Package 'weights'

May 8, 2026

Title Weighting and Weighted Statistics

Version 1.1.2

Date 2025-06-18

Imports Hmisc, mice, gdata, stats, graphics, utils, lme4

Suggests pscl, vioplot, glmnet, nnet, MASS, mgcv

Description Provides a variety of functions for producing simple weighted statistics, such as weighted Pearson's correlations, partial correlations, Chi-Squared statistics, histograms, and t-tests as well as simple weighting graphics including weighted histograms, box plots, bar plots, and violin plots. Also includes software for quickly recoding survey data and plotting estimates from interaction terms in regressions (and multiply imputed regressions) both with and without weights and summarizing various types of regressions. Some portions of this package were assisted by AI-generated suggestions using OpenAI's GPT model, with human review and integration.

License GPL (>= 2)

Encoding UTF-8

LazyData true

NeedsCompilation yes

Author Josh Pasek [aut, cre] (ORCID: <https://orcid.org/0000-0001-6099-6119>),
Alex Tahk [ctb] (ORCID: <https://orcid.org/0000-0001-7895-9420>),
Gene Culter [ctb],
Marcus Schwemmle [ctb],
Some code modified from R-core [ctb]

Maintainer Josh Pasek <josh@joshpasek.com>

Repository CRAN

Date/Publication 2025-06-18 16:10:02 UTC

Contents

`anes04` *Demographic Data From 2004 American National Election Studies (ANES)*

Description

A dataset containing demographic data from the 2004 American National Election Studies. The data include 5 variables: "female" (A Logical Variable Indicating Sex), "age" (Numerically Coded, Ranging From 18 to a Topcode of 90), "educats" (5 Education Categories corresponding to 1-Less than A High School Degree, 2-High School Gradutate, 3-Some College, 4-College Graduate, 5-Post College Education), "racecats" (6 Racial Categories), and "married" (A Logical Variable Indicating the Respondent's Marital Status, with one point of missing data). Dataset is designed show how production of survey weights works in practice.

Usage

```
data(anes04)
```

Format

The format is: chr "anes04"

Source

http://www.electionstudies.org

`coeffer` *Extract model coefficients with standard errors and significance stars*

Description

`coeffer` is a generic function to extract estimates, standard errors, p-values, and significance stars from a fitted model object. It supports a variety of common model types including linear, generalized linear, ordinal, mixed-effects, additive, penalized, and multinomial regression models.

Usage

```
coeffer(x, digits = 2, vertical = TRUE, approx.p = FALSE, s = "lambda.1se", ...)
```

Arguments

`x`	A fitted model object. Supported classes include `lm`, `glm`, `polr`, `lmerMod`, `gam`, `glmnet`, and `multinom`.
`digits`	Number of digits to retain in internal rounding (used for formatting).
`vertical`	Logical; included for compatibility, but not used by most methods.
`approx.p`	Logical; if `TRUE`, attempts to compute approximate p-values for models that do not provide them (e.g., `lmerMod`). Defaults to `FALSE`.
`s`	Sets `s = "lambda.1se"` or sets `s` to other value for `glmnet` models.
`...`	Additional arguments passed to methods.

Details

For models that do not provide p-values (e.g., `lmer`, `glmnet`), `approx.p = TRUE` will attempt to calculate approximate p-values using standard normal approximations based on the ratio of estimate to standard error. This should be used with caution.

`multinom` models return a list of coefficient sets, one for each outcome level.

Value

A list with the following components (or a list of such lists for `multinom` models):

- `rn` — Coefficient names
- `est` — Point estimates
- `ses` — Standard errors
- `pval` — P-values, where available (otherwise `NA`)
- `star` — Significance stars based on p-values
- `cps` — Cutpoint names for ordinal models (otherwise `NULL`)

Author(s)

Josh Pasek

See Also

summary, onetable, pR2, polr, multinom, lmer, gam, glmnet

Examples

```
data(mtcars)
mod1 <- lm(mpg ~ wt + hp, data = mtcars)
coeffer(mod1)

mod2 <- glm(am ~ wt + hp, data = mtcars, family = binomial)
coeffer(mod2)

library(MASS)
mod3 <- polr(Sat ~ Infl + Type + Cont, data = housing)
coeffer(mod3)

library(lme4)
mod4 <- lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
coeffer(mod4)
coeffer(mod4, approx.p = TRUE)

library(mgcv)
mod5 <- gam(mpg ~ s(wt) + hp, data = mtcars)
coeffer(mod5)

library(glmnet)
x <- model.matrix(mpg ~ wt + hp, data = mtcars)[, -1]
y <- mtcars$mpg
mod6 <- glmnet(x, y)
coeffer(mod6, s = mod6$lambda.min)

library(nnet)
mod7 <- multinom(vs ~ wt + hp, data = mtcars, trace = FALSE)
coeffer(mod7)
```

---

dummify *Separate a factor into separate dummy variables for each level.*

Description

dummify creates a matrix with columns signifying separate dummy variables for each level of a factor. The column names are the former levels of the factor.

Usage

```
dummify(x, show.na=FALSE, keep.na=FALSE)
```

Arguments

x	x is a factor the researcher desires to split into separate dummy variables.
show.na	If show.na is 'TRUE', output will include a column idicating the cases that are missing.
keep.na	If keep.na is 'TRUE', output vectors will have "NA"s for cases that were originally missing.

Value

dummify returns a matrix with a number of rows equal to the length of x and a number of columns equal to the number of levels of x.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

Examples

```
data("anes04")

anes04$agecats <- cut(anes04$age, c(17, 25,35,45,55,65, 99))
levels(anes04$agecats) <- c("age1824", "age2534", "age3544",
          "age4554", "age5564", "age6599")

agedums <- dummify(anes04$agecats)
table(anes04$agecats)
summary(agedums)
```

---

findn *Summarize key model information including sample size and fit statistics*

Description

findn is a generic function that extracts useful summary information from a model object. It supports linear models (lm), generalized linear models (glm), ordinal regression models from polr, mixed-effects models from lmer, generalized additive models from gam, and multinomial models from multinom.

Usage

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

Arguments

x	A fitted model object of class lm, glm, polr, lmerMod, glmerMod, gam, or multinom.
...	Additional arguments passed to methods (currently unused).

Value

A named list with the following components, where available:

- `type` — A character string describing the model type
- `n` — The number of observations used in the model
- `r.squared` — R-squared (for OLS and GAM models)
- `adj.r.squared` — Adjusted R-squared (for OLS models)
- `mcfadden` — McFadden's pseudo-R^2 (for GLMs and `polr`, if `pscl` is installed)
- `aic` — AIC value for the model

The object is assigned class `"findn"` with a custom `print` method for display.

Author(s)

Josh Pasek

See Also

`summary`, `AIC`, `pR2`, `polr`, `multinom`, `lmer`, `gam`

Examples

```
data(mtcars)
mod1 <- lm(mpg ~ wt + hp, data = mtcars)
findn(mod1)

mod2 <- glm(am ~ wt + hp, data = mtcars, family = binomial)
findn(mod2)

library(MASS)
mod3 <- polr(Sat ~ Infl + Type + Cont, data = housing)
findn(mod3)

library(lme4)
mod4 <- lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
findn(mod4)

library(mgcv)
mod5 <- gam(mpg ~ s(wt) + hp, data = mtcars)
findn(mod5)

library(nnet)
mod6 <- multinom(vs ~ wt + hp, data = mtcars, trace = FALSE)
findn(mod6)
```

nalevs *Recode variables to 0-1 scale*

Description

nalevs takes as an input any vector and recodes it to range from 0 to 1, to treat specified levels as missing, to treat specified levels as 0, 1, .5, or the mean (weighted or unweighted) of the levels present after coding.

Usage

```
nalevs(x, naset=NULL, setmid=NULL, set1=NULL, set0=NULL,
setmean=NULL, weight=NULL)
```

Arguments

x	A vector to be recoded to range from 0 to 1.
naset	A vector of values of x to be coded as NA.
setmid	A vector of values of x to be recoded to .5.
set1	A vector of values of x to be recoded to 1.
set0	A vector of values of x to be recoded to 0.
setmean	A vector of values of x to be recoded to the mean (if no weight is specified) or weighted mean (if a weight is specified) of values of x after all recoding.
weight	A vector of weights for x if weighted means are desired for values listed for setmean.

Value

A vector of length equal to that of x of class numeric.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

Examples

```
data(anes04)
summary(anes04$age)
summary(nalevs(anes04$age))
table(anes04$educcats)
table(nalevs(anes04$educcats, naset=c(2, 4)))
```

onetable *Create a clean regression summary table from one or more models*

Description

onetable extracts and formats coefficients, standard errors, p-values, significance stars, and model fit statistics from one or more model objects. It returns a matrix-style table suitable for printing or export. Models may include linear models, generalized linear models, ordinal regressions, mixed-effects models, generalized additive models, penalized regressions, and multinomial regressions.

Usage

```
onetable(..., digits = 2, p.digits = 3,
         fitstats = c("r.squared", "adj.r.squared", "mcfadden", "aic", "n"),
         model.names = NULL, collapse = FALSE, formatted = TRUE,
         show.cutpoints = TRUE, approx.p = FALSE)
```

Arguments

...	One or more fitted model objects. Supported classes include lm, glm, polr, lmerMod, gam, glmnet, and multinom.
digits	Number of digits to display for estimates and standard errors (default is 2).
p.digits	Number of digits to display for p-values (default is 3).
fitstats	A character vector of model fit statistics to display. Options include "r.squared", "adj.r.squared", "mcfadden", "aic", and "n".
model.names	An optional character vector to name the models in the output.
collapse	If TRUE, each model is displayed in a single column with format estimate (se)***.
formatted	If TRUE, values are formatted using rd for readability. If FALSE, raw numeric values are returned.
show.cutpoints	If TRUE, includes cutpoints for ordinal models (e.g., from polr).
approx.p	If TRUE, attempts to compute approximate p-values (e.g., from t-statistics) for models that do not provide them natively (e.g., lmerMod).

Value

A character matrix with one row per coefficient (and optionally model fit statistics), and one or more columns depending on whether collapse = TRUE. The object is suitable for display using functions like kable or export to LaTeX or HTML tables.

See Also

coeffer, findn, rd, pR2, polr, multinom, lmer, gam, glmnet, kable

Examples

```
data(mtcars)
mod1 <- lm(mpg ~ wt + hp, data = mtcars)
mod2 <- glm(am ~ wt + hp, data = mtcars, family = binomial)

onetable(mod1, mod2)

# Collapsed form
onetable(mod1, mod2, collapse = TRUE)

# Use formatted = FALSE for raw numeric output
onetable(mod1, mod2, formatted = FALSE)

# Add approximate p-values for mixed models
library(lme4)
mod3 <- lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
onetable(mod3, approx.p = TRUE)
```

`plotwtdinteraction` *Functions to Identify and Plot Predicted Probabilities As Well As Two- and Three-Way Interactions From Regressions With or Without Weights and Standard Errors*

Description

`plotwtdinteraction` produces a plot from a regression object to illustrate a two- or three-way interaction for a prototypical individual holding constant all other variables (or other counterfactuals, depending on type). Prototypical individual is identified as the mean (numeric), median (ordinal), and/or modal (factors and logical variables) values for all measures. Standard errors are illustrated with polygons by default.

`findwtdinteraction` generates a table of point estimates from a regression object to illustrate a two- or three-way interaction for a prototypical individual holding constant all other variables. Prototypical individual is identified as the mean (numeric), median (ordinal), and/or modal (factors and logical variables) values for all measures. Standard errors are illustrated with polygons by default.

`plotinteractpreds` plots an object from `findwtdinteraction`.

These functions are known to be compatible with `lm`, `glm`, as well as multiply imputed `lm` and `glm` data generated with the `mice` package. They are also compatible with `gam` and `bam` regressions from the mgcv package under default.

ordinal regressions (polr) and multinomial regressions (multinom) do not currently support standard errors, additional methods are still being added.

Note, this set of functions is still in beta, please let me know if you run into any bugs when using it.

Important: If you are using a regression output from a multiply imputed dataset with a continuous variable as an interacting term, you should always specify the levels (acrosslevs, bylevs, or atlevs) for the variable, as imputations can change the set of levels that are available and thus can make the point estimates across imputed datasets incompatible with one-another.

Usage

```
plotwtdinteraction(x, across, by=NULL, at=NULL, acrosslevs=NULL, bylevs=NULL,
atlevs=NULL, weight=NULL, dvname=NULL, acclevnames=NULL, bylevnames=NULL,
atlevnames=NULL, stdzacross=FALSE, stdzby=FALSE, stdzat=FALSE, limitlevs=20,
type="response", seplot=TRUE, ylim=NULL, main=NULL, xlab=NULL, ylab=NULL,
legend=TRUE, placement="bottomright", lwd=3, add=FALSE, addby = TRUE, addat=FALSE,
mfrow=NULL, linecol=NULL, secol=NULL, showbynamelegend=FALSE,
showatnamelegend=FALSE, showoutnamelegend = FALSE,
lty=NULL, density=30, startangle=45, approach="prototypical", data=NULL,
nsim=100, xlim=NULL, ...)

findwtdinteraction(x, across, by=NULL, at=NULL, acrosslevs=NULL, bylevs=NULL,
atlevs=NULL, weight=NULL, dvname=NULL, acclevnames=NULL, bylevnames=NULL,
atlevnames=NULL, stdzacross=FALSE, stdzby=FALSE, stdzat=FALSE, limitlevs=20,
type="response", approach="prototypical", data=NULL, nsim=100)

plotinteractpreds(out, seplot=TRUE, ylim=NULL, main=NULL, xlab=NULL, ylab=NULL,
legend=TRUE, placement="bottomright", lwd=3, add=FALSE, addby = TRUE,
addat=FALSE, mfrow=NULL, linecol=NULL, secol=NULL, showbynamelegend=FALSE,
showatnamelegend=FALSE, showoutnamelegend = FALSE, lty=NULL,
density=30, startangle=45, xlim=NULL, ...)
```

Arguments

x	x is a regression object in lm, glm, or mira (multiply imputed) format that includes the variables to be plotted.
out	out is an object estimate using findwtdinteraction that should be plotted.
across	across specifies the name of the variable, in quotation marks, that was used in the regression that should be plotted on the X axis.
by	by specifies the name of the variable, in quotation marks, that was used in the regression that should form each of the separate lines in the regression.
at	at (optional) specifies the name of the variable, in quotation marks, that represents the third-way of a 3-way interaction. Depending on specifications, this can either be plotted as additional lines or as separate graphs.
acrosslevs	acrosslevs (optional) specifies the unique levels of the variable across that should be estimated across the x axis. If this is not specified, each unique level of the across variable will be used.
bylevs	bylevs (optional) specifies the unique levels of the variable by that should yield separate lines. If this is not specified, each unique level of the by variable will be used.
atlevs	atlevs (optional) specifies the unique levels of the variable at that should yield separate figures or lines. If this is not specified, each unique level of the at variable will be used.
weight	weight (optional) allows the user to introduce a separate weight that was not used in the original regression. If the regression was run using weights, those weights will always be used to generate estimates of the prototypical individual to be used.

dvname	dvname (optional) allows the user to relabel the dependent variable for printouts.
acclevnames	dvname (optional) allows the user to specify the names for the specified levels of the accross variable.
bylevnames	dvname (optional) allows the user to specify the names for the specified levels of the by variable.
atlevnames	dvname (optional) allows the user to specify the names for the specified levels of the at variable.
stdzacross	dvname (optional) shows levels of across variable in (weighted) standard deviation units. This defaults to showing 1SD below mean and 1SD above mean; specifying acrosslevs to other values will provide results in SD units instead of variable units.
stdzby	dvname (optional) shows levels of by variable in (weighted) standard deviation units. This defaults to showing 1SD below mean and 1SD above mean; specifying bylevs to other values will provide results in SD units instead of variable units.
stdzat	dvname (optional) shows levels of at variable in (weighted) standard deviation units. This defaults to showing 1SD below mean and 1SD above mean; specifying atlevs to other values will provide results in SD units instead of variable units.
limitlevs	limitlevs sets the number of different levels that any given interacting variable can have. This is meant to prevent inadvertent generation and plotting of tons of point estimates for continuous variables. The default is set to 20.
type	type sets the type of prediction to be used for generation of the estimates. This defaults to "response" but can be used with any type of model prediction for which only one numeric estimate is given. (Not currently compatible with estimates derived from polr regression).
seplot	seplot (optional) if set to TRUE, plots will include polygons illustrating standard errors.
ylim	ylim (optional) passes on y-axis limits to plot function.
main	main (optional) passes on title to plot function.
xlab	xlab (optional) passes on x-axis labels to plot function.
ylab	ylab (optional) passes on y-axis labels to plot function.
legend	legend (optional) if TRUE will produce a legend on the interaction figure.
placement	placement (optional) passes to legend function a location for the legend. Can be set to "bottomright", "bottomleft", "topright", and "topleft".
lwd	lwd (optional) specifies the line strength for plots, this passes on to the plot command.
add	add (optional) logical statement to add the results to an existing plot (at=TRUE) rather than generating a new one (at=FALSE is the default).
addby	addby (optional) logical statement specifying whether the levels of by should be different plots (addby=TRUE) or if each level of by should generate a new plot (addby=TRUE is the default). This only influences some types of plots.

addat addat (optional) logical statement specifying whether the levels of at should be different plots (addat=TRUE) or if each level of at should generate a new plot (addat=FALSE is the default)

mfrow mfrow (optional) temporarily changes the number of plots per page in par for the purpose of generating current plots. This should generally only be used for 3-way interactions. It takes commands of the form c(2,3), specifying the number of rows and columns in the graphics interface. The algorithm defaults to putting all 3-way interactions on a single page with a width of 2.

linecol linecol (optional) Specifies the colors of lines in the figure(s). For two-way interactions, this should be a vector of the same length as bylevs. For 3-way interactions, the colors demarcate the levels of at instead and should be the same length as atlevs.

secol secol (optional) Specifies the colors of standard error in the figure(s). For two-way interactions, this should be a vector of the same length as bylevs. For 3-way interactions, the colors demarcate the levels of at instead and should be the same length as atlevs.

showbynamelegend
showbynamelegend (optional) adds name of by variable to names of value levels in legend.

showatnamelegend
showatnamelegend (optional) adds name of at variable to names of value levels in legend.

showoutnamelegend
showoutnamelegend (optional) adds name of DV to legend in multinomial logit plots only.

lty lty (optional) line type to pass on to plot.

density density (optional) line density for standard error plots.

startangle startangle (optional) line angle for standard error plots.

approach approach determines whether you want to estimate counterfactuals for a prototypical individual approach="prototypical" (the default), for the entire population approach="population", or for individuals in the subgroups specified in the by and at categories approach="at", approach="by", approach="atby".

data data (optional) allows you to replace the dataset used in the regression to produce other prototypical values.

nsim nsim (optional) set the number of bootstrapped simulations to use to generate standard errors for lmer-style regressions. Note that this SIGNIFICANLY increases the time to run, so test with smaller numbers before running.

xlim xlim (optional) set the range of x axis, passes to plot.

... ... (optional) Additional arguments to be passed on to plot command (or future methods of findwtdinteraction).

Value

A table or figure illustrating the predicted values of the dependent variable across levels of the independent variables for a prototypical respondent.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

`rd` *Round Numbers To Text With No Leading Zero*

Description

Rounds numbers to text and drops leading zeros in the process.

Usage

```
rd(x, digits=2, add=TRUE, max=(digits+3))
```

Arguments

`x`	A vector of values to be rounded (must be numeric).
`digits`	The number of digits to round to (must be an integer).
`add`	An optional dichotomous indicator for whether additional digits should be added if no numbers appear in pre-set digit level.
`max`	Maximum number of digits to be shown if `add=TRUE`.

Value

A vector of length equal to that of `x` of class `character`.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

Examples

```
rd(seq(0, 1, by=.1))
```

starmaker *Produce stars from p values for tables.*

Description

Recodes p values to stars for use in tables.

Usage

```
starmaker(x, p.levels=c(.001, .01, .05, .1), symbols=c("***", "**", "*", "+"))
```

Arguments

x	A vector of p values to be turned into stars (must be numeric).
p.levels	A vector of the maximum p value for each symbol used (p<p.level).
symbols	A vector of the symbols to be displayed for each p value.

Value

A vector of length equal to that of x of class character.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

Examples

```
starmaker(seq(0, .15, by=.01))
cbind(p=seq(0, .15, by=.01), star=starmaker(seq(0, .15, by=.01)))
```

stdz *Standardizes any numerical vector, with weights.*

Description

stdz produces a standardized copy of any input variable. It can also standardize a weighted variable to produce a copy of the original variable standardized around its weighted mean and variance.

Usage

```
stdz(x, weight=NULL)
```

Arguments

x	x should be a numerical vector which the researcher wishes to standardize.
weight	weight is an optional vector of weights to be used to determining the weighted mean and variance for standardization.

Value

A vector of length equal to x with a (weighted) mean of zero and a (weighted) standard deviation of 1.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

See Also

wtd.cor wtd.chi.sq wtd.t.test

Examples

```
test <- c(1,1,1,1,1,1,2,2,2,3,3,3,4,4)
weight <- c(.5,.5,.5,.5,.5,1,1,1,1,2,2,2,2,2)

summary(stdz(test))
summary(stdz(test, weight))
Hmisc::wtd.mean(stdz(test, weight), weight)
Hmisc::wtd.var(stdz(test, weight), weight)
```

---

wpct *Provides a weighted table of percentages for any variable.*

Description

wpct produces a weighted table of the proportion of data in each category for any variable. This is simply a weighted frequency table divided by its sum.

Usage

```
wpct(x, weight=NULL, na.rm=TRUE, ...)
```

Arguments

x	x should be a vector for which a set of proportions is desired.
weight	weight is a vector of weights to be used to determining the weighted proportion in each category of x.
na.rm	If na.rm is true, missing data will be dropped. If na.rm is false, missing data will return an error.
...	... (optional) Additional arguments to be passed on to wtd.table.

Value

A table object of length equal to the number of separate values of x.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

Examples

```
test <- c(1,1,1,1,1,1,2,2,2,3,3,3,4,4)
weight <- c(.5,.5,.5,.5,.5,1,1,1,1,2,2,2,2,2)

wpct(test)
wpct(test, weight)
```

---

wtd.anova *Weighted one-way ANOVA*

Description

wtd.anova performs a weighted analysis of variance across groups using a continuous response variable and a grouping factor.

Usage

```
wtd.anova(response, group, weight = NULL)
```

Arguments

response	Numeric vector of outcome values.
group	Factor indicating group membership.
weight	Optional numeric vector of weights. If NULL, equal weights are used.

Value

A data frame with rows for "Between" and "Within" group variance and columns for SS, df, MS, F statistic, and p-value.

Author(s)

Josh Pasek

See Also

aov, lm

Examples

```
set.seed(1)
group <- rep(c("A", "B", "C"), each = 10)
x <- c(rnorm(10), rnorm(10, mean = 1), rnorm(10, mean = 2))
w <- runif(30, 0.5, 2)
wtd.anova(x, group, weight = w)
```

---

wtd.barplot *Weighted barplot*

Description

wtd.barplot is a wrapper around barplot that creates barplots of counts or proportions using weights. Note that for now this only works in the special case of a single weighted variable. Formulas will be added later.

Usage

```
wtd.barplot(x, weight = NULL, percent = FALSE, horiz = FALSE, ...)
```

Arguments

x	Categorical variable (factor or character).
weight	Optional numeric vector of weights.
percent	If TRUE, display percentages instead of raw counts.
horiz	If TRUE, draw bars horizontally.
...	Additional arguments passed to barplot.

Value

A barplot is drawn. No value is returned.

Author(s)

Josh Pasek

See Also

barplot, table

Examples

```
x <- sample(c("Yes", "No"), 100, replace = TRUE)
w <- runif(100, 0.5, 2)
wtd.barplot(x, weight = w)
```

`wtd.boxplot` *Weighted boxplot*

Description

`wtd.boxplot` produces boxplots for weighted data by group, accounting for weights when computing medians and quartiles.

Usage

```
wtd.boxplot(x, group = NULL, weight = NULL, show.outliers = TRUE,
            whisker.mult = 1.5, box.col = "lightgray", border = "black", ...)
```

Arguments

`x`	Numeric vector of values.
`group`	Optional grouping factor.
`weight`	Optional numeric vector of weights. If `NULL`, equal weights are assumed.
`show.outliers`	Logical. If `TRUE`, show weighted outliers based on interquartile range.
`whisker.mult`	Numeric multiplier for the IQR to define whiskers (default is 1.5, as in standard boxplots).
`box.col`	Color for the box portion of the plot.
`border`	Color for the boxplot borders.
`...`	Additional graphical parameters passed to `plot` or `rect`.

Value

A base R graphic is produced showing weighted boxplots by group. No value is returned.

Author(s)

Josh Pasek

See Also

`boxplot`, `wtd.quantile`, `wtd.median`

Examples

```
set.seed(123)
x <- rnorm(100)
group <- rep(letters[1:2], each = 50)
w <- runif(100, 0.5, 2)
wtd.boxplot(x, group, weight = w)
```

wtd.chi.sq *Produces weighted chi-squared tests.*

Description

wtd.chi.sq produces weighted chi-squared tests for two- and three-variable contingency tables. Decomposes parts of three-variable contingency tables as well. Note that weights run with the default parameters here treat the weights as an estimate of the precision of the information. A prior version of this software was set to default to mean1=FALSE.

Usage

```
wtd.chi.sq(var1, var2, var3=NULL, weight=NULL, na.rm=TRUE,
drop.missing.levels=TRUE, mean1=TRUE)
```

Arguments

var1	var1 is a vector of values which the researcher would like to use to divide any data set.
var2	var2 is a vector of values which the researcher would like to use to divide any data set.
var3	var3 is an optional additional vector of values which the researcher would like to use to divide any data set.
weight	weight is an optional vector of weights to be used to determine the weighted chi-squared for all analyses.
na.rm	na.rm removes missing data from analyses.
drop.missing.levels	drop.missing.levels drops missing levels from variables.
mean1	mean1 is an optional parameter for determining whether the weights should be forced to have an average value of 1. If this is set as false, the weighted correlations will be produced with the assumption that the true N of the data is equivalent to the sum of the weights.

Value

A two-way chi-squared produces a vector including a single chi-squared value, degrees of freedom measure, and p-value for each analysis.

A three-way chi-squared produces a matrix with a single chi-squared value, degrees of freedom measure, and p-value for each of seven analyses. These include: (1) the values using a three-way contingency table, (2) the values for a two-way contingency table with each pair of variables, and (3) assessments for whether the relations between each pair of variables are significantly different across levels of the third variable.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

See Also

wtd.cor wtd.t.test

Examples

```
var1 <- c(1,1,1,1,1,2,2,2,2,2,3,3,3,3,3)
var2 <- c(1,1,2,2,3,3,1,1,2,2,3,3,1,1,2)
var3 <- c(1,2,3,1,2,3,1,2,3,1,2,3,1,2,3)
weight <- c(.5,.5,.5,.5,.5,1,1,1,1,1,2,2,2,2,2)

wtd.chi.sq(var1, var2)
wtd.chi.sq(var1, var2, weight=weight)

wtd.chi.sq(var1, var2, var3)
wtd.chi.sq(var1, var2, var3, weight=weight)
```

---

wtd.cor *Produces weighted correlations with standard errors and significance. For a faster version without standard errors and p values, use the* wtd.cors *function.*

Description

wtd.cor produces a Pearsons correlation coefficient comparing two variables or matrices. Note that weights run with the default parameters here treat the weights as an estimate of the precision of the information. For survey data, users should run this code with bootstrapped standard errors bootse=TRUE, which are robust to heteroskadesticity, although these will vary slightly each time the weights are run. A prior version of this software was set to default to mean1=FALSE and bootse=FALSE.

Usage

```
wtd.cor(x, y=NULL, weight=NULL, mean1=TRUE, collapse=TRUE, bootse=FALSE,
bootp=FALSE, bootn=1000)
```

Arguments

x	x should be a matrix or vector which the researcher wishes to correlate with y.
y	y should be a numerical vector or matrix which the researcher wishes to correlate with x. If y is NULL, x will be used instead
weight	weight is an optional vector of weights to be used to determining the weighted mean and variance for calculation of the correlations.

mean1	mean1 is an optional parameter for determining whether the weights should be forced to have an average value of 1. If this is set as false, the weighted correlations will be produced with the assumption that the true N of the data is equivalent to the sum of the weights.
collapse	collapse is an indicator for whether the data should be collapsed to a simpler form if either x or y is a vector instead of a matrix.
bootse	bootse is an optional parameter that produces bootstrapped standard errors. This should be used to address heteroskadesticity issues when weights indicate probabilities of selection rather than the precision of estimates.
bootp	bootp is an optional parameter that produces bootstrapped p values instead of estimating p values from the standard errors. This parameter only operates when bootse=TRUE.
bootn	bootn is an optional parameter that is used to indicate the number of bootstraps that should be run for bootse and bootp.

Value

A list with matrices for the estimated correlation coefficient, the standard error on that correlation coefficient, the t-value for that correlation coefficient, and the p value for the significance of the correlation. If the list can be simplified, simplification will be done.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

See Also

wtd.cors stdz wtd.t.test wtd.chi.sq

Examples

```
test <- c(1,1,1,1,1,1,2,2,2,3,3,3,4,4)
t2 <- rev(test)
weight <- c(.5,.5,.5,.5,.5,1,1,1,1,2,2,2,2,2)

wtd.cor(test, t2)
wtd.cor(test, t2, weight)
wtd.cor(test, t2, weight, bootse=TRUE)
```

---

wtd.cors *Produces weighted correlations quickly using C.*

Description

wtd.cors produces a Pearsons correlation coefficient comparing two variables or matrices.

Usage

```
wtd.cors(x, y=NULL, weight=NULL)
```

Arguments

x	x should be a matrix or vector which the researcher wishes to correlate with y.
y	y should be a numerical vector or matrix which the researcher wishes to correlate with x. If y is NULL, x will be used instead
weight	weight is an optional vector of weights to be used to determining the weighted mean and variance for calculation of the correlations.

Value

A matrix of the estimated correlation coefficients.

Author(s)

Marcus Schwemmle at GfK programmed the C code, R wrapper by Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com).

See Also

wtd.cor stdz wtd.t.test wtd.chi.sq

Examples

```
test <- c(1,1,1,1,1,1,2,2,2,3,3,3,4,4)
t2 <- rev(test)
weight <- c(.5,.5,.5,.5,1,1,1,1,2,2,2,2,2)

wtd.cors(test, t2)
wtd.cors(test, t2, weight)
```

---

wtd.cov *Produces weighted covariances with standard errors and significance.*

Description

wtd.cov produces a covariance matrix comparing two variables or matrices, using a set of weights. Standard errors, t-values, and p-values are estimated via a regression-based approach. If no weights are provided, unweighted covariance is returned.

Usage

```
wtd.cov(x, y=NULL, weight=NULL, collapse=TRUE)
```

Arguments

x	A matrix or vector of values to be compared. If y is NULL, x will be used for both variables.
y	A vector or matrix to be compared with x. Defaults to NULL.
weight	Optional weights used to compute the weighted covariance. If NULL, equal weighting is assumed.
collapse	Logical indicator for whether the results should be simplified when the output is a vector.

Value

A list containing:

- covariance — Weighted covariance matrix
- std.err — Standard error of the covariance estimate
- t.value — T-statistic associated with the covariance
- p.value — P-value for the t-statistic

If the results are scalar or one-dimensional, a simplified matrix will be returned.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (https://www.joshpasek.com)

See Also

wtd.cor, wtd.partial.cov, onecor.wtd, wtd.var, stdz

Examples

```
x <- c(1, 2, 3, 4)
y <- c(2, 4, 6, 8)
w <- c(1, 2, 1, 1)

wtd.cov(x, y)
wtd.cov(x, y, weight = w)
```

wtd.hist *Weighted Histograms*

Description

Produces weighted histograms by adding a "weight" option to the his.default function from the graphics package (Copyright R-core). The code here was copied from that function and modified slightly to allow for weighted histograms as well as unweighted histograms. The generic function hist computes a histogram of the given data values. If plot=TRUE, the resulting object of class "histogram" is plotted by plot.histogram, before it is returned.

Usage

```
wtd.hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, weight = NULL, ...)
```

Arguments

x	a vector of values for which the histogram is desired.
breaks	one of: • a vector giving the breakpoints between histogram cells, • a single number giving the number of cells for the histogram, • a character string naming an algorithm to compute the number of cells (see 'Details'), • a function to compute the number of cells. In the last three cases the number is a suggestion only.
freq	logical; if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE *if and only if* breaks are equidistant (and probability is not specified).
probability	an *alias* for !freq, for S compatibility.
include.lowest	logical; if TRUE, an x[i] equal to the breaks value will be included in the first (or last, for right = FALSE) bar. This will be ignored (with a warning) unless breaks is a vector.
right	logical; if TRUE, the histogram cells are right-closed (left open) intervals.

density	the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.
angle	the slope of shading lines, given as an angle in degrees (counter-clockwise).
col	a colour to be used to fill the bars. The default of NULL yields unfilled bars.
border	the color of the border around the bars. The default is to use the standard foreground color.
main, xlab, ylab	these arguments to title have useful defaults here.
xlim, ylim	the range of x and y values with sensible defaults. Note that xlim is *not* used to define the histogram (breaks), but only for plotting (when plot = TRUE).
axes	logical. If TRUE (default), axes are draw if the plot is drawn.
plot	logical. If TRUE (default), a histogram is plotted, otherwise a list of breaks and counts is returned. In the latter case, a warning is used if (typically graphical) arguments are specified that only apply to the plot = TRUE case.
labels	logical or character. Additionally draw labels on top of bars, if not FALSE; see plot.histogram in the graphics package.
nclass	numeric (integer). For S(-PLUS) compatibility only, nclass is equivalent to breaks for a scalar or character argument.
weight	numeric. Defines a set of weights to produce a weighted histogram. Will default to 1 for each case if no other weight is defined.
...	further arguments and graphical parameters passed to plot.histogram and thence to title and axis (if plot=TRUE).

Details

The definition of *histogram* differs by source (with country-specific biases). R's default with equi-spaced breaks (also the default) is to plot the (weighted) counts in the cells defined by breaks. Thus the height of a rectangle is proportional to the (weighted) number of points falling into the cell, as is the area *provided* the breaks are equally-spaced.

The default with non-equi-spaced breaks is to give a plot of area one, in which the *area* of the rectangles is the fraction of the data points falling in the cells.

If right = TRUE (default), the histogram cells are intervals of the form (a, b], i.e., they include their right-hand endpoint, but not their left one, with the exception of the first cell when include.lowest is TRUE.

For right = FALSE, the intervals are of the form [a, b), and include.lowest means '*include highest*'.

The default for breaks is "Sturges": see nclass.Sturges. Other names for which algorithms are supplied are "Scott" and "FD" / "Freedman-Diaconis" (with corresponding functions nclass.scott and nclass.FD). Case is ignored and partial matching is used. Alternatively, a function can be supplied which will compute the intended number of breaks as a function of x.

Value

an object of class "histogram" which is a list with components:

breaks	the $n+1$ cell boundaries (= breaks if that was a vector). These are the nominal breaks, not with the boundary fuzz.
counts	n values; for each cell, the number of x[] inside.
density	values for each bin such that the area under the histogram totals 1. $\hat{f}(x_i\omega_i)$ / $f^(x[i]\omega[i])$, as estimated density values. If all(diff(breaks) == 1), they are the relative frequencies counts/n and in general satisfy $\sum_i \hat{f}(x_i\omega_i)(b_{i+1} - b_i) = 1$ / $sum[i; f^(x[i]\omega[i])(b[i+1] - b[i])] = 1$, where b_i = breaks[i].
intensities	same as density. Deprecated, but retained for compatibility.
mids	the n cell midpoints.
xname	a character string with the actual x argument name.
equidist	logical, indicating if the distances between breaks are all the same.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com) was responsible for the updates to the hist function necessary to implement weighted counts. The hist.default code from the graphics package on which the current function was based was written by R-core (in 2010). All modifications are noted in code and the copyright for all original code remains with R-core.

Examples

```
var1 <- c(1:100)
wgt <- var1/mean(var1)
par(mfrow=c(2, 2))
wtd.hist(var1)
wtd.hist(var1, weight=wgt)
wtd.hist(var1, weight=var1)
```

---

wtd.median *Weighted median*

Description

wtd.median computes the median of a numeric vector using weights.

Usage

```
wtd.median(x, weight = NULL, na.rm = TRUE)
```

Arguments

x	Numeric vector of values.
weight	Optional numeric vector of weights.
na.rm	Logical. If TRUE, removes missing values.

Value

A single numeric value representing the weighted median.

Author(s)

Josh Pasek

See Also

wtd.quantile, median

Examples

```
x <- c(1, 2, 3, 4, 5)
w <- c(1, 1, 5, 1, 1)
wtd.median(x, weight = w)
```

---

wtd.partial.cor *Computes weighted partial correlations, controlling for covariates*

Description

wtd.partial.cor estimates the weighted partial correlation between two variables or sets of variables, controlling for additional covariates. This function uses weighted regression to residualize the inputs and computes the correlation of the residuals, providing standard errors and significance tests.

Usage

```
wtd.partial.cor(x, y = NULL, preds = NULL, weight = NULL, collapse = TRUE)
```

Arguments

x	A numeric vector or matrix. Each column will be residualized on preds.
y	An optional numeric vector or matrix. If NULL, x is used as both inputs.
preds	Covariates to control for via weighted linear regression.
weight	Optional weights to be applied in the regression and correlation steps.
collapse	Logical. If TRUE, simplifies the output to a matrix when possible.

Value

A list with:

- `correlation` — Estimated partial correlations
- `std.err` — Standard errors
- `t.value` — T-statistics
- `p.value` — P-values

When `collapse = TRUE`, the result is simplified when possible.

Author(s)

Josh Pasek (https://www.joshpasek.com)

See Also

`wtd.partial.cov`, `wtd.cor`, `onecor.wtd`

Examples

```
set.seed(456)
x <- rnorm(100)
y <- 0.4 * x + rnorm(100)
z <- rnorm(100)
w <- runif(100, 1, 2)

wtd.partial.cor(x, y, preds = z, weight = w)
```

`wtd.partial.cov` *Computes weighted partial covariances, controlling for covariates*

Description

`wtd.partial.cov` estimates the weighted partial covariance between two variables or sets of variables, controlling for additional covariates. The function uses weighted linear regression to residualize both dependent and independent variables before computing weighted covariances among the residuals.

Usage

```
wtd.partial.cov(x, y = NULL, preds = NULL, weight = NULL, collapse = TRUE)
```

Arguments

x	A numeric vector or matrix. Each column will be residualized on preds and used in the partial covariance calculation.
y	An optional numeric vector or matrix. If NULL, x will be used in both dimensions.
preds	A vector, matrix, or data frame of covariates to control for via linear regression.
weight	An optional numeric vector of weights. If NULL, equal weights are assumed.
collapse	Logical. If TRUE, the output will be simplified to a matrix if possible.

Value

A list with the following components:

- covariance — Weighted partial covariance estimates
- std.err — Standard errors of the covariance estimates
- t.value — T-statistics
- p.value — P-values

If the covariance matrix is a vector or scalar, the result is simplified when collapse = TRUE.

Author(s)

Josh Pasek (https://www.joshpasek.com)

See Also

wtd.partial.cor, wtd.cov

Examples

```
set.seed(123)
x <- rnorm(100)
y <- 0.5 * x + rnorm(100)
z <- rnorm(100)
w <- runif(100, 0.5, 1.5)

wtd.partial.cov(x, y, preds = z, weight = w)
```

`wtd.t.test` *Produces weighted Student's t-tests with standard errors and significance.*

Description

`wtd.t.test` produces either one- or two-sample t-tests comparing weighted data streams to one another. Note that weights run with the default parameters here treat the weights as an estimate of the precision of the information. For survey data, users should run this code with bootstrapped standard errors `bootse=TRUE`, which are robust to heteroskadesticity, although these will vary slightly each time the weights are run. A prior version of this software was set to default to `mean1=FALSE` and `bootse=FALSE`.

Usage

```
wtd.t.test(x, y=0, weight=NULL, weighty=NULL, samedata=TRUE,
alternative="two.tailed", mean1=TRUE, bootse=FALSE, bootp=FALSE,
bootn=1000, drops="pairwise")
```

Arguments

`x`	`x` is a numerical vector which the researcher wishes to test against `y`.
`y`	`y` can be either a single number representing an alternative hypothesis or a second numerical vector which the researcher wishes to compare against `x`.
`weight`	`weight` is an optional vector of weights to be used to determine the weighted mean and variance for the `x` vector for all t-tests. If `weighty` is unspecified and `samedata` is TRUE, this weight will be assumed to apply to both `x` and `y`.
`weighty`	`weighty` is an optional vector of weights to be used to determine the weighted mean and variance for the `y` vector for two-sample t-tests. If `weighty` is unspecified and `samedata` is TRUE, this weight will be assumed to equal `weightx`. If `weighty` is unspecified and `samedata` is FALSE, this weight will be assumed to equal 1 for all cases.
`samedata`	`samedata` is an optional identifier for whether the `x` and `y` data come from the same data stream for a two-sample test. If true, `wtd.t.test` assumes that `weighty` should equal `weightx` if (1) `weighty` is unspecified, and (2) the lengths of the two vectors are identical.
`alternative`	`alternative` is an optional marker for whether one or two-tailed p-values shoould be returned. By default, two-tailed values will be returned (`type="two.tailed"`). To set to one-tailed values, alternative can be set to `type="greater"` to test `x>y` or `type="less"` to test `x<y`.
`mean1`	`mean1` is an optional parameter for determining whether the weights should be forced to have an average value of 1. If this is set as false, the weighted correlations will be produced with the assumption that the true N of the data is equivalent to the sum of the weights.

bootse	bootse is an optional parameter that produces bootstrapped standard errors. This should be used to address heteroskadesticity issues when weights indicate probabilities of selection rather than the precision of estimates.
bootp	bootp is an optional parameter that produces bootstrapped p values instead of estimating p values from the standard errors. This parameter only operates when bootse=TRUE.
bootn	bootn is an optional parameter that is used to indicate the number of bootstraps that should be run for bootse and bootp.
drops	drops is set to limit a t-test on the same data to cases with nonmissing data for x, y, and weights (if specified). If drops is anything other than "pairwise", means for x and y are calculated on all available data rather than data that are available for both x and y. This parameter does nothing if x and y are not from the same dataset.

Value

A list element with an identifier for the test; coefficients for the t value, degrees of freedom, and p value of the t-test; and additional statistics of potential interest.

Author(s)

Josh Pasek, Professor of Communication & Media and Political Science at the University of Michigan (www.joshpasek.com). Gene Culter added code for a one-tailed version of the test.

See Also

stdz wtd.cor wtd.chi.sq

Examples

```
test <- c(1,1,1,1,1,1,2,2,2,3,3,3,4,4)
t2 <- rev(test)+1
weight <- c(.5,.5,.5,.5,.5,1,1,1,1,2,2,2,2,2)

wtd.t.test(test, t2)
wtd.t.test(test, t2, weight)
wtd.t.test(test, t2, weight, bootse=TRUE)
```

---

wtd.violinplot *Draw weighted violin plots by group*

Description

wtd.violinplot produces violin plots for weighted data by group using kernel density estimation.

Usage

```
wtd.violinplot(x, group = NULL, weight = NULL,
               bw = "nrd0", adjust = 1,
               col = "gray", border = "black",
               names = NULL, width = 0.4,
               na.rm = TRUE, ...)
```

Arguments

x	Numeric vector of values.
group	Optional grouping factor indicating which group each value belongs to. If NULL, all values are treated as a single group.
weight	Optional numeric vector of weights, the same length as x. If NULL, equal weights are used.
bw	The smoothing bandwidth to be used, passed to density. Default is "nrd0".
adjust	A multiplicative bandwidth adjustment. The bandwidth used is actually adjust * bw. See density.
col	Color(s) for the filled violin shapes. Passed to vioplot.
border	Color(s) for the outline of the violins.
names	Optional vector of group names to be displayed on the x-axis. If NULL, levels of the group factor are used.
width	Width of the violin plots. Passed to vioplot.
na.rm	Logical. Should missing values be removed? Default is TRUE.
...	Additional graphical parameters passed to vioplot.

Details

This function uses kernel density estimates with weights to generate violin plots for each level of the grouping variable. Internally, it calls density with the weights argument, and constructs violin plots using vioplot.

Value

A base R plot is produced showing weighted violin plots by group. No value is returned.

Author(s)

Josh Pasek

See Also

vioplot, density, wtd.hist, wtd.boxplot

Examples

```
set.seed(123)
x <- c(rnorm(100), rnorm(100, mean = 2))
group <- rep(c("A", "B"), each = 100)
wts <- c(rep(1, 100), runif(100, 0.5, 2))

wtd.violinplot(x, group = group, weight = wts,
               col = c("lightblue", "lightgreen"))

x2 <- c(seq(0,2,length.out=100), seq(0,6,length.out=100))
wts2 <- rep(1, 200)

wtd.violinplot(x2, group = group, weight = wts2,
               col = c("lightblue", "lightgreen"))

wtd.violinplot(x2, group = group, weight = (wts2+.1)/(x2+.1),
               col = c("lightblue", "lightgreen"))
```

`wtd.xtab` *Weighted cross-tabulations using up to three categorical variables*

Description

`wtd.xtab` creates 2-way or 3-way weighted cross-tabulations. It uses weighted counts and optionally returns row, column, or total percentages. The function outputs either a matrix or a list of matrices for easier interpretation than base R's default array structure.

Usage

```
wtd.xtab(var1, var2, var3 = NULL,
         weight = NULL,
         percent = c("none", "row", "column", "total"),
         na.rm = TRUE,
         drop.missing.levels = TRUE,
         mean1 = TRUE,
         digits = 1)
```

Arguments

`var1`	A categorical variable to appear as rows in the table.
`var2`	A categorical variable to appear as columns in the table.
`var3`	An optional third categorical variable used to split the table (i.e., one table per level of `var3`).
`weight`	A numeric vector of weights. If `NULL`, equal weights are assumed.
`percent`	How percentages should be computed: `"none"` (default), `"row"`, `"column"`, or `"total"`.

na.rm	Logical. If TRUE, removes observations with missing values on any input variable.
drop.missing.levels	Logical. If TRUE, drops unused factor levels in var1, var2, and var3.
mean1	Logical. If TRUE, normalizes the weights to have a mean of 1.
digits	Number of digits to which percentages should be rounded (only used if percent != "none").

Details

This function provides a cleaner and more interpretable alternative to xtabs when working with weights and categorical variables. It simplifies 2-way and 3-way tabulations and avoids confusing multi-dimensional array output.

Value

If var3 is NULL, returns a list with:

- counts — A matrix of weighted counts
- percent — A matrix of percentages (or NULL if percent = "none")

If var3 is specified, returns a named list where each element corresponds to a level of var3 and contains:

- counts — A matrix of weighted counts
- percent — A matrix of percentages (or NULL)

Author(s)

Josh Pasek

See Also

wtd.chi.sq, xtabs, wtd.table

Examples

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
mtcars$am <- factor(mtcars$am)
mtcars$gear <- factor(mtcars$gear)
mtcars$wt_cat <- cut(mtcars$wt, 3)

# Two-way table
wtd.xtab(mtcars$cyl, mtcars$am)

# With row percentages
wtd.xtab(mtcars$cyl, mtcars$am, weight = mtcars$wt, percent = "row")

# Three-way table, split by gear
wtd.xtab(mtcars$cyl, mtcars$am, mtcars$gear, weight = mtcars$wt)
```

```
# Column percentages by weight class
wtd.xtab(mtcars$cyl, mtcars$am, mtcars$wt_cat, weight = mtcars$wt, percent = "column")
```

Index