq_poly", "density",
             "violin", "box", "strip", "bs", "vbs"), # violin, box, strip
    stat=c("count", "proportion", "density"),

    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"),

# ------------------------------------------------------
# Form of the histogram - bin the continuous variable x
counts=FALSE,
bin_start=NULL, bin_width=NULL, bin_end=NULL, breaks="Sturges",
cumulate=c("off", "on", "both"), reg="snow2",  # Cumulative histogram

# ------------------------------------------------------
# Density (smooth curve) plot
show_histogram=TRUE,
bandwidth=NULL, kind=c("general", "normal", "both"),
fill_normal=NULL, fill_hist=getOption("se_fill"),
color_normal="gray20", line_width=NULL,
x_pt=NULL, y_axis=FALSE,
rug=FALSE, color_rug="black", size_rug=0.5,

# ------------------------------------------------
# Integrated violin/box/strip plot
vbs_plot="vbs", vbs_ratio=0.9, bw=NULL, bw_iter=10,
violin_fill=getOption("violin_fill"),
box_fill=getOption("box_fill"),
pt_size=NULL,
vbs_pt_fill="black",
vbs_mean=FALSE, fences=FALSE, n_min_pivot=1,
k=1.5, box_adj=FALSE, a=-4, b=3,

ID="row.name", ID_size=0.60,
MD_cut=0, out_cut=0, out_shape="circle", out_size=1,

# ------------------------------------------------------------------
# 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,
axis_fmt=c("K", ",", ".", ""), axis_x_pre="", axis_y_pre="",

# ------------------------------------------------------------------------
# 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,
    use_plotly=getOption("lessR.use_plotly"),
    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, facet1=NULL, facet2=NULL,

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

Arguments

x	Variable(s) to analyze. May be a single numeric variable, a vector in the workspace, multiple variables combined with c, or an entire data frame. If omitted, defaults to all numeric variables in the data frame d.
by	A categorical grouping variable. When supplied, overlapping histograms or other distributional displays are drawn on the same coordinate system.
facet	A categorical conditioning variable, or a vector of two such variables, that activates Trellis (lattice) graphics to produce separate panels for each level or combination of levels.
data	Optional data frame that contains the variables of interest. Default is d.
filter	Logical expression or integer index vector identifying the rows to retain for analysis.
type	Type of univariate display. Default is "histogram". Additional options include: "freq_poly" (frequency polygon), "density" (kernel density curve), "violin", "strip" (one-dimensional scatter plot), and "vbs" (integrated violin-box-strip display). When using "vbs", the vbs_plot parameter may specify a subset such as "bs" for a box-strip display.
stat	Specifies the vertical axis content. Default is "count". Use "proportion" for relative frequencies, or "density" for density scaling in histogram mode.
n_row	Optional number of rows for multi-panel displays created by facet.
n_col	Optional number of columns for multi-panel displays. If n_col = 1, facet strips appear at the left rather than top.
aspect	Aspect ratio of lattice panels (height/width). Default "fill" expands panels to available space. Use 1 for square panels or "xy" for banking to 45 degrees.
theme	Color theme. Use style to set persistent defaults.

fill	Fill color for bars or density regions. May be a single color, named palette ("grays", "hcl", "blues", "reds", "greens"), or custom vector.
color	Border color for bars or density curves. May be a vector. Default follows style.
transparency	Transparency of filled areas. Default from style.
counts	If TRUE, display counts for each bin on the histogram.
bin_start	Optional starting value of the first bin.
bin_width	Optional bin width, with or without a specified bin start.
bin_end	Optional value that must fall inside the final bin.
breaks	Method used to compute bins or an explicit set of breakpoints. Default is "Sturges"; options include "Scott" and "FD".
cumulate	Produces a cumulative histogram when set to "on". Use "both" to superimpose the standard histogram.
reg	Color of the regular histogram when cumulate="both".
show_histogram	When plotting a density curve, optionally draw a histogram behind it.
bandwidth	Bandwidth for kernel density estimation. Default begins with bw.nrd0 but may be iteratively increased for smoother curves.
kind	Determines whether to display the general density curve, the normal density curve, or both.
fill_normal	Fill color used for the normal density curve.
fill_hist	Fill color for histograms under density curves.
color_normal	Border color for the normal density curve.
line_width	Line width of density curves.
x_pt	Value on the x-axis from which to illustrate a unit interval of density area. Applies when kind = "general".
y_axis	If TRUE, display the density axis.
rug	If TRUE, draw a rug plot beneath the density curve.
color_rug	Color of rug ticks.
size_rug	Line width of rug ticks.
vbs_plot	Controls which components of the violin-box-strip display appear. Characters "v", "b", and "s" indicate violin, box, and strip components. Upper/lower case permitted. Adjust colors with style parameters such as violin_fill, box_fill, out_fill, etc.
vbs_ratio	Relative height of the violin (and box) component.
bw	Bandwidth for violins; larger values yield smoother shapes.
bw_iter	Number of iterations for modifying the bandwidth to produce smoother violins.
violin_fill	Fill color for violins.
box_fill	Fill color for box plots.

pt_size	Size of points in strip plots.
vbs_pt_fill	Point color in the strip component of the VBS display. Default is black.
vbs_mean	If TRUE, show the mean in box plots.
fences	If TRUE, draw Tukey inner fences.
n_min_pivot	Minimum sample size required for a group to appear in a VBS pivot table (default 1).
k	IQR multiplier for determining whisker length.
box_adj	If TRUE, adjust box plot for skewness using medcouple.
a, b	Scaling factors for adjusted boxplot whiskers.
ID	Variable supplying labels for selected points (outlier identification). Defaults to row names.
ID_size	Point label text size.
MD_cut	Mahalanobis distance cutoff for outliers in 2-variable plots.
out_cut	Number or proportion of extreme points to label. Interpreted differently for univariate vs. bivariate displays.
out_shape	Plotting symbol for outliers.
out_size	Point size for labeled outliers.
xlab	X-axis label. Defaults to variable name or label.
ylab	Y-axis label. Defaults to "Frequency" or "Proportion".
main	Main title for the plot.
sub	Subtitle (not yet implemented).
lab_adjust	Two-element vector adjusting axis-label positions.
margin_adjust	Four-element vector adjusting plot margins.
rotate_x	Rotation (in degrees) of x-axis tick labels.
rotate_y	Rotation (in degrees) of y-axis tick labels.
offset	Distance between tick labels and axis.
scale_x	Vector specifying start, end, and number of intervals for the x-axis.
axis_fmt	Formatting of axis numbers ("K", ",", ".", or "").
axis_x_pre	Prefix for x-axis labels (e.g., "$").
axis_y_pre	Prefix for y-axis labels.
add	Add text or geometric objects (rectangles, lines, arrows, horizontal/vertical lines, etc.) to the plot.
x1	First x-coordinate for added objects.
y1	First y-coordinate.
x2	Second x-coordinate (for applicable object types).

y2	Second y-coordinate.
quiet	If TRUE, suppress textual output.
do_plot	If TRUE, generate the plot.
use_plotly	If TRUE (default), draw a Plotly interactive version in addition to the static plot.
pdf_file	Optional file name to direct static graphics output to PDF.
width	Width of plot window (inches).
height	Height of plot window (inches).
digits_d	Digits of precision for printed statistics.
Rmd	Name of an R Markdown file to generate containing the analysis.
n_cat	Maximum number of unique integer values to treat as categorical. **Deprecated**.
rows	**Deprecated**; use filter.
facet1	**Deprecated**; use facet.
facet2	**Deprecated**; use a two-element vector in facet.
eval_df	If TRUE, checks that variables exist in the supplied data frame. Must be FALSE in Shiny or pipe workflows.
fun_call	Function call used internally for knitr compatibility.
...	Additional graphics parameters passed to hist, par, and related functions (e.g., xlim, ylim, cex.main, col.main, cex, col.lab).

Details

OVERVIEW

X() implements the univariate, distributional view for numeric variables. Internally it still relies on the base R hist function for histogram binning when type = "histogram", but extends that core infrastructure to a larger family of displays including frequency polygons, kernel density estimates, violin plots, strip plots, and the integrated violin-box-strip (VBS) display. For histogram-based displays, the freq option of hist is always set to FALSE, so counts, proportions, or densities are controlled through the lessR arguments (stat and type) rather than directly through hist.

For plotting densities, the recommended approach is to use type = "density", which computes and displays a kernel density estimate. The older usage of stat = "density" is maintained for backward compatibility but is not recommended for new code.

VARIABLES AND TRELLIS PLOTS

At a minimum there is one primary numeric variable, x, which results in a single univariate display (histogram, density, VBS, etc.). Facet graphics (also called Trellis graphics), from Deepayan Sarkar's **lattice** package, may be requested by supplying one or two categorical conditioning variables to the facet argument. A single facet variable produces one panel per level of that variable; a two-element vector in facet produces a panel for each combination of levels of the two facet variables.

In each panel, the same numeric primary variable x is displayed, conditioning on the facet levels. The combination of x with facet thus yields a grid of small multiples that share scales and styling, facilitating comparisons across subgroups without changing the underlying analysis.

BOXPLOTS AND THE VBS DISPLAY

For a single variable, the preferred summary display is often the integrated violin-box-strip (VBS) plot, requested with type = "vbs". Only the violin or only the box portion can be obtained either via the corresponding aliases or by setting vbs_plot to "v" or "b". To view a boxplot of a continuous variable across the levels of a categorical variable, either as part of the full VBS plot or alone, there are two primary styles:

1. X(x, by = g, type = "box")
2. X(x, facet = f, type = "box")

Both styles convey the same information about numeric-by-category distributions. The difference is visual: with by, multiple boxplots appear in a single panel using the qualitative color palette from getColors, with all hues chosen to be comparable in perceived saturation and brightness. With facet, a separate panel is drawn for each group, typically using a single hue. The same default qualitative colors are used throughout lessR, including in Chart.

DATA

The data may be supplied as a vector from the global environment (user workspace), as one or more variables within a data frame, or as an entire data frame. The default input data frame is d. A different source data frame may be specified with the data argument. When multiple variables are listed, only those of numeric type are analyzed by X().

Variables in a data frame are referenced directly by their names; there is no need to use d$name, with, or attach. If a name exists both as a vector in the global environment and as a variable in the specified data frame, the vector in the global environment takes precedence.

To obtain a histogram (or other univariate display) for each numeric variable in the default data frame d, call X() without specifying x. For a different data frame, set the data argument accordingly. To restrict analysis to a subset of variables, specify them with : or c, such as m01:m03 or c(m01, m02, m03).

The filter parameter subsets rows (cases) of the input data frame according to either (a) a logical expression or (b) a vector of integer row indices. Logical expressions use standard R logical operators as described in Logic (e.g., \& for "and", | for "or", ! for "not") and relational operators as described in Comparison (e.g., == for equality, != for inequality, > for "greater than"). A vector of integers can be created with standard R syntax; see the Examples section for illustrations.

COLORS

Colors for individual elements of the plot can be controlled directly through arguments such as fill (for areas) and color (for borders), or indirectly through the global style system. The style function allows selection of a color theme for the entire analysis. The default theme is "lightbronze", with additional themes such as "gray" for grayscale and several color-emphasis themes (e.g., "red", "green").

For black backgrounds and partially transparent colors, use style(sub_theme = "black"). For all color options, including fill, a setting of "off" is equivalent to "transparent".

For fill, you may specify a single color, a vector of colors, or a color range. Besides standard color names, lessR provides multiple hue-balanced palettes, including "hues", as well as pre-specified ranges "blues", "reds", and "greens". Standard R palettes "rainbow", "terrain", and "heat" are also available. Custom ranges can be generated via getColors.

HISTOGRAMS AND RELATED OUTPUT

When type = "histogram", the output includes a frequency histogram with lessR's default background and grid styling, along with optional relative frequency and/or cumulative histograms. Summary statistics and a table of bin information (bin boundaries, midpoints, counts, proportions, cumulative counts, and cumulative proportions) are provided as part of the returned object.

Bin construction can follow the default Sturges rule or be customized via bin_width, bin_start, and bin_end, or via the breaks argument (which can accept numeric vectors or named rules such as "Scott" or "FD" as in hist). lessR adds more informative error checking and guidance when user-specified bins do not fully span the data or otherwise lead to problematic binning.

If multiple variables are supplied (including a complete data frame), X() analyzes each numeric variable in turn. The related CountAll function performs a similar role across mixed types, producing bar charts for categorical variables and histograms for numeric variables. Faceting via facet can be combined with X() to create trellised displays across conditioning variables.

VARIABLE LABELS

If variable labels exist (e.g., created by Read), they are used by default for the horizontal axis label and in the text output. Variable names are still available, but labels provide more descriptive and publication-ready annotations.

ONLY VARIABLES ARE REFERENCED

In lessR functions, the primary argument x (and any other variable arguments) must be specified as variable names, not expressions. The variables must exist either in the referenced data frame (such as the default d) or in the global environment. Direct expressions are not evaluated. For example:

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

Instead, assign the expression to a named object and then reference that name:

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

ERROR DETECTION

A common error for beginning users of hist is to specify a sequence of break points that does not fully span the data, often constructed with seq. Base R then produces a cryptic error. In contrast, X() (and the Histogram() alias) checks for this problem before calling hist and, if detected, issues a more informative message along with guidance on how to correct the specification.

In addition, full control of the binning is available via bin_width and bin_start. If bin_start is specified without bin_width, the bin width is determined by the default Sturges rule.

PDF OUTPUT

To obtain PDF output directly, use the pdf_file argument, optionally with width and height to control the device size in inches. The resulting files are written to the current working directory, which can be set explicitly via setwd. This mechanism facilitates high-quality, publication-ready graphics saved directly from X().

Value

The output may be assigned to an R object; otherwise it is printed directly to the console. The returned object contains two types of components:

- Readable output: character strings such as tables and summaries, formatted for display.
- Statistics: numerical values suitable for further computation or inclusion in dynamic documents.

These components are designed to support R Markdown workflows. Any element of the saved object can be inserted into an R Markdown document by referencing its name, preceded by the object name and the $ operator (see examples). Only elements relevant to the requested analysis are returned. For example, bin proportions appear only when a histogram is requested.

To save the results, assign the output to an object, such as h <- Histogram(Salary). Use names(h) to view the available components, and display any one by typing, for example, h$out_freq. Output can be viewed interactively at the console or embedded in R Markdown documents along with narrative 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 proportions
cumulate: Bin cumulative counts
cprop: Bin cumulative proportions

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.

Hubert, M. and Vandervieren, E. (2008). An adjusted boxplot for skewed distributions, Computational Statistics and Data Analysis 52, 51865201.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer.

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC. URL: https://plotly.com/r/

See Also

XY, Chart, getColors, style.

Examples

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

# make sure default style is active
style()

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

# histogram with all defaults
X(Salary, type="histogram")
# with deprecated alias
Histogram(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
X(Salary, color="off")

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

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

```
X(Salary, fill="rusts", color="brown", transparency=.1)

# histogram with purple color theme, translucent gold bars
style("purple", sub_theme="black")
X(Salary)
# back to default color theme
style()

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

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

# histogram with specified bin width
X(Salary, bin_width=20000, xlab="My Variable")

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

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

# histogram with non-default values for x- and y-axes
d[2,4] <- 45000
X(Salary, scale_x=c(20000,160000,8))

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

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

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

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

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

# default density plot
X(Salary, type="density")
```

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

# display only the general estimated density
#  so do not display the estimated normal curve
# specify the bandwidth for the general density curve
X(Salary, type="density", bandwidth=8000)

# display only the general estimated density and a corresponding
#  interval of unit width around x_pt
X(Salary, type="density", 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
X(c(Years, Salary), type="density")

# ------------------------------------------------
# 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
X(X)

# histograms for all numeric variables in data frame called d
# d is the default name, so does not need to be specified with data
X()

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

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

# integrated violin/box/scatter plot (VBS)
d <- Read("Employee")
X(Salary, type="vbs")

X(Years, by=Gender, pt_size=1.25,
     fill=c("olivedrab3", "gold1"),
     color=c("darkgreen", "gold4"), type="vbs")



# by variable, different colors for different values of the variable
```

```
# two panels
X(Salary, facet=Dept)

# large sample size
x <- rnorm(10000)
X(x)

# custom colors for outliers, which might not appear in this subset data
style(out_fill="hotpink", out2_fill="purple")
X(Salary, type="vbs")
style()

# no violin plot or scatterplot, just a boxplot
X(Salary, type="box")

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

d <- rd("Employee")

# Place a message in the top-right of the graph
# Use \n to indicate a new line
X(Salary, add="Salaries in 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>
X(Salary, add=c("rect", "Salaries in our Company"),
      x1=c(82000, 100000), y1=c(7.7, 7), x2=118000, y2=6.2)
```

---

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)
```

XY *Scatterplots including Time Series*

Description

lessR introduces the concept of a *data view* visualization function, in which the choice of visualization function directly reflects the structure of the data and the analyst's goal for understanding the data. The function XY() visualizes the joint distribution of two continuous variables, or more generally the relationship among multiple variables, along with associated summary statistics. The specific representation is selected with the type argument:

- Scatterplot of points with type = "scatter" (default)
- Smoothed scatterplot with type = "smooth"
- Contour plot (2-dimensional density display) with type = "contour"

For the standard scatterplot, stratification divides the data into groups via the arguments by and facet. The by argument overlays groups in a single coordinate system for direct comparison, while facet produces coordinated small multiples on separate panels. Depending on the combination of variable types supplied to x and y, XY() can also create time series, bubble plots, and Cleveland-style dot or lollipop plots.

When running in RStudio, static graphics are drawn in the Plots window. When the default use_plotly=TRUE, the interactive **plotly** version of the scatterplot (including time series) display is also drawn in the Viewer window.

A scatterplot displays the joint values of one or more variables as 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). With a common syntax, XY(x, y, ...) generates a family of related 1- or 2-dimensional relationship plots, possibly enhanced with fitted lines, ellipses, and outlier labeling. Categorical variables should typically be defined as factors, and date variables should be stored as Date objects. When x is of class Date, the display is treated as a time-indexed series.

XY() therefore serves as the relationship-oriented counterpart to Chart() (aggregated values) and X() (univariate distributions) within the three-function visualization framework implemented in **lessR**.

Usage

```
XY(

    # ------------------------------------------------------
    # Data from which to construct the plot for x- and y-axis
    x, y=NULL, data=d, filter=NULL,

    # ------------------------------------------------
    # Stratification: Same panel or Trellis (facet) plot [x, or x and y]
    by=NULL, facet=NULL,
    n_row=NULL, n_col=NULL, aspect="fill",
```

```
# ------------------------------------------------------------
# Types of plots
type=c("scatter", "smooth", "contour"),


# -------------------------------
# Enhancements and customizations
# -------------------------------

# ------------------------------------------------------------------------
# Analogy of physical Marks on paper that create the points and labels
# See  ?style  for more options with the  style()  function
theme=getOption("theme"),
fill=NULL, color=NULL,
transparency=getOption("trans_pt_fill"),

enhance=FALSE, means=TRUE,
size=NULL, size_cut=NULL, shape="circle", line_width=1.5,
segments=FALSE, segments_y=FALSE, segments_x=FALSE,

# ----------------------
# Sort and jitter points
sort=c("0", "-", "+"),
jitter_x=NULL, jitter_y=NULL,

# ----------------
# 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"), fit_lwd=getOption("fit_lwd"),
fit_new=NULL, plot_errors=FALSE, ellipse=0,

# ------------------------------------------------------------------------
# Time series and forecasting, plot x values sequentially [xDate, y or Y]
ts_unit=NULL, ts_ahead=0, ts_method=c("es", "lm"),
ts_source=c("fable", "classic"), ts_agg=c("sum", "mean"),
ts_NA=NULL, ts_format=NULL, ts_fitted=FALSE, ts_PI=.95,
ts_trend=NULL, ts_seasons=NULL, ts_error=NULL,
ts_alpha=NULL, ts_beta=NULL, ts_gamma=NULL, ts_new_x=NULL,
ts_stack=FALSE, ts_area_fill="transparent", ts_area_split=0,
ts_n_x_tics=NULL,
```

```
# Run chart (indicate with .Index for the name of the x-variable)
show_runs=FALSE, center_line=c("off", "mean", "median", "zero"),

# ---------------------------------
# Integrated violin/box/scatter plot  [x only]
vbs_plot="vbs", vbs_ratio=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, n_min_pivot=1,
k=1.5, box_adj=FALSE, a=-4, b=3,

# -----------
# Bubble plot [xCategorical, or xCategorical and yCategorical]
radius=NULL, power=0.5, low_fill=NULL, hi_fill=NULL,

# ------------------------------------------------------------
# Large data sets, smoothing, contours and binning  [x and y]
smooth_points=100, smooth_size=1,
smooth_power=0.25, smooth_bins=128, n_bins=1,
contour_n=10, contour_nbins=50, contour_points=FALSE,

# -----------------------------------------------------
# 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,


# ----------------------
# Axes labels and values
# ----------------------

# ------------------------
# Axis labels and spacing
xlab=NULL, ylab=NULL, main=NULL, sub=NULL,
label_adjust=c(0,0), margin_adjust=c(0,0,0,0),  # top, right, bottom, left
pad_x=c(0,0), pad_y=c(0,0),

# ---------------------
# Axis values specified
scale_x=NULL, scale_y=NULL, origin_x=NULL, origin_y=NULL,

# ---------------------
# Axis values formatted
rotate_x=getOption("rotate_x"), rotate_y=getOption("rotate_y"),
offset=getOption("offset"),
axis_fmt=c("K", ",", ".", ""), axis_x_prefix="", axis_y_prefix="",
xy_ticks=TRUE, n_axis_x_skip=0, n_axis_y_skip=0,
```

```
    # ------
    # legend
    legend_title=NULL,


    # -------------
    # Miscellaneous
    # -------------

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

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

    # ------------------------------------------------------------
    # Deprecated, to be removed in future versions
    n_cat=getOption("n_cat"), value_labels=NULL,  # use R factors instead
    rows=NULL, facet1=NULL, facet2=NULL, smooth=FALSE,
    stat=c("mean", "sum", "sd", "deviation", "min", "median", "max"),
    stat_x=c("count", "proportion", "%"),

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

Arguments

x	The *variable* plotted on the x-axis. 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 the points are plotted sequentially and connected with line segments. If named .Index, a run chart is generated from the corresponding y variable. Can specify multiple x-variables or multiple y-variables as vectors, but not both.
y	The *variable* plotted on the vertical y-axis. Can be continuous or categorical.
data	Optional data frame that contains one or both of x and y. Default data frame is d.
filter	A logical expression or a vector of integers that specify the row numbers to retain, defining a subset of rows of the data frame to analyze.
by	A *grouping variable*, a categorical variable that provides separate group profiles of the primary numeric variables x and, optionally, y on the *same* plot. For two-

variable plots, the same grouping applies to the panels of a **Trellis graphic** if facet1 is specified.

facet — One categorical variable, or a vector of two categorical variables, that activates **facet graphics** (Trellis plots) via the **lattice** package, providing a separate plot of the primary variable(s) x and y for each level of the variable or combination of two levels.

n_row — Optional specification for the number of rows 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 (facet) graphics. Specify n_col or n_row, but not both. The default is 1.

aspect — Lattice parameter for the aspect ratio of the Trellis panels (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 that "banks" to 45 degrees, that is, with the typical line slope approximately 45 degrees.

type — Display type. The default is "scatter" for a scatter plot. Set to "smooth" for a **smoothed density plot** for two numeric variables, or "contour" for a **contour plot** summarizing the bivariate density.

theme — Color theme for this analysis. Make persistent across analyses with style.

fill — Fill color for the points or for the area under a line chart. Can also be set via the lessR function getColors to select from a variety of color palettes. For points, the default is pt_fill; for the area under a line chart, violin_fill. For a line chart, set to "on" to use the default color.

color — Border color of the points or line color for a 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, which is 0.1 for points. Bubbles have a default transparency of 0.3.

enhance — For a two-variable scatterplot, if TRUE, automatically add the 0.95 data ellipse, labels for outliers beyond a Mahalanobis distance of 6 from the ellipse center, the least squares line for all data, the least squares line for the data excluding outliers, and horizontal and vertical lines representing the means of the two variables.

means — If one variable is categorical (factor) and the other is continuous, then if TRUE the group means are plotted in the scatterplot by default. Also applies to a 1-D scatterplot.

size — When set to a constant, the scaling factor for **standard points** (not bubbles) or for a line, with default of 1.0 for points and 2.0 for a line. Set to 0 to suppress plotting of points or lines. If ts_area_fill is used for a line chart, then the 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.

size_cut — For a bubble plot in which bubble sizes are defined by a size variable, controls whether and how many bubble labels are shown. If 1 (or TRUE), show the value of the sizing variable for selected bubbles in the center of the bubbles, except when the bubble is too small. If 0 (or FALSE), no values are displayed. If a number greater than 1, display labels only for the indicated number of values (e.g., the maximum and minimum when set to 2). The color of the displayed text is set by bubble_text from the style function.

shape — Plot character(s). The default value is "circle", with both a default exterior color and filled interior, explicitly specified by "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 integers 0–25 as defined by the base R points function also apply. If plotting levels for different groups according to by, specify a vector of shapes with one shape per group, or set to "vary" to have shapes selected automatically across by groups.

line_width — Width of the line segments that connect adjacent points, such as in time series plots. Set to 0 to remove line segments.

segments — Designed for interaction plots of means: connects each pair of successive points with a line segment. Pass a data frame of means, such as from pivot. To turn off connecting line segments for sorted, equal-interval 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 a 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 — Sort the values of y by the values of x, such as for a Cleveland dot plot (numeric x paired with a categorical y with unique values). If x is a vector of two variables, sort by their difference.

jitter_x — Randomly perturb the plotted points of a scatterplot horizontally within the limits of the specified value, or set to NULL to rely upon the computed default value.

jitter_y — Vertical jitter. Defaults to 0. Same interpretation as jitter_x.

ID — Name of a variable that provides the **labels for selected plotted points for outlier identification**, row names of the data frame by default. To label all points, use the add parameter (see below).

ID_size — Size of the plotted labels. Modify label text color with the style function parameter ID_color.

MD_cut — Mahalanobis distance cutoff to define an outlier in a two-variable scatterplot.

out_cut — Count or proportion of plotted points to label, in order of their distance from the center (means) of the univariate distribution or scatterplot, sorted from most to least extreme. For two-variable plots, distance from the center is based on Mahalanobis distance.

`out_shape` Shape of outlier points in a two-variable scatterplot or VBS plot. Modify fill color from the current `theme` with `out_fill` and `out2_fill` in `style`.

`out_size` Size of outlier points in a two-variable scatterplot or VBS plot.

`fit` Type of **best fit line**. Default is `"off"`. Options include `"loess"` for a non-linear smooth, `"lm"` for a linear least squares model, `"null"` for the null (intercept-only) model, `"exp"` for exponential growth or decay, `"power"` for a general power model in conjunction with `fit_power`, and `"quad"` for a quadratic (power 2) model. If potential outliers are identified according to `out_cut`, a second (dashed) fit line is displayed based on the data *excluding* those outliers.

`fit_power` Power that describes the response `y` as a power function of the predictor `x`, required when `fit = "power"`. Optionally, and experimentally, may apply to `fit = "exp"`.

`fit_se` Confidence level for the error band displayed around the line of best fit. Default is 0.95 when a fit line is specified, but is turned off when `plot_errors = TRUE`. Can be a vector to display multiple confidence bands. Set to 0 to suppress error bands.

`fit_color` Color of the fit line.

`fit_lwd` Line width of the fit line.

`fit_new` When `fit` is set to a fitted curve such as `"lm"` or `"quad"`, predicted values are computed at the `x`-values supplied here.

`plot_errors` If `TRUE`, plot line segments joining 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 one `x`-variable and one `y`-variable according to the contours of the corresponding bivariate normal density. Specify the confidence level(s) as a single number or a vector of values between 0 and 1 to plot one or more ellipses. Ellipses are not available in the interactive `plotly` version. For Trellis graphics, only the maximum level is used, with one ellipse per panel. Modify fill and border colors with the `style` parameters `ellipse_fill` and `ellipse_color`.

`ts_unit` Time unit for plotting **time series data** when `x` is of type `Date`. Default is to infer the unit from the observed date spacing. Specify `ts_unit` to override the inferred unit and, when requested, aggregate to a coarser unit. Valid values are `"days"`, `"weeks"`, `"months"`, `"quarters"`, `"years"`, and `"days7"` (daily data with a 7-day seasonal cycle).

`ts_ahead` Number of `ts_unit` steps ahead of the last time period to forecast.

`ts_method` Forecasting method. Default is `"es"` for exponential smoothing models. Alternatively, choose `"lm"` for a least squares linear regression model that accounts for seasonality.

`ts_source` Source of forecasting functions. Default is `"fable"` for access to `ETS()` and `TSLM()` from the **fable** package and related functions from the **fpp3** ecosystem, or `"classic"` for base R forecasting functions.

`ts_agg` Function used to aggregate over time according to `ts_unit`. Default is `"sum"` with an option for `"mean"`.

ts_NA — By default, missing y values (NA) are not plotted, leaving gaps. Alternatively, specify a value, usually 0, to replace NA and plot that value (e.g., 0) at the corresponding date. Forecasting with missing data is not supported.

ts_format — Optional format string for as.Date() describing the values of the date variable on the x-axis, needed if the function cannot infer the date format. For example, the character string "09/01/2024" can be described by "%m/%d/%Y". See details for more information.

ts_fitted — If TRUE and ts_ahead is used for forecasting, display for each data point the fitted value, level, trend, and seasonal component.

ts_PI — Prediction interval level about the forecasted values, with default of 0.95.

ts_trend — Trend parameter. Default value is NULL, allowing the procedure to choose the specification that yields the best-fitting model for the data. Values "A" and "M" indicate additive and multiplicative models, respectively. Specify "N" to omit the trend component. See the forecasting documentation for valid combinations of trend, season, and error.

ts_seasons — Seasonality parameter. See ts_trend.

ts_error — Error parameter. See ts_trend.

ts_alpha — Exponential smoothing level parameter. Sets the value for HoltWinters(), and acts as a suggestion for ETS() in the "fable" framework.

ts_beta — Exponential smoothing trend parameter. See ts_alpha.

ts_gamma — Exponential smoothing seasonal parameter. See ts_alpha.

ts_new_x — A data frame of predictor variable names and new values for exogenous regressors used for model fitting and forecasting. The number of rows of new data values determines the number of future time periods for which forecasts are generated. Currently applies only to the default "fable" linear model for ts_method = "lm".

ts_stack — If TRUE, multiple time plots are stacked on each other, with area set to TRUE by default.

ts_area_fill — Fill under line segments, if present. If ts_stack is TRUE, the default is a gradient from the default color range (e.g., "blues"). If not specified, and fill is set with no plotted points and ts_area_fill is not specified, then fill generally controls the area under the line segments.

ts_area_split — Applies only to a Trellis plot activated with facet1. Value of y that defines a reference line splitting the filled area under the time series line. Values of y below this threshold lie below the reference line; values above lie above the line.

ts_n_x_tics — Suggested number of ticks for dates on the x-axis, overriding the default of approximately seven ticks.

show_runs — If TRUE, display the individual runs in the run analysis. Also sets run to TRUE. Customize the color of the line segments with segments_color via style.

center_line — Plots a dashed line through the middle of a run chart. Default center line is the "median", suitable when values randomly vary about the median. Alternatively, "mean" and "zero" specify that the center line goes through the mean or zero, respectively. Currently not supported for Trellis plots.

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 in upper- or lower-case. Generalize to Trellis plots with the facet1 and facet2 parameters, but currently only for horizontal displays. Modify fill and border colors from the current theme with the style parameters violin_fill, violin_color, box_fill and box_color.

vbs_ratio — Height of the violin plot relative to the plot area. Make the violin (and the accompanying box plot) larger or smaller by adjusting either the plot area or this value.

bw — Bandwidth for the smoothness of the violin plot. Higher values give smoother plots. Default is to calculate a bandwidth that provides a relatively smooth density estimate.

bw_iter — Number of iterations used to adjust the default R bandwidth for further smoothing of the density estimate. When set, the iterations and corresponding results are displayed.

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 violin is based on the current theme color. To use the pt_fill and pt_color values specified by style, set vbs_pt_fill = "default". Otherwise, set to any desired color.

vbs_mean — Show the mean on the box plot with a strip in 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.

n_min_pivot — For the pivot table underlying a VBS plot over at least a by or facet1 categorical variable, sets the minimum sample size for a group for the corresponding row to be displayed. Default is 1 for all non-empty groups. Set to 0 to view all groups.

k — IQR multiplier that determines 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 a robust measure of skewness according to Hubert and Vandervieren (2008).

a, b — Scaling factors for the adjusted box plot to set the lengths of the whiskers. If explicitly set, box_adj is activated.

radius — Scaling factor of the bubbles in a **bubble plot**, which sets the radius of the largest displayed bubble in inches. To activate bubble scaling, set size to a third variable, or for categorical variables (factors), the bubble sizes represent frequencies.

power — Relative scaling of bubble sizes. The default value of 0.5 scales bubbles so that the area of each bubble is proportional to the corresponding sizing variable. A value of 1 scales bubble radii directly by the sizing variable, increasing the size differences between values.

low_fill	For a categorical variable and the resulting bubble plot, or a matrix of such plots, sets the low end of a color gradient for bubble fill.
hi_fill	For a categorical variable and the resulting bubble plot, or a matrix of such plots, sets the high end of a color gradient for bubble fill.
smooth_points	Number of points superimposed on the density plot in the areas of lowest density to help identify outliers, thereby controlling how dark the smoothed display becomes.
smooth_size	Size of points superimposed on the density plot. Default value is 1, resulting in very small points.
smooth_power	Exponent of the function that maps the density scale to the color scale. Smaller values than the default of 0.25 yield darker plots.
smooth_bins	Number of bins in each direction for the density estimation.
n_bins	Number of bins for a single numeric x-variable when visualizing the mean or median of a numeric y-variable for each bin. Points are plotted as bubbles, with bubble size proportional to the bin sample size, unless size is specified as a constant. Default is 1 (no binning).
contour_n	Number of contour levels in a contour plot, with default of 10.
contour_nbins	Number of bins constructed for each of x and y, forming the 2D grid from which densities are estimated.
contour_points	If TRUE, plot the original scatterplot points in a light gray with white borders, representing the data from which the contour curves are estimated. Color is not affected by other color settings, but point size can be changed with size. Default size is 0.72.
bin	If values = "count", display the frequency distribution for a frequency polygon.
bin_start	Optional starting value of the bins for a **frequency polygon**.
bin_width	Optional specified bin width. Also sets bin = TRUE.
bin_end	Optional value that lies within the last bin, so the actual endpoint of the last bin may be larger than the specified value.
breaks	Method for calculating bins, or an explicit specification of the bins, such as via seq or other options provided by hist. Also sets bin = TRUE.
cumulate	If TRUE, display a cumulative frequency polygon.
xlab, ylab	**Axis labels** for the x- and y-axes. If not specified, labels default to the corresponding variable labels (if present), otherwise to the variable names. If xy_ticks is FALSE, no ylab is displayed. Customize these and related parameters with options such as lab_color in style.
main	Label for the plot title. If corresponding variable labels exist, the title is set by default from those labels.
sub	Subtitle for the graph, below xlab. Not yet implemented.

label_adjust — Two-element vector (x-axis label, y-axis label) that adjusts the positions of axis labels in approximate inches. Positive values move labels away from the plot edge. Not applicable to Trellis graphics.

margin_adjust — Four-element vector (top, right, bottom, left) that adjusts plot margins in approximate inches. Positive values move the corresponding margin away from the plot edge. Can be used with `offset` to move axis values into the expanded margin space. Not applicable to Trellis graphics.

pad_x — Proportion of padding added to the left and right sides of the x-axis. Values range from 0 to 1 for each of the two elements. If only one element is specified, it is applied to both sides.

pad_y — Proportion of padding added to the bottom and top sides of the y-axis. Values range from 0 to 1 for each of the two elements. If only one element is specified, it is applied to both sides.

scale_x — If specified, a vector of three values that define the numeric x-axis: starting value, ending value, and number of intervals.

scale_y — If specified, a vector of three values that define the numeric y-axis: starting value, ending value, and number of intervals.

origin_x — Origin of the x-axis. Default is the minimum value of x, except for time series plots and when `stat` is set to `"count"` or a related option, in which case the origin defaults to 0.

origin_y — Origin of the y-axis. Default is the minimum value of y, except for time series plots and when `stat` is set to `"count"` or a related option, in which case the origin defaults to 0.

rotate_x — **Rotation in degrees of the axis value labels** on the x-axis, usually to accommodate longer labels, typically used with `offset`. If set to 90, labels are perpendicular to the axis and a different placement algorithm is used so that `offset` is not needed.

rotate_y — Rotation in degrees of the axis value labels on the y-axis, usually to accommodate longer labels, typically used with `offset`.

offset — Spacing between axis value labels and the axis. Default is 0.5. Larger values (e.g., 1.0) create additional space for labels, especially when rotated. Can be used with `margin_adjust` to create additional margin space for axis labels and values.

axis_fmt — Numeric format of axis labels for both axes. Default rounds thousands to `"K"` (e.g., 100000 becomes 100K). Alternatives include `","` to insert commas in large numbers with a decimal point, `"."` to insert periods, or `""` to turn off formatting.

axis_x_prefix — Prefix for axis labels on the x-axis, such as `"$"`.

axis_y_prefix — Prefix for axis labels on the y-axis, such as `"$"`.

xy_ticks — Flag indicating whether tick marks and associated **value labels** on the axes are displayed. To rotate axis values, use `rotate_x`, `rotate_y`, and `offset` (see style).

n_axis_x_skip — Particularly useful for Trellis or facet plots with many labels on the x-axis. Specifies the spacing for skipping labels. A value of 0 includes all labels (default). A value of 1 skips every other label, 2 includes every third label, etc. Also consider rotate_x.

n_axis_y_skip — Same as n_axis_x_skip but applies to the y-axis.

legend_title — Title of the legend for a multiple-variable x or y plot.

add — **Overlay one or more objects**, text or geometric figures, on the plot. Possible values include any text (first argument) or "text"; "labels" to label each point with the row name; and geometric objects "rect" (rectangle), "line", "arrow", "v_line" (vertical line), and "h_line" (horizontal line). The value "means" draws vertical and horizontal lines at the respective means. Does not apply to Trellis graphics. Customize with parameters such as add_fill and add_color via style.

x1 — First x-coordinate for each overlaid object; may be "mean_x". Not used for "h_line".

y1 — First y-coordinate for each overlaid object; may be "mean_y". Not used for "v_line".

x2 — Second x-coordinate for each object; may be "mean_x". Used only for "rect", "line", and "arrow".

y2 — Second y-coordinate for each object; may be "mean_y". Used only for "rect", "line", and "arrow".

quiet — If TRUE, suppress text output. Can change the system default with style.

do_plot — If TRUE (default), generate the plot.

use_plotly — If TRUE (default), draw a plotly interactive plot in the RStudio Viewer window in addition to the static plot in the Plots window. Not all options are available in the interactive version, but key options are supported.

pdf_file — If specified, direct PDF graphics to the given file name.

width — Width of the plot window in inches, default 5 except within RStudio, where the default maintains an approximately square plotting area.

height — Height of the plot window in inches, default 4.5 except for 1-D scatterplots and when running in RStudio.

digits_d — Number of significant digits for each displayed summary statistic.

n_cat — Maximum number of unique, equally spaced integer values of a variable for which it will be analyzed as categorical rather than continuous. Default is 0. Use to treat such variables as informal factors. *Deprecated*. Best to convert integer-coded categoricals to factors explicitly.

value_labels — For factors, default labels are the factor levels; for character variables, default labels are the character values. Optionally provide labels for the x-axis to override defaults. If the variable is a factor and value_labels is NULL, then labels are set to the factor levels with spaces replaced by line breaks. If x and y share

	the same scale, labels also apply to the y-axis. Control label size with `axis_cex` and `axis_x_cex` from the lessR `style` function.
`rows`	*Deprecated*. Old parameter name; use `filter`.
`facet1`	*Deprecated*. Old parameter name; use `facet`.
`facet2`	*Deprecated*. Old parameter name; use a two-element vector in `facet`.
`smooth`	*Deprecated*. Use parameter `type`.
`stat`	Apply a **specified aggregation** such as `"mean"` to the numerical `y` variable at each level of a categorical `x` variable. The resulting dot plot (Cleveland plot) is analogous to a bar chart but emphasizes position along a common scale.
`stat_x`	If no `y` variable is specified, constructs a frequency polygon for `x` with access to `bin_width`. Either use the default `"count"` for each bin or `"proportion"`, also indicated by `%`.
`eval_df`	Controls whether to check for an existing data frame and specified variables. Default is `TRUE`, unless the **shiny** package is loaded, in which case it is set to `FALSE` so that Shiny can run. Must be set to `FALSE` when using the pipe operator `%\>%`.
`fun_call`	Function call. Used with **knitr** to pass the function call when obtained from the abbreviated helper `sp`.
`...`	Other parameters for non-Trellis graphics as recognized by base R functions `plot` and `par`, including `cex.main` for the size of the title, `col.main` for the color of the title, and `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 horizontal `x`-axis. A second primary variable, `y`, defines the vertical `y`-axis. Either `x` or `y` (but not both simultaneously) may be a vector of variables. The simplest usage—single `x` and `y`—produces a single scatterplot on one panel. With `type = "scatter"` this is a standard point plot; alternative values such as `"smooth"` and `"contour"` invoke 2-D kernel density and contour displays, respectively.

For numeric primary variables, multiple plots can appear on a single panel in two ways. First, by defining groups with the `by` argument: `by` identifies a categorical grouping variable, and a separate scatterplot layer is drawn for each of its levels. Group levels are distinguished by color and/or shape. By default, colors vary across groups; for two groups, a common pattern is a filled symbol for one group and a point with transparent interior for the other.

Second, multiple numeric `x`-variables or multiple `y`-variables can be supplied as a vector, which produces multiple series on the same panel. This is commonly used for time series overlays and multi-series line plots.

Trellis graphics (facets), from Deepayan Sarkar's (2009) `lattice` package, may also be used. A variable specified with `facet` is a conditioning variable. A single `facet` variable produces one panel for each of its levels; a length-two vector of facet variables produces one panel for each cross-classified combination of their levels. Within each panel, `x` and `y` are plotted as usual, and an additional grouping variable specified with `by` may be applied to all panels. When `x` has at most

1000 unique values, XY() can provide a brief diagnostic of the maximum number of repeated values for each level of facet.

Control the panel dimensions and the overall size of the Trellis display with width and height for the graphics device, n_row and n_col for the number of panels in each direction, and aspect for the panel height-to-width ratio. The plot window is the active graphics device (e.g., the standard R plotting window, RStudio Plots pane, or a pdf file when pdf_file is specified).

CATEGORICAL VARIABLES

Categorical variables have relatively few unique values and are recommended to be often coded as factors. However, categorical variables may also be coded numerically, such as Likert responses from 1 to 5. The structuring arguments by and facet apply when at least one of x or y is numeric, but that numeric variable may represent either a truly continuous scale or a discrete Likert-type scale.

Scatterplots of Likert-type data can be challenging because the number of possible joint values is small. For two five-point scales there are only 25 possible combinations, so many points overlap at the same coordinates. For such situations, jittering, or dot/mean plots (using stat) may provide clearer displays.

DATA

The default input data frame is d. Another data frame can be specified with the data argument. Regardless of the name of the data frame, variables can be referenced directly by name—no need to attach the data frame or use d$name. Referenced variables may live in the data frame, the global environment, or both.

The plotted values can be the raw data themselves, or summaries derived from them. With a single numeric variable, counts or proportions can be plotted on the y-axis, optionally after binning x. For a categorical x-variable paired with a continuous y-variable, summary statistics such as means can be plotted at each level of x. When x is continuous and binning is desired, XY() uses the same binning parameters as Histogram, such as bin_width, to override defaults. The stat parameter controls what is plotted (e.g., "data", "mean", "median"). By default, connecting line segments are drawn, yielding a frequency polygon. Turn off line segments by setting line_width = 0.

The filter parameter subsets rows (cases) of the input data frame according to a logical expression or a vector of integers specifying row numbers to retain. Use standard R logical operators as described in Logic—for example, & (and), | (or), and ! (not)—and relational operators as described in Comparison such as == (equality), != (not equal), and > (greater than). To specify rows directly, create an integer vector using standard R syntax. See Examples.

VALUE LABELS

Deprecated. Use factor() *instead.*

The value_labels option can override the axis tick labels with user-supplied values. This is particularly useful for Likert-style data coded as integers: for example, a data value 0 can be displayed as "Strongly Disagree". These labels apply to integer categorical variables and to factor variables. Any spaces in a label are translated into line breaks to improve readability.

In current workflows, the recommended approach is to define factors with the desired labels directly, typically via factors, which allows convenient creation of labeled factors for one or more

variables in a single statement.

VARIABLE LABELS

Base R does not natively support variable labels, but lessR stores labels in the data frame alongside the variables, typically created by Read or VariableLabels. When variable labels exist, XY() uses them by default for axis labels and in text output. Otherwise, the variable names are used.

TWO VARIABLE PLOT

With two variables specified, XY() behaves as follows. If the values of x are unsorted, have unequal intervals, or there is missing data in either variable, a standard scatterplot is produced (for type = "scatter"). When x is sorted with equal intervals and there are no missing values, the default display connects adjacent points with line segments, yielding a function or time-series style plot.

Supplying multiple continuous x-variables against a single y, or vice versa, produces multiple series on the same graph. The points for the second series reuse the first series' color but with transparent fill; for more series, additional colors from the current theme are used.

SCATTERPLOT MATRIX

If x is a vector of continuous variables and y is omitted, XY() generates a scatterplot matrix. The matrix adopts the current color theme. Individual colors (e.g., fill, color) can be overridden. The lower triangle shows the pairwise scatterplots, and the upper triangle shows the corresponding correlation coefficients. By default a non-linear loess fit line is added to each scatterplot; the fit parameter can be used to request a linear least squares line instead, along with fit_color to set its color.

SIZE VARIABLE

Specifying a numeric size variable activates a bubble plot in which the area of each bubble is determined by the corresponding value of size (modified by radius and power).

To vary shapes explicitly across groups, use shape and provide a vector of values (e.g., created with c). One shape is used for each level of the grouping variable in by. To vary colors, use fill; if fill is specified without shape, colors vary but shapes do not. To vary both, specify both shape and fill with values for each by level.

Beyond the named shapes such as "circle", any single character (letters, digits, "+", "*", "#") may be used as a plotting symbol. Within a single specification, either use standard named shapes or single characters, but not both.

SCATTERPLOT ELLIPSE

For a scatterplot of two numeric variables, the ellipse argument requests one or more data ellipses, based on the contours of the corresponding bivariate normal density. For a single x- and y-variable pair, setting ellipse to a numeric value between 0 and 1 (e.g., 0.95) draws the corresponding probability contour. A vector of values produces multiple ellipses. XY() expands the axes as needed so that ellipses extending beyond the range of the data remain fully visible. For Trellis graphics, only the largest ellipse level is used (one ellipse per panel). Ellipse fill and border colors are controlled via ellipse_fill and ellipse_color in style.

TIME CHARTS

See https://web.pdx.edu/~gerbing/lessR/examples/Time.html for additional examples and explanation.

Run Chart:

When x is the special variable .Index, XY() produces a run chart of y. The y-values are plotted against their index positions (1, 2, ...), and run-chart diagnostics such as center lines and runs analysis can be displayed via the corresponding arguments.

Time Series Defined with Dates:

If x is of type Date or an R time-series object, XY() produces a time series plot for each specified variable. Time-series data can be supplied in "long" format (a single column of values plus a date column) or in "wide" format (multiple time-series columns sharing a date index). XY() will attempt to convert character string date values (e.g., "08/18/1952") to Date via as.Date(), using default date formats when possible.

XY() makes a reasonable attempt to decode common date formats, but some formats (e.g., those with month names rather than numbers) may require an explicit format string via ts_format. If the default conversion fails or is ambiguous, specify the correct format using examples such as those in the table below. Note that ts_format can always be provided for explicit specification of the date format.

Example Date	Format
"2022-09-01"	"%Y-%m-%d"
"2022/9/1"	"%Y/%m/%d"
"2022.09.01"	"%Y.%m.%d"
"09/01/2022"	"%m/%d/%Y"
"9/1/22"	"%m/%d/%y"
"September 1, 2022"	"%B %d, %Y"
"Sep 1, 2022"	"%b %d, %Y"
"20220901"	"%Y%m%d"

XY() also converts character-string dates such as "2024 Aug" and "2024 Q1", interpreting three-letter month abbreviations and quarter codes Q1–Q4.

Time unit and aggregation. The ts_unit argument controls the time unit used for plotting and, when specified, can also be used to aggregate the series to a coarser time unit. By default, XY() attempts to infer a regular time unit from the observed spacing of the dates (see 'Detecting the Regularity of Dates'). You may specify ts_unit to override the inferred unit, including treating irregular dates as a regular unit for aggregation (for example, ts_unit="weeks"). The special value "days7" retains daily observations but evaluates seasonality on a 7-day cycle.

When aggregation is requested, the date index is aggregated with endpoints() and period.apply() from the xts package. The aggregation function is specified with ts_agg, default "sum", with "mean" as a common alternative.

Detecting the Regularity of Dates (ts_unit):

For time series operations, **lessR** attempts to infer the regularity of a date variable from the observed sequence of dates. The inferred unit is returned internally as `tu_exist` and is one of `"days"`, `"weeks"`, `"years"`, `"quarters"`, `"months"`, or `"unknown"`.

The decision rules are:

1. **Pre-processing.** Dates are sorted and reduced to unique values. When a grouping variable is present, regularity is evaluated from the first level only (assuming dates are already time-sorted within that level). Day gaps are computed as positive differences between successive dates.
2. **Trivial / short series.** If there are fewer than two dates, or if no positive gaps are available, the unit is set to `"days"`.
3. **Fine units (gap-based).** Let `gap` be the median of the positive day gaps (median is robust to occasional missing dates).
   - If `gap == 1`, then `tu_exist = "days"`.
   - If `gap == 7`, then the series is a *candidate* weekly series. The gap pattern is further examined to distinguish a regular weekly series (primarily 7-day gaps, with only occasional longer gaps) from an irregular reporting schedule (for example, many 14-day gaps). If judged sufficiently weekly, then `tu_exist = "weeks"`; otherwise `tu_exist = "unknown"`.
4. **Coarse units (calendar-based).** If the unit is still `"unknown"` after the fine-unit checks, the dates are examined for calendar structure. Let `year`, `month`, and `day` denote the corresponding calendar fields.
   - **Years.** If there is at most one observation per year and all observations share the same month and day-of-month (for example, January 1 each year), then `tu_exist = "years"`. Missing years are permitted.
   - **Quarters.** If there is at most one observation per calendar quarter (unique year–quarter combinations) *and* the median day gap between observations falls in the range 60–122 days, then the existing time unit is deemed as "quarters"

   . The reporting month and day-of-month may vary across quarters, and missing quarters are permitted. The gap range excludes sparse or irregular series that happen to have at most one observation per quarter by chance.
5. **Months.** If the time variable is a Date but the observations occur exactly once per calendar month with no duplicates within a month and the sequence of months is regular (missing months become explicit NAs if present), then lessR treats the series as monthly and converts the index accordingly.
6. **Missing periods.** When a regular unit is accepted (`"weeks"`, `"years"`, `"quarters"`, or `"months"`), a flag `has_missing_periods` is set to `TRUE` if the sequence of inferred periods contains gaps (for example, a 14-day gap in an otherwise weekly series, or a missing month in an otherwise monthly series). If the series is classified as `"unknown"`, `has_missing_periods` is not used.

If the user explicitly specifies `ts_unit`, that value is used for aggregation and modeling, overriding the inferred unit. In particular, a user-specified `ts_unit = "months"` may treat one observation per month as monthly even if the reporting day varies across months, and a user-specified `ts_unit = "weeks"` may treat dates as weekly even if the inferred regularity is `"unknown"`.

Missing Data:

For missing data, if a date is present but the corresponding y-value is NA, the line is broken at that point (no segment is drawn). If both the date and value are absent (the entire row is missing), the line connects the nearest observed dates, spanning the gap in calendar time but skipping the missing tick label. For example, if "2021-01-07" and "2021-01-09" are present but "2021-01-08" is absent, the plot includes points for the 7th and 9th connected by a line; the 8th does not appear on the axis.

Forecasting:

Forecasting is activated by setting ts_ahead to a positive integer. The trend, seasonal, and error components for exponential smoothing and regression models follow the fable conventions and are controlled by ts_trend, ts_seasons, and ts_error. XY() supports four forecasting engines: ETS() and TSLM() from fable via ts_source = "fable", and classic HoltWinters() and regression decomposition (stl() + lm()) via ts_source = "classic". Multiplicative components require positive data. Additive trend is linear, while multiplicative trend corresponds to exponential growth or decay; additive and multiplicative seasonality scale the seasonal pattern differently as the series level changes.

Exponential smoothing via ETS() (the default when ts_source = "fable") generally provides flexible model specification and often superior predictive performance compared to classic Holt–Winters. For regression with seasonality, TSLM() and the stl()+lm() approach both rely on least squares but handle seasonal structure differently. Setting ts_fitted = TRUE provides fitted values and decomposed trend and seasonal components for inspection.

2-D KERNEL DENSITY

Set type = "smooth" to invoke smoothScatter and display a 2-D kernel density estimate for large data sets. The display respects the current theme. The smooth_points argument controls how many points from low-density regions are superimposed, smooth_bins sets the number of bins in each direction for density estimation, and smooth_power controls the transformation from density scale to color scale. Larger smooth_power values reduce saturation in low-density regions. These arguments map directly to nrpoints, nbin, and transformation in smoothScatter. Grid lines are disabled by default for smooth density plots, but can be re-enabled via the appropriate styling options (e.g., grid_color in style).

Alternatively, set type = "contour" to plot contour lines of the estimated bivariate density, with the level resolution controlled by contour_n and contour_nbins.

COLORS

A global color theme for XY() and other lessR graphics can be set with style (e.g., style(theme = "lightbronze")). The default theme is "lightbronze". A grayscale theme is available via "gray", and other themes (e.g., "sienna", "darkred") are described in style. The sub_theme = "black" option yields a black background with partially transparent plotted colors.

Individual graphical elements (points, lines, panels, grid lines, etc.) can be customized with additional style arguments, such as panel_fill. For a subtle warm background, try panel_fill = "snow"; very light grays such as "gray99" or "gray97" provide a neutral tone.

For many color options, the value "off" is equivalent to "transparent", removing the corresponding fill or borde

See showColors for a display of all named R colors and their RGB values.

ANNOTATIONS

The add argument and its related coordinates (x1, y1, x2, y2) annotate the plot with text and geometric objects. Each object type requires a specific set of coordinates, summarized below. x-coordinates may take the special value "mean_x" and y-coordinates may take "mean_y".

Value	Object	Required Coordinates
"text"	text	x1, y1
"point"	point	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"	horizontal and vertical lines at the means	

The value of add specifies the object type. For a single object, provide one value and its required coordinates. For multiple placements of the same object, supply coordinate vectors. For multiple different objects, supply a vector of add values and, for each coordinate argument (x1, y1, x2, y2), a vector whose elements correspond to the sequence of objects in add.

Styling options such as add_color, add_fill, and transparency may also be given as vectors, allowing different objects to have different colors or other properties.

PDF OUTPUT

To obtain pdf output, set pdf_file to a file name, optionally with width and height to control device size. The pdf is written to the current working directory, which can be explicitly set with setwd.

ADDITIONAL OPTIONS

Many commonly used graphical parameters from the base R function plot are passed through by XY(), including:

cex.main, col.lab, font.sub, etc. Settings for main/sub-titles and axis annotation; see title and par.

main Main title of the graph; see title.

xlim Limits of the x-axis, expressed as c(x1, x2). Note that x1 > x2 is allowed and produces a reversed axis.

ylim Limits of the y-axis.

ONLY VARIABLES ARE REFERENCED

A referenced variable in a lessR plotting function such as XY must be a variable name (or vector of variable names), not an arbitrary expression. The variable must exist either in the referenced data frame (e.g., the default d) or in the user's workspace (global environment). Expressions are not

evaluated in place. For example:

```
> XY(rnorm(50), rnorm(50)) # does NOT work
```

Instead, create named objects and reference them directly:

```
> X <- rnorm(50)   # create vector X in user workspace
> Y <- rnorm(50)   # create vector Y in user workspace
> XY(X, Y)         # directly reference X and Y
```

Value

The output may be assigned to an R object; otherwise it is printed directly to the console. Each component appears only when the corresponding analytic option is activated. For example, outlier identification must be enabled (e.g., via MD_cut) for out_outliers to be included in the output.

To save the results, assign the output to an object, such as p <- XY(Years, Salary). Use names(p) to view the available components, and access any one by prefixing with p$, for example p$out_stats. Output can be viewed interactively at the console or inserted into R~Markdown documents for reproducible reporting.

READABLE OUTPUT

out_stats: Correlational analysis.
out_outliers: Mahalanobis Distance values for detected outliers.
out_frcst: Forecasted values.
out_fitted: Fitted values for the observed data.
out_coefs: Linear and seasonal coefficients from forecasting models.
out_smooth: Smoothing parameters from exponential smoothing models.
out_bubble: Bubble-plot settings, including radius and power.
out_reg: Regression statistics produced when fit is specified.
out_parm: Parameter settings for a VBS plot of a continuous variable.
out_pivot: Pivot table for VBS plots based on any combination of by and facet variables.
out_by: Pivot table for VBS plots aggregated by the by variable.
out_facet1: Pivot table for VBS plots aggregated by the first facet variable.
out_facet2: Pivot table for VBS plots aggregated by the second facet variable.

STATISTICS

outliers: Row numbers corresponding to detected 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.

Gerbing, D. W. (2023). R Data Analysis without Programming, 2nd edition, Chapter 10, 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.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). Melbourne, Australia: OTexts. Retrieved from https://otexts.com/fpp3/

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

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC. URL: https://plotly.com/r/

See Also

X, Chart, 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
#--------------------------------------------------

# scatterplot with all defaults
XY(Years, Salary)
# or use Plot in place of XY, the older method

XY(Years, Salary, by=Gender, size=2, fit="lm",
     fill=c(M="olivedrab3", W="gold1"),
     color=c(M="darkgreen", W="gold4"))

# maximum information, minimum input: scatterplot +
#  means, outliers, ellipse, least-squares lines with and w/o outliers
XY(Years, Salary, enhance=TRUE)

# extend x and y axes
XY(Years, Salary, scale_x=c(-10, 35, 10), scale_y=c(0,200000,10))

XY(Years, Salary, add="Hi", x1=c(12, 16, 18), y1=c(80000, 100000, 60000))

d <- factors(Gender, levels=c("M", "W"))
XY(Years, Salary, facet=Gender)
d <- dd
```

```
# just males employed more than 5 years
XY(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 <- XY(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="powderblue")
XY(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")
XY(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
XY(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")
XY(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)
XY(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
XY(Years, Salary, fit="loess", span=1.25)

# 2-D kernel density (more useful for larger sample sizes)
XY(Years, Salary, type="smooth")
```

```
#-------------------------------------------------------
# scatterplot matrix from a vector of numeric variables
#-------------------------------------------------------

# with least squares fit line
XY(c(Salary, Years, Pre), 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"))
XY(Years, Salary, facet=Gender)
d <- Read("Employee", quiet=TRUE)



# all three by (categorical) variables
XY(Years, Salary, facet=c(Dept, Gender), by=Plan, n_axis_y_skip=1)

# vary both shape and color with a least-squares fit line for each group
style(color=c("darkgreen", "brown"))
XY(Years, Salary, facet=Gender, fit="lm", shape=c("W","M"), size=.8)
style("gray")

# compare the men and women Salary according to Years worked
#   with an ellipse for each group
XY(Years, Salary, by=Gender, ellipse=.50)




# time charts
#------------
# run chart, with default fill area
XY(.Index, Salary, ts_area_fill="on")

# two run charts in same panel
# or could do a multivariate time series
XY(.Index, c(Pre, Post))

# Trellis graphics run chart with custom line width, no points
XY(.Index, Salary, facet=Gender, line_width=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)
XY(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)
XY(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)
XY(month, x, ts_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
XY(Month, Price, filter=(Company=="Apple"))
# Trellis plots, one for each company
XY(Month, Price, facet=Company, n_col=1)
# all three plots on the same panel, three shades of blue
XY(Month, Price, by=Company, color="blues")
# exponential smoothing forecast for next 12 months,
#   aggregate monthly data by mean over quarters
XY(Month, Price, ts_ahead=12, ts_unit="quarters")

#-----------------------------------------
# analysis of a single categorical variable
#-----------------------------------------
d <- rd("Employee")

# default 1-D bubble plot
# frequency plot, replaces bar chart
XY(Dept)



# plot of frequencies for each category (level), replaces bar chart
XY(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)
XY(Dept, Salary)
style()


#-------------------
# Cleveland dot plot
#-------------------

# standard scatterplot
XY(Salary, row_names, segments_y=FALSE)

# Cleveland dot plot with two x-variables
XY(c(Pre, Post), row_names)


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

# add text at the one location specified by x1 and x2
XY(Years, Salary, add="Hi There", x1=12, y1=80000)
# add text at three different specified locations
XY(Years, Salary, add="Hi", x1=c(12, 16, 18), y1=c(80000, 100000, 60000))

# add three different text blocks at three different specified locations
XY(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
XY(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
XY(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)
XY(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
XY(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))
XY(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 <- Read("Mach4", quiet=TRUE)  # Likert data, 0 to 5
XY(m06, m07)




#---------------
# 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
XY(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)
XY(x, c(y1, y2))
```

Index

∗ .
  ., 4
∗ **bar chart**
  Chart, 10
  CountAll, 51
∗ **binomial process**
  simFlips, 137
∗ **bubble chart**
  Chart, 10
∗ **central limit theorem**
  simCLT, 135
∗ **chart**
  Chart, 10
∗ **color**
  Chart, 10
  getColors, 70
  showColors, 132
  showPalettes, 133
  X, 179
  XY, 196
∗ **confidence interval**
  simCImean, 134
  simMeans, 138
∗ **correlation**
  corCFA, 28
  corEFA, 35
  corPrint, 37
  corProp, 38
  corRead, 40
  corReflect, 42
  corReorder, 46
  corScree, 49
∗ **csv**
  Correlation, 43
  details, 64
  label, 78
  Read, 103
  style, 144
  VariableLabels, 174
  Write, 176
∗ **datasets**
  dataAnova_1way, 52
  dataAnova_2way, 53
  dataAnova_rb, 53
  dataAnova_rbf, 54
  dataAnova_sp, 55
  dataBodyMeas, 56
  dataCars93, 57
  dataEmployee, 58
  dataEmployee_lbl, 58
  dataFreqTable99, 59
  dataJackets, 60
  dataLearn, 60
  dataMach4, 61
  dataMach4_lbl, 62
  dataReading, 63
  dataStockPrice, 63
  dataWeightLoss, 64
∗ **descriptive**
  CountAll, 51
∗ **factor analysis**
  corCFA, 28
  corEFA, 35
∗ **grouping variable**
  XY, 196
∗ **hcl**
  getColors, 70
∗ **histogram**
  CountAll, 51
  X, 179
∗ **hplot**
  savePlotly, 129
∗ **kurtosis**
  kurtosis, 77
∗ **labels**
  label, 78
  VariableLabels, 174
∗ **logit**

Logit, 79
* **long-form**
rename, 124
reshape_long, 126
train_test, 161
* **merge**
Merge, 84
* **names**
to, 160
* **nested models**
Nest, 87
* **pie chart**
Chart, 10
* **pivot**
pivot, 91
* **plot**
XY, 196
* **power**
ttestPower, 171
* **print.out_all**
print.out_all, 96
* **print.out**
print.out, 95
* **print**
values, 173
* **probability**
prob_norm, 97
prob_tcut, 98
prob_znorm, 100
* **proportionality**
corProp, 38
* **proportion**
Prop_test, 101
* **radar chart**
Chart, 10
* **read**
details, 64
Read, 103
* **recode**
recode, 109
* **regPlot**
regPlot, 111
* **regression**
ANOVA, 5
Logit, 79
Model, 86
Regression, 113
* **rescale**
rescale, 125
* **reshape**
rename, 124
reshape_long, 126
reshape_wide, 128
train_test, 161
* **scree**
corScree, 49
* **see**
see, 131
* **sets**
style, 144
* **skew**
skew, 140
* **smd**
ttest, 165
* **sort**
order_by, 90
* **stl**
STL, 141
* **subset**
Subset, 153
* **summary**
SummaryStats, 155
* **sunburst**
Chart, 10
* **t-cutoff**
prob_tcut, 98
* **t-distribution**
prob_tcut, 98
* **t.test**
ttest, 165
ttestPower, 171
* **time series**
STL, 141
* **transform**
Transform, 162
* **treemap**
Chart, 10
* **values**
values, 173
* **wide-form**
rename, 124
reshape_long, 126
reshape_wide, 128
train_test, 161
* **write**
Correlation, 43

Write, 176
., 4, *153*

aggregate, *91*, *93, 94*
ANOVA, 5, *86*
anova, *82, 83*, *87–89*, *117*, *123*
aov, *5*, *7*, *9*
attach, *185*
av *(*ANOVA*)*, 5
av_brief *(*ANOVA*)*, 5

BarChart, *51*
BarChart *(*Chart*)*, 10
barplot, *14*, *17*, *23*
binom.test, *103*
BoxPlot *(*X*)*, 179
boxplot, *158*
bx *(*X*)*, 179

c, *12*, *31*, *42*, *48*, *90*, *93*, *120*, *154*, *156, 157*, *181*, *185*, *210*
ca *(*CountAll*)*, 51
cfa *(*corCFA*)*, 28
Chart, 10, *69, 70*, *185*, *188*, *216*
chisq.test, *17*
class, *8*, *36*, *83*, *89*, *122*
Comparison, *7*, *18*, *82*, *118*, *185*, *209*
confint, *82, 83*, *87*, *117*, *123*
cooks.distance, *82, 83*, *87*, *117*, *123*
cor, *41*, *44*, *117*
cor.test, *44*, *46*
corCFA, 28, *35*, *42*
corEFA, 35, *49*
corPrint, 37
corProp, 38
corRead, *31*, 40, *104*, *108*
corReflect, 42
Correlation, *29*, *31*, *33*, *36–38*, *40, 41*, *43*, 43, *48*, *50*
corReorder, *38*, 46
corScree, 49
CountAll, 51, *186*
cov, *44*, *46*
cp *(*corProp*)*, 38
cr *(*Correlation*)*, 43
cr_brief *(*Correlation*)*, 43

dataAnova_1way, 52
dataAnova_2way, 53
dataAnova_rb, 53
dataAnova_rbf, 54
dataAnova_sp, 55
dataBodyMeas, 56
dataCars93, 57
dataEmployee, 58
dataEmployee_lbl, 58
dataFreqTable99, 59
dataJackets, 60
dataLearn, 60
dataMach4, 61
dataMach4_lbl, *61*, 62
dataReading, 63
dataStockPrice, 63
dataWeightLoss, 64
db *(*details*)*, 64
Density *(*X*)*, 179
density, *169*
details, 64, *103*, *108*
dnorm, *100, 101*

efa *(*corEFA*)*, 35
Extract, *4, 5*, *132*

factanal, *35, 36*, *49*
factor, *109, 110*, *154*, *163*
factors, 66, *209*
fitted, *82, 83*, *87*, *117*, *123*
Flows, 68
formula, *6*, *79, 80*, *83*, *86, 87*, *113, 114*, *123*, *158*, *166, 167*, *169*

getColors, *14*, *20*, *23*, 70, *73*, *147*, *150*, *185, 186*, *188*, *200*
glm, *79*, *81–83*, *87–89*

hcl, *74*
hcl.colors, *73, 74*
hclust, *48*
hist, *184*, *186*, *205*
Histogram, *51*, *66*, *107*, *117*, *119*, *175*, *209*
Histogram *(*X*)*, 179
hs *(*X*)*, 179

interact, 76
interaction.plot, *7*, *9*

kurtosis, 77

label, *66*, 78, *103*, *107, 108*, *175*

legend, *16*, *17*, *20*, *23*
lm, *7*, *86–89*, *112*, *113*, *117*, *123*, *136*
Logic, *7*, *17*, *82*, *118*, *185*, *209*
Logit, 79, *86*
lowess, *119*
lr *(*Logit*)*, 79

Merge, 84
merge, *84*, *85*
Model, 86
model, *80*, *165*
model *(*Model*)*, 86
model_brief *(*Model*)*, 86
mrg *(*Merge*)*, 84

names, *161*
Nest, 87, *123*
nt *(*Nest*)*, 87

options, *82*, *121*, *150*, *152*
order, *90*, *91*
order_by, 90

palette.colors, *74*
par, *17*, *184*, *208*, *214*
paste, *161*
paste0, *160*
pc *(*Chart*)*, 10
pdf, *50*
PieChart *(*Chart*)*, 10
pivot, 91, *155*, *201*
Plot, *172*
Plot *(*XY*)*, 196
plot, *98*, *100*, *101*, *172*, *208*, *214*
plot.density, *169*
pnorm, *98*, *99*
points, *201*
power.t.test, *172*
predict, *82*, *117*
print, *173*
print.out, 95
print.out_all, 96
prob_norm, 97
prob_tcut, 98
prob_znorm, 100
prop *(*Prop_test*)*, 101
Prop_test, 101

qt, *99*

rbind, *84*, *85*
rbinom, *138*
rd *(*Read*)*, 103
rd.cor *(*corRead*)*, 40
rd_lbl *(*Read*)*, 103
Read, *21*, *32*, *51*, *66*, *78*, *79*, *82*, 103, *106*, *114*, *115*, *118*, *121*, *157*, *168*, *173–175*, *178*, *186*, *210*
read.csv, *108*
read.fwf, *108*
read.table, *40*, *41*, *65*
Read2 *(*Read*)*, 103
recode, *43*, 109, *125*
reflect *(*corReflect*)*, 42
reg *(*Regression*)*, 113
reg_brief *(*Regression*)*, 113
regPlot, *7*, 111, *117*, *119*, *123*
Regression, *86*, *95*, *96*, *104*, *107*, *111*, *113*, 113, *174*, *175*
rename, 124
reord *(*corReorder*)*, 46
rescale, *120*, 125
reshape, *126–129*
reshape_long, 126
reshape_wide, 128
resid, *82*, *83*, *87*, *117*, *123*
residuals, *82*
row.names, *154*
rstudent, *82*, *83*, *87*, *117*, *123*

save, *178*
savePlotly, *23*, 129
saveWidget, *130*
scale, *120*, *126*
scales *(*corCFA*)*, 28
ScatterPlot, *117*, *119*, *171*, *172*
ScatterPlot *(*XY*)*, 196
scree *(*corScree*)*, 49
see, 131
seq, *120*, *186*, *205*
setwd, *21*, *50*, *168*, *187*, *214*
showColors, *73*, *74*, 132, *213*
showPalettes, *73*, 133
simCImean, 134
simCLT, 135
simFlips, 137
simMeans, 138
skew, 140
smoothScatter, *213*

sp (XY), 196
ss (SummaryStats), 155
ss_brief (SummaryStats), 155
STL, 141
stl, *143*
style, *6*, *14–17*, *20*, *32*, *44*, *47*, *72*, *81*, *83*, *84*, *86*, *90*, *93*, *105*, *109*, *115*, *117*, *121*, 144, *150*, *154*, *156*, *163*, *166*, *167*, *174*, *177*, *181*, *182*, *185*, *188*, *200–208*, *210*, *213*, *216*
Subset, 153
subset, *5*, *153*, *154*
summary, *82*, *117*, *158*
summary.glm, *83*
summary.lm, *87*, *123*
SummaryStats, *51*, 155

t.test, *165*, *167*, *169*
table, *23*
title, *214*
to, *41*, 160
train_test, 161
Transform, 162
transform, *110*, *162*, *163*
tt (ttest), 165
tt_brief (ttest), 165
ttest, *86*, 165
ttestPower, *169*, 171
ttp (ttestPower), 171
TukeyHSD, *7*, *9*

unclass, *8*, *36*, *89*, *122*, *158*

values, 173
VariableLabels, *21*, *78*, *103*, *107*, *108*, 174, *210*
ViolinPlot (X), 179
vl (VariableLabels), 174
vp (X), 179

with, *185*
Write, *106*, 176
write.csv, *178*
write.table, *177*, *178*
wrt (Write), 176
wrt_r (Write), 176
wrt_x (Write), 176

X, *23*, 179, *216*
xAnd, 191
xNum, 192
xP, 193
xRow, 193
xU, 194
xW, 195
XY, *23*, *188*, 196