Package 'vtable'

March 15, 2023

Type Package

Title Variable Table for Variable Documentation

Version 1.4.2

Description Automatically generates HTML variable documentation including variable names, labels, classes, value labels (if applicable), value ranges, and summary statistics. See the vignette ``vtable" for a package overview.

License MIT + file LICENSE

Encoding UTF-8

Depends kableExtra

Imports utils, stats, rstudioapi, sjlabelled, haven, knitr

Suggests rmarkdown, survey

RoxygenNote 7.2.3

VignetteBuilder knitr

URL https://nickch-k.github.io/vtable/

BugReports https://github.com/NickCH-K/vtable/issues

NeedsCompilation no

Author Nick Huntington-Klein [aut, cre]

Maintainer Nick Huntington-Klein <nhuntington-klein@seattleu.edu>

Repository CRAN

Date/Publication 2023-03-14 23:00:02 UTC

R topics documented:

`countNA` *Number of missing values in a vector*

Description

This function calculates the number of values in a vector that are NA.

Usage

```
countNA(x)
```

Arguments

`x`	A vector.

Details

This function just shorthand for `sum(is.na(x))`, with a shorter name for reference in the `vtable` or `sumtable` `summ` option.

Examples

```
x <- c(1, 1, NA, 2, 3, NA)
countNA(x)
```

`dftoHTML` *Data Frame to HTML Function*

Description

This function takes a data frame or matrix with column names and outputs an HTML table version of that data frame.

Usage

```
dftoHTML(
  data,
  out = NA,
  file = NA,
  note = NA,
  note.align = "l",
  anchor = NA,
  col.width = NA,
  col.align = NA,
  row.names = FALSE,
  no.escape = NA
)
```

Arguments

data	Data set; accepts any format with column names.
out	Determines where the completed table is sent. Set to "browser" to open HTML file in browser using browseURL(), "viewer" to open in RStudio viewer using viewer(), if available, or "htmlreturn" to return the HTML code. Defaults to Defaults to "viewer" if RStudio is running and "browser" if it isn't.
file	Saves the completed variable table file to HTML with this filepath. May be combined with any value of out.
note	Table note to go after the last row of the table.
note.align	Alignment of table note, l, r, or c.
anchor	Character variable to be used to set an <a name> tag for the table.
col.width	Vector of page-width percentages, on 0-100 scale, overriding default column widths in HTML table. Must have a number of elements equal to the number of columns in the resulting table.
col.align	Vector of 'left', 'right', 'center', etc. to be used with the HTML table text-align attribute in each column. If you want to get tricky, you can add a ";" afterwards and keep putting in whatever CSS attributes you want. They will be applied to the whole column.
row.names	Flag determining whether or not the row names should be included in the table. Defaults to FALSE.
no.escape	Vector of column indices for which special characters should not be escaped (perhaps they include markup text of their own).

Details

This function is designed to feed HTML versions of variable tables to vtable(), sumtable(), and labeltable().

Multi-column cells are supported. Set the cell's contents to "content_MULTICOL_c_5" where "content" is the content of the cell, "c" is the cell's alignment (l, c, r), and 5 is the number of columns to span. Then fill in the cells that need to be deleted to make room with "DELETECELL".

If the first column and row begins with the text "HEADERROW", then the first row will be put above the column names.

Examples

```
if(interactive()) {
df <- data.frame(var1 = 1:4,var2=5:8,var3=c('A','B','C','D'),
    var4=as.factor(c('A','B','C','C')),var5=c(TRUE,TRUE,FALSE,FALSE))
dftoHTML(df,out="browser")
}
```

`dftoLaTeX` *Data Frame to LaTeX Function*

Description

This function takes a data frame or matrix with column names and outputs a lightly-formatted LaTeX table version of that data frame.

Usage

```
dftoLaTeX(
  data,
  file = NA,
  fit.page = NA,
  frag = TRUE,
  title = NA,
  note = NA,
  note.align = "l",
  anchor = NA,
  align = NA,
  row.names = FALSE,
  no.escape = NA
)
```

Arguments

`data`	Data set; accepts any format with column names.
`file`	Saves the completed table to LaTeX with this filepath.
`fit.page`	uses a LaTeX resizebox to force the table to a certain width. Often `'\textwidth'`.
`frag`	Set to TRUE to produce only the LaTeX table itself, or FALSE to produce a fully buildable LaTeX. Defaults to TRUE.
`title`	Character variable with the title of the table.
`note`	Table note to go after the last row of the table.

note.align	Set the alignment for the multi-column table note. Usually "l", but if you have a long note you might want to set it with "p"
anchor	Character variable to be used to set a label tag for the table.
align	Character variable with standard LaTeX formatting for alignment, for example 'lccc'. You can also use this to force column widths with p in standard LaTeX style. Defaults to the first column being left-aligned and all others centered. Be sure to escape special characters, in particular backslashes (i.e. p{.25\\textwidth} instead of p{.25\textwidth}).
row.names	Flag determining whether or not the row names should be included in the table. Defaults to FALSE.
no.escape	Vector of column indices for which special characters should not be escaped (perhaps they include markup text of their own).

Details

This function is designed to feed LaTeX versions of variable tables to vtable(), sumtable(), and labeltable().

Multi-column cells are supported. Wrap the cell's contents in a multicolumn tag as normal, and then fill in any cells that need to be deleted to make room for the multi-column cell with "DELETECELL". Or use the MULTICOL syntax of dftoHTML, that works too.

If the first column and row begins with the text "HEADERROW", then the first row will be put above the column names.

Examples

```
df <- data.frame(var1 = 1:4,var2=5:8,var3=c('A','B','C','D'),
    var4=as.factor(c('A','B','C','C')),var5=c(TRUE,TRUE,FALSE,FALSE))
dftoLaTeX(df, align = 'ccccc')
```

---

formatfunc *Function-returning wrapper for format*

Description

This function takes a set of options for the format() function and returns a function that itself calls format() with those settings.

Usage

```
formatfunc(
  percent = FALSE,
  prefix = "",
  suffix = "",
  scale = 1,
  digits = NULL,
```

```
  nsmall = 0L,
  big.mark = "",
  trim = TRUE,
  scientific = FALSE,
  ...
)
```

Arguments

percent	Whether to apply percentage formatting. Set to TRUE if 1 = 100%. Or, optionally, set to any other number that represents 100%. So percent = TRUE or percent = 1 will interpret .9 as 90%, or percent = 100 will format 90 as 90%.
prefix	A prefix to apply to the formatted number. For example, prefix = '$' would format 4 as $4.
suffix	A suffix to apply to the formatted number. If specified alongside percent, the suffix comes after the %.
scale	A scalar value to be multiplied by all numbers prior to formatting. scale = 1/1000, for example, would convert the units into thousands. This is applied before digits.
digits	Number of significant digits.
nsmall	The minimum number of digits to the right of the decimal point.
big.mark	A character to mark thousands places, for example producing "1,000" instead of "1000".
trim	Whether numbers should be trimmed to their own size, rather than being right-justified to a common width. Unlike the actual format(), this defaults to TRUE. Note that in most vtable applications, the formatting function is applied one value at a time, rather than to a vector, so trim = FALSE may not work as intended.
scientific	Whether numbers should be encoded in scientific format. Unlike the actual format(), this defaults to FALSE.
...	Arguments to be passed to format(). See help(format). All other parameters listed above except for percent, prefix, or suffix are also just part of format, but may be of particular interest, or have been included to show how defaults have changed.

Details

The only differences are:

1. scientific is set to FALSE by default, and trim is set to TRUE 2. Passing a NA value produces '' instead of 'NA'. 3. In addition to standard format() options, it also accepts a percent option to apply percentage formatting, and prefix and suffix options to apply prefixes or suffixes to formatted numbers. 4. Has an attribute 'big.mark' storing the 'big.mark' option chosen.

This is in the spirit of the label_ functions in the scales package, except that it uses format()'s focus on significant digits instead of fixed decimal places, which is good for numbers that range across multiple orders of magnitude, common in sumtable() and vtable().

Examples

```
x <- c(1, 1000, .000235, 1298.255, NA)
my.formatting.func = formatfunc(digits = 3, prefix = '$')
my.formatting.func(x)
```

`independence.test` *Group-Independence Test Function*

Description

This function takes in two variables of equal length, the first of which is a categorical variable, and performs a test of independence between them. It returns a character string with the results of that test for putting in a table.

Usage

```
independence.test(
  x,
  y,
  w = NA,
  factor.test = NA,
  numeric.test = NA,
  star.cutoffs = c(0.01, 0.05, 0.1),
  star.markers = c("***", "**", "*"),
  digits = 3,
  fixed.digits = FALSE,
  format = "{name}={stat}{stars}",
  opts = list()
)
```

Arguments

`x`	A categorical variable.
`y`	A variable to test for independence with `x`. This can be a factor or numeric variable. If you want a numeric variable treated as categorical, convert to a factor first.
`w`	A vector of weights to pass to the appropriate test.
`factor.test`	Used when `y` is a factor, a function that takes `x` and `y` as its first arguments and returns a list with three arguments: (1) The name of the test for printing, (2) the test statistic, and (3) the p-value. Defaults to a Chi-squared test if there are no weights, or a design-based F statistic (Rao & Scott Aadjustment, see `survey::svychisq`) with weights, which requires that the `survey` package be installed. WARNING: the Chi-squared test's assumptions fail with small sample sizes. This function will be attempted for all non-numeric `y`.

numeric.test	Used when y is numeric, a function that takes x and y as its first arguments and returns a list with three arguments: (1) The name of the test for printing, (2) the test statistic, and (3) the p-value. Defaults to a group differences F test.
star.cutoffs	A numeric vector indicating the p-value cutoffs to use for reporting significance stars. Defaults to c(.01,.05,.1). If you don't want stars, remove them from the format argument.
star.markers	A character vector indicating the symbols to use to indicate significance cutoffs associated with star.cuoffs. Defaults to c('***','**','*'). If you don't want stars, remove them from the format argument.
digits	Number of digits after the decimal to round the test statistic and p-value to.
fixed.digits	FALSE will cut off trailing 0s when rounding. TRUE retains them. Defaults to FALSE.
format	The way in which the four elements returned by (or calculated after) the test - {name}, {stat}, {pval}, and {stars} - will be arranged in the string output. Note that the default '{name}={stat}{stars}' does not contain the p-value, and also does not contain superscript for the stars since it doesn't know what markup language you're aiming for. For LaTeX you may prefer '{name}$={stat}^{{stars}}$', and for HTML '{name}={stat}^{stars}'.
opts	The options listed above, entered in named-list format.

Details

In an attempt (and perhaps an encouragement) to use this function in weird ways, and because it's not really expected to be used directly, input is not sanitized. Have fun!

Examples

```
data(mtcars)
independence.test(mtcars$cyl,mtcars$mpg)
```

---

is.round *Checks if information is lost by rounding*

Description

This function takes a vector and checks if any information is lost by rounding to a certain number of digits.

Usage

```
is.round(x, digits = 0)
```

Arguments

x	A vector.
digits	How many digits to round to.

Details

Returns TRUE if rounding to digits digits after the decimal can be done without losing information.

Examples

```
is.round(1:5)

x <- c(1, 1.2, 1.23)
is.round(x)
is.round(x,digits=2)
```

---

labeltable *Label Table Function*

Description

This function output a descriptive table listing, for each value of a given variable, either the label of that value, or all values of another variable associated with that value. The table is output either to the console or as an HTML file that can be viewed continuously while working with data.

Usage

```
labeltable(
  var,
  ...,
  out = NA,
  count = FALSE,
  percent = FALSE,
  file = NA,
  desc = NA,
  note = NA,
  note.align = NA,
  anchor = NA
)
```

Arguments

var	A vector. Label table will show, for each of the values of this variable, its label (if labels can be found with sjlabelled::get_labels()), or the values in the ... variables.
...	As described above. If specified, will show the values of these variables, instead of the labels of var, even if labels can be found.

out	Determines where the completed table is sent. Set to "browser" to open HTML file in browser using browseURL(), "viewer" to open in RStudio viewer using viewer(), if available. Use "htmlreturn" to return the HTML code to R, "return" to return the completed variable table to R in data frame form, or "kable" to return it in knitr::kable() form. Combine out = "csv" with file to write to CSV (dropping most formatting). Additional options include "latex" for a LaTeX table or "latexpage" for a full buildable LaTeX page. Defaults to "viewer" if RStudio is running, "browser" if it isn't, or a "kable" passed through kableExtra::kable_styling() defaults if it's an RMarkdown document being built with knitr.
count	Set to TRUE to also report the number of observations for each value of var in the data.
percent	Set to TRUE to also report the percentage of non-missing observation for each value of var in the data.
file	Saves the completed variable table file to HTML with this filepath. May be combined with any value of out, although note that out = "return" and out = "kable" will still save the standard labeltable HTML file as with out = "viewer" or out = "browser"..
desc	Description of variable (or labeling system) to be included with the table.
note	Table note to go after the last row of the table.
note.align	Set the alignment for the multi-column table note. Usually "l", but if you have a long note in LaTeX you might want to set it with "p"
anchor	Character variable to be used to set an anchor link in HTML tables, or a label tag in LaTeX.

Details

Outputting the label table as a help file will make it easy to search through value labels, or to see the correspondence between the values of one variable and the values of another.

Labels that are not in the data will also be reported in the table.

Examples

```
if(interactive()){
#Input a single labelled variable to see a table relating values to labels.
#Values not present in the data will be included in the table but moved to the end.
library(sjlabelled)
data(efc)
labeltable(efc$e15relat)

#Include multiple variables to see, for each value of the first variable,
#each value of the others present in the data.
data(efc)
labeltable(efc$e15relat,efc$e16sex,efc$e42dep)

#Commonly, the multi-variable version might be used to recover the original
#values of encoded variables
```

```
data(USJudgeRatings)
USJudgeRatings$Judge <- row.names(USJudgeRatings)
USJudgeRatings$JudgeID <- as.numeric(as.factor(USJudgeRatings$Judge))
labeltable(USJudgeRatings$JudgeID,USJudgeRatings$Judge)
}
```

---

notNA *Number of nonmissing values in a vector*

Description

This function calculates the number of values in a vector that are not NA.

Usage

```
notNA(x, big.mark = NULL, scientific = FALSE, ...)
```

Arguments

x	A vector.
big.mark	Argument to pass to format(), if a formatted string is desired.
scientific	Argument to pass to format() if big.mark is specified. Defaults to FALSE, unlike in format().
...	Other arguments to pass to format(). Ignored if big.mark is not specified.

Details

This function just shorthand for sum(!is.na(x)), with a shorter name for reference in the vtable or sumtable summ option.

If big.mark is specified, will return a formatted string instead of a number, where the formatting is based on format(x, big.mark = big.mark, scientific = FALSE, ...).

Examples

```
x <- c(1, 1, NA, 2, 3, NA)
notNA(x)
notNA(1:10000, big.mark = ',')
```

nuniq *Number of unique values in a vector*

Description

This function takes a vector and returns the number of unique values in that vector.

Usage

```
nuniq(x)
```

Arguments

x	A vector.

Details

This function is just shorthand for `length(unique(x))`, with a shorter name for reference in the `vtable` or `sumtable` `summ` option.

Examples

```
x <- c(1, 1, 2, 3, 4, 4, 4)
nuniq(x)
```

pctile *Returns a vector of 100 percentiles*

Description

This function calculates 100 percentiles of a vector and returns all of them.

Usage

```
pctile(x)
```

Arguments

x	A vector.

Details

This function just shorthand for `quantile(x,1:100/100)`, with a shorter name for reference in the `vtable` or `sumtable` `summ` option, and which works with `sumtable` `summ.names` styling.

Examples

```
x <- 1:500
pctile(x)[50]
quantile(x,.5)
median(x)
```

---

propNA *Proportion or number of missing values in a vector*

Description

This function calculates the proportion of values in a vector that are NA.

Usage

```
propNA(x)
```

Arguments

x	A vector.

Details

This function just shorthand for `mean(is.na(x))`, with a shorter name for reference in the `vtable` or `sumtable` `summ` option.

Examples

```
x <- c(1, 1, NA, 2, 3, NA)
propNA(x)
```

---

sumtable *Summary Table Function*

Description

This function will output a summary statistics variable table either to the console or as an HTML file that can be viewed continuously while working with data, or sent to file for use elsewhere. `st()` is the same thing but requires fewer key presses to type.

Usage

```
sumtable(
  data,
  vars = NA,
  out = NA,
  file = NA,
  summ = NA,
  summ.names = NA,
  add.median = FALSE,
  group = NA,
  group.long = FALSE,
  group.test = FALSE,
  group.weights = NA,
  col.breaks = NA,
  digits = 2,
  fixed.digits = FALSE,
  numformat = formatfunc(digits = digits, big.mark = ""),
  skip.format = c("notNA(x)", "propNA(x)", "countNA(x)", obs.function),
  factor.percent = TRUE,
  factor.counts = TRUE,
  factor.numeric = FALSE,
  logical.numeric = FALSE,
  logical.labels = c("No", "Yes"),
  labels = NA,
  title = "Summary Statistics",
  note = NA,
  anchor = NA,
  col.width = NA,
  col.align = NA,
  align = NA,
  note.align = "l",
  fit.page = "\\textwidth",
  simple.kable = FALSE,
  obs.function = NA,
  opts = list()
)

st(
  data,
  vars = NA,
  out = NA,
  file = NA,
  summ = NA,
  summ.names = NA,
  add.median = FALSE,
  group = NA,
  group.long = FALSE,
  group.test = FALSE,
```

```
  group.weights = NA,
  col.breaks = NA,
  digits = 2,
  fixed.digits = FALSE,
  numformat = formatfunc(digits = digits, big.mark = ""),
  skip.format = c("notNA(x)", "propNA(x)", "countNA(x)", obs.function),
  factor.percent = TRUE,
  factor.counts = TRUE,
  factor.numeric = FALSE,
  logical.numeric = FALSE,
  logical.labels = c("No", "Yes"),
  labels = NA,
  title = "Summary Statistics",
  note = NA,
  anchor = NA,
  col.width = NA,
  col.align = NA,
  align = NA,
  note.align = "l",
  fit.page = "\\textwidth",
  simple.kable = FALSE,
  obs.function = NA,
  opts = list()
)
```

Arguments

`data`	Data set; accepts any format with column names.
`vars`	Character vector of column names to include, in the order you'd like them included. Defaults to all numeric, factor, and logical variables, plus any character variables with six or fewer unique values. You can include strings that aren't columns in the data (including blanks) - these will create rows that are blank except for the string (left-aligned), for spacers or subtitles.
`out`	Determines where the completed table is sent. Set to `"browser"` to open HTML file in browser using `browseURL()`, `"viewer"` to open in RStudio viewer using `viewer()`, if available. Use `"htmlreturn"` to return the HTML code to R, `"latex"` to return LaTeX code to R (use `"latexdoc"` to get a full buildable document rather than a fragment), `"return"` to return the completed summary table to R in data frame form, or `"kable"` to return it in `knitr::kable()` form. Combine `out = "csv"` with `file` to write to CSV (dropping most formatting). Defaults to `"viewer"` if RStudio is running, `"browser"` if it isn't, or a `"kable"` passed through `kableExtra::kable_styling()` defaults if it's an RMarkdown document being built with `knitr`.
`file`	Saves the completed summary table file to file with this filepath. May be combined with any value of `out`, although note that `out = "return"` and `out = "kable"` will still save the standard sumtable HTML file as with `out = "viewer"` or `out = "browser"`.

summ — Character vector of summary statistics to include for numeric and logical variables, in the form 'function(x)'. Defaults to c('notNA(x)','mean(x)','sd(x)','min(x)','pctile if there's one column, or c('notNA(x)','mean(x)','sd(x)') if there's more than one. If all variables in a column are factors it defaults to c('sum(x)','mean(x)') for the factor dummies. If the table has multiple variable-columns and you want different statistics in each, include a list of character vectors instead. This option is flexible, and allows any summary statistic function that takes in a column and returns a single number. For example, summ=c('mean(x)','mean(log(x))') will provide the mean of each variable as well as the mean of the log of each variable. Keep in mind the special vtable package helper functions designed specifically for this option propNA, countNA, notNA, and notNA, which report counts and proportions of NAs, or counts of not-NAs, in the vectors, nuniq, which reports the number of unique values, and pctile, which returns a vector of the 100 percentiles of the variable. NAs will be omitted from all calculations other than propNA(x) and countNA(x).

summ.names — Character vector of names for the summary statistics included. If summ is at default, defaults to c('N','Mean','Std. Dev.','Min','Pctl. 25','Pctl. 75','Max') (or the appropriate shortened version with multiple columns) unless all variables in the column are factors in which case it defaults to c('N','Percent'). If summ has been set but summ.names has not, defaults to summ with the (x)s removed and the first letter capitalized. If the table has multiple variable-columns and you want different statistics in each, include a list of character vectors instead.

add.median — Adds "median(x)" to the set of default summary statistics. Has no effect if "summ" is also specified.

group — Character variable with the name of a column in the data set that statistics are to be calculated over. Value labels will be used if found for numeric variables. Changes the default summ to c('mean(x)','sd(x)').

group.long — By default, if group is specified, each group will get its own set of columns. Set group.long = TRUE to instead basically just make a regular sumtable() for each group and stack them on top of each other. Good for when you have lots of groups. You can also set it to 'l', 'c', or 'r' to determine how the group names are aligned. Defaults to centered.

group.test — Set to TRUE to perform tests of whether each variable in the table varies over values of group. Only works with group.long = FALSE. Performs a joint F-test (using anova(lm)) for numeric variables, and a Chi-square test of independence (chisq.test) for categorical variables. If you want to adjust things like which tests are used, significance star levels, etc., see the help file for independence.test and pass in a named list of options for that function.

group.weights — *THIS OPTION DOES NOT AUTOMATICALLY WEIGHT ALL CALCULATIONS.* This is mostly to be used with group and group.long = FALSE, and while it's more flexible than that, you've gotta read this to figure out how else to use it. That's why I gave it the weird name. Set this to a vector of weights, or a string representing a column name with weights. If summ is not customized, this will replace 'mean(x)' and 'sd(x)' with the equivalent weighted versions 'weighted.mean(x, w = wts)' and 'weighted.sd(x, w = wts)' It will also add weights to the default group.test tests. This will not add weights

to any other calculations, or to any custom group.test weights (although you can always do that yourself by customizing summ and passing in weights with this argument-the weights can be referred to in your function as wts). This is generally intended for things like post-matching balance tables. If you specify a column name, that column will be removed from the rest of the table, so if you want it to be kept, specify this as a numeric vector instead. If you have a variable in your data called 'wts' that will mess the use of this option up, I recommend changing that.

col.breaks Numeric vector indicating the variables (or number of elements of vars) after which to start a new column. So for example with a data set with six variables, c(3,5) would put the first three variables in the first column, the next two in the middle, and the last on the right. Cannot be combined with group unless group.long = TRUE.

digits Number of digits after the decimal place to report. Set to a single number for consistent digits, or a vector the same length as summ for different digits for each calculation, or a list of vectors that match up to a multi-column summ. Defaults to 0 for the first calculation (N, usually) and 2 afterwards.

fixed.digits Deprecated; currently only works if numformat = NA. FALSE will cut off trailing 0s when rounding. TRUE retains them. Defaults to FALSE.

numformat A function that takes a numeric input and produces labeled output, which you might construct using the formatfunc function or the label_ functions from the scales package. Provide a single function to apply to all variables, or a list of functions the same length as the number of variables to format each variable differently. The formatting function will skip over notNA, countNA, propNA calculations by default. Factor percentages will ignore this entirely; you can use NA to skip them when specifying a list. Alternately, you can specify strings giving the shorthand for the appropriate formatting: the string containing 'comma' will set big.mark = ',', 'decimal' will set big.mark = '.', decimal.mark = ',', 'percent' will do percentage formatting (with 1 = 100%), and 'A|B' will use 'A' as a prefix and 'B' as a suffix (specifying suffix optional, so numformat = '$' gives '$3'). Anything more complex than that will require you pass a formatfunc or similar function. Specifying a character vector will respect your digits option if digits is a single value rather than a vector or list, but will otherwise use the defaults of those functions. You can mix together specifying your own functions and specifying character strings. At the moment there is no way to do different formatting for different columns of the same variable, other than skip.format. Set to NA to revert to the old way of formatting.

skip.format Set of functions in summ that are not subject to format. Does nothing if format is not specified.

factor.percent Set to TRUE to show factor means as percentages instead of proportions, i.e. 50% with a column header of "Percent" rather than .5 with a column header of "Mean". Defaults to TRUE.

factor.counts Set to TRUE to show a count of each factor level in the first column. Defaults to TRUE.

factor.numeric By default, factor variable dummies basically ignore the summ argument and show count (or nothing) in the first column and percent or proportion in the sec-

ond. Set this to TRUE to instead treat the dummies like numeric binary variables with values 0 and 1.

`logical.numeric`
: By default, logical variables are treated as factors with TRUE = "Yes" and FALSE = "No". Set this to FALSE to instead treat them as numeric variables rather than factors, with TRUE = 1 and FALSE = 0.

`logical.labels`
: When turning logicals into factors, use these labels for FALSE and TRUE, respectively, rather than "No" and "Yes".

`labels`
: Variable labels. labels will accept four formats: (1) A vector of the same length as the number of variables in the data that will be included in the table (tricky to use if many are being dropped, also won't work for your group variable), in the same order as the variables in the data set, (2) A matrix or data frame with two columns and more than one row, where the first column contains variable names (in any order) and the second contains labels, (3) A matrix or data frame where the column names (in any order) contain variable names and the first row contains labels, or (4) TRUE to look in the data for variable labels set by the haven package, set_label() from sjlabelled, or label() from Hmisc.

`title`
: Character variable with the title of the table.

`note`
: Table note to go after the last row of the table. Will follow significance star note if group.test = TRUE.

`anchor`
: Character variable to be used to set an anchor link in HTML tables, or a label tag in LaTeX.

`col.width`
: Vector of page-width percentages, on 0-100 scale, overriding default column widths in an HTML table. Must have a number of elements equal to the number of columns in the resulting table.

`col.align`
: For HTML output, a character vector indicating the HTML text-align attributes to be used in the table (for example col.align = c('left','center','center'). Defaults to variable-name columns left-aligned and all others right-aligned (with a little extra padding between columns with col.breaks). If you want to get tricky, you can add a ";" afterwards and keep putting in whatever CSS attributes you want. They will be applied to the whole column.

`align`
: For LaTeX output, string indicating the alignment of each column. Use standard LaTeX syntax (i.e. l|ccc). Defaults to left in the first column and right-aligned afterwards, with @{\hskip .2in} spacers if you have col.breaks. If col.width is specified, defaults to all p{} columns with widths set by col.width. If you want the columns aligned on a decimal point, see this explainer.

`note.align`
: For LaTeX output, set the alignment for the multi-column table note. Usually "l", but if you have a long note in LaTeX you might want to set it with "p"

`fit.page`
: For LaTeX output, uses a resizebox to force the table to a certain width. Set to NA to omit.

`simple.kable`
: For out = 'kable', if you want the kable printed to console rather than HTML or PDF, then the multi-column headers and table notes won't work. Set simple.kable = TRUE to skip both.

`obs.function`
: The function to use (and, potentially, format) to count the number of observations for the N column. This should take a vector and return a single number

or string. Uses the same string formatting as summ. If not specified, will check if numformat is specified using formatfunc or a string. If not, this will be 'notNA(x)'. If it is, will be 'notNA(x)' with the big.mark argument set to match the first function listed in numformat.

opts — The same sumtable options as above, but in a named list format. Useful for applying the same set of options to multiple sumtables.

Details

There are many, many functions in R that will produce a summary statisics table for you. So why use sumtable()? sumtable() serves two main purposes:

(1) In the same spirit as vtable(), it makes it easy to view the summary statistics *as you work*, either in the Viewer pane or in a browser window.

(2) sumtable() is designed to *have nice defaults* and is not really intended for deep customization. It's got lots of options, sure, but they're only intended to go so far. So you can have a summary statistics table without much work.

Keeping with point (2), sumtable() is designed for use by people who want the kind of table that sumtable() produces, which is itself heavily influenced by the kinds of summary statistics tables you often see in economics papers. In that regard it is most similar to stargazer::stargazer() except that it can handle tibbles, factor variables, grouping, and produce multicolumn tables, or summarytools::dfSummary() or skimr::skim() except that it is easier to export with nice formatting. If you want a lot of control over your summary statistics table, check out the packages gtsummary, arsenal, qwraps2, or Amisc, and about a million more.

If you would like to include a sumtable in an RMarkdown document, it should just work! If you leave out blank, it will default to a nicely-formatted knitr::kable(), although this will drop some formatting elements like multi-column cells (or do out="kable" to get an unformatted kable that you can format yourself). If you prefer the vtable package formatting, then use out="latex" if outputting to LaTeX or out="htmlreturn" for HTML, both with results="asis" in the code chunk. Alternately, in HTML, you can use the file option to write to file and use a <iframe> to include it.

Examples

```
# Examples are only run interactively because they open HTML pages in Viewer or a browser.
if (interactive()) {
data(iris)

# Sumtable handles both numeric and factor variables
st(iris)

# Output to LaTeX as well for easy integration
# with RMarkdown, or \input{} into your LaTeX docs
# (specify file too  to save the result)
st(iris, out = 'latex')

# Summary statistics by group
iris$SL.above.median <- iris$Sepal.Length > median(iris$Sepal.Length)
st(iris, group = 'SL.above.median')
```

```
# Add a group test, or report by-group in "long" format
st(iris, group = 'SL.above.median', group.test = TRUE)
st(iris, group = 'SL.above.median', group.long = TRUE)

# Going all out! Adding variable labels with labels,
# spacers and variable "category" titles with vars,
# Changing the presentation of the factor variable,
# and putting the factor in its own column with col.breaks
var.labs <- data.frame(var = c('SL.above.median','Sepal.Length',
                               'Sepal.Width','Petal.Length',
                               'Petal.Width'),
                       labels = c('Above-median Sepal Length','Sepal Length',
                       'Sepal Width','Petal Length',
                       'Petal Width'))
st(iris,
    labels = var.labs,
    vars = c('Sepal Variables','SL.above.median','Sepal.Length','Sepal.Width',
    'Petal Variables','Petal.Length','Petal.Width',
    'Species'),
    factor.percent = FALSE,
    col.breaks = 7)

# Format the results
# use rep so there are enough observations to see the comma separators
irisrep = do.call('rbind', replicate(100, iris, simplify = FALSE))
# Comma separator for thousands, including for N.
st(irisrep, numformat = 'comma')
# Dollar formatting for sepal.width, decimal (1.000,00) formatting for the rest
st(iris, numformat = c('decimal','Sepal.Width' = '$'))
# Custom formatting throughout, note the big.mark = ',' will also be picked up by N
st(irisrep, numformat = formatfunc(digits = 2, nsmall = 2, big.mark = ','))

}
```

`vtable` *Variable Table Function*

Description

This function will output a descriptive variable table either to the console or as an HTML file that can be viewed continuously while working with data. `vt()` is the same thing but requires fewer key presses to type.

Usage

```
vtable(
  data,
  out = NA,
  file = NA,
  labels = NA,
```

```
  class = TRUE,
  values = TRUE,
  missing = FALSE,
  index = FALSE,
  factor.limit = 5,
  char.values = FALSE,
  data.title = NA,
  desc = NA,
  note = NA,
  note.align = "l",
  anchor = NA,
  col.width = NA,
  col.align = NA,
  align = NA,
  fit.page = NA,
  summ = NA,
  lush = FALSE,
  opts = list()
)

vt(
  data,
  out = NA,
  file = NA,
  labels = NA,
  class = TRUE,
  values = TRUE,
  missing = FALSE,
  index = FALSE,
  factor.limit = 5,
  char.values = FALSE,
  data.title = NA,
  desc = NA,
  note = NA,
  note.align = "l",
  anchor = NA,
  col.width = NA,
  col.align = NA,
  align = NA,
  fit.page = NA,
  summ = NA,
  lush = FALSE,
  opts = list()
)
```

Arguments

`data`	Data set; accepts any format with column names. If variable labels are set with the haven package, `set_label()` from sjlabelled, or `label()` from Hmisc,

vtable will extract them automatically.

out — Determines where the completed table is sent. Set to "browser" to open HTML file in browser using browseURL(), "viewer" to open in RStudio viewer using viewer(), if available. Use "htmlreturn" to return the HTML code to R. Use "return" to return the completed variable table to R in data frame form or "kable" to return it as a knitr::kable(). Additional options include "csv" to write to CSV in conjunction with file (although this will drop most additional formatting), "latex" for a LaTeX table or "latexpage" for a full buildable LaTeX page. Defaults to "viewer" if RStudio is running, "browser" if it isn't, or a "kable" passed through kableExtra::kable_styling() defaults if it's an RMarkdown document being built with knitr.

file — Saves the completed variable table file to HTML or .tex with this filepath. May be combined with any value of out, although note that out = "return" and out = "kable" will still save the standard vtable HTML file as with out = "viewer" or out = "browser".

labels — Variable labels. labels will accept three formats: (1) A vector of the same length as the number of variables in the data, in the same order as the variables in the data set, (2) A matrix or data frame with two columns and more than one row, where the first column contains variable names (in any order) and the second contains labels, or (3) A matrix or data frame where the column names (in any order) contain variable names and the first row contains labels. Setting the labels parameter will override any variable labels already in the data. Set to "omit" if the data set has embedded labels but you don't want any labels in the table.

class — Set to TRUE to include variable classes in the variable table. Defaults to TRUE.

values — Set to TRUE to include the range of values of each variable: min and max for numeric variables, list of factors for factor or ordered variables, and 'TRUE FALSE' for logicals. values will detect and use value labels set by the sjlabelled or haven packages, as long as every value is labelled. Defaults to TRUE.

missing — Set to TRUE to include the number of NAs in the variable. Defaults to FALSE.

index — Set to TRUE to include the index number of the column with the variable name. Defaults to FALSE.

factor.limit — Sets maximum number of factors that will be included if values = TRUE. Set to 0 for no limit. Defaults to 5.

char.values — Set to TRUE to include values of character variables as though they were factors, if values = TRUE. Or, set to a character vector of variable names to list values of only those character variables. Defaults to FALSE. Has no effect if values = FALSE.

data.title — Character variable with the title of the dataset.

desc — Character variable offering a brief description of the dataset itself. This will by default include information on the number of observations and the number of columns. To remove this, set desc='omit', or include any description and then include 'omit' as the last four characters.

note — Table note to go after the last row of the table.

note.align — Set the alignment for the multi-column table note. Usually "l", but if you have a long note in LaTeX you might want to set it with "p"

anchor — Character variable to be used to set an anchor link in HTML tables, or a label tag in LaTeX.

col.width — Vector of page-width percentages, on 0-100 scale, overriding default column widths in HTML table. Must have a number of elements equal to the number of columns in the resulting table.

col.align — For HTML output, a character vector indicating the HTML text-align attributes to be used in the table (for example col.align = c('left','center','center'). Defaults to all left-aligned. If you want to get tricky, you can add a ";" afterwards and keep putting in whatever CSS attributes you want. They will be applied to the whole column.

align — For LaTeX output, string indicating the alignment of each column. Use standard LaTeX syntax (i.e. l|ccc). Defaults to all p{} columns with widths set using the same defaults as with col.width. Be sure to escape special characters, in particular backslashes (i.e. p{.25\\textwidth} instead of p{.25\textwidth}).

fit.page — For LaTeX output, uses a resizebox to force the table to a certain width. Set to NA to omit. Often '\textwidth'.

summ — Character vector of summary statistics to include for numeric and logical variables, in the form 'function(x)'. This option is flexible, and allows any summary statistic function that takes in a column and returns a single number. For example, summ=c('mean(x)','mean(log(x))') will provide the mean of each variable as well as the mean of the log of each variable. Keep in mind the special vtable package helper functions designed specifically for this option propNA, countNA, and notNA, which report counts and proportions of NAs, or counts of not-NAs, in the vectors, nuniq, which reports the number of unique values, and pctile, which returns a vector of the 100 percentiles of the variable. NAs will be omitted from all calculations other than propNA(x) and countNA(x).

lush — Set to TRUE to select a set of options with more information: sets char.values and missing to TRUE, and sets summ to c('mean(x)', 'sd(x)', 'nuniq(x)'). summ can be overwritten by setting summ to something else.

opts — The same vtable options as above, but in a named list format. Useful for applying the same set of options to multiple vtables.

Details

Outputting the variable table as a help file will make it easy to search through variable names or labels, or to refer to information about the variables easily.

This function is in a similar spirit to promptData(), but focuses on variable documentation rather than dataset documentation.

If you would like to include a vtable in an RMarkdown document, it should just work! If you leave out blank, it will default to a nicely-formatted knitr::kable(), although this will drop some formatting elements like multi-column cells (or do out="kable" to get an unformatted kable that you can format yourself). If you prefer the vtable package formatting, then use out="latex" if outputting to LaTeX or out="htmlreturn" for HTML, both with results="asis" in the code chunk. Alternately, in HTML, you can use the file option to write to file and use a <iframe> to include it.

Examples

```
if(interactive()){
df <- data.frame(var1 = 1:4,var2=5:8,var3=c('A','B','C','D'),
    var4=as.factor(c('A','B','C','C')),var5=c(TRUE,TRUE,FALSE,FALSE))

#Demonstrating different options:
vtable(df,labels=c('Number 1','Number 2','Some Letters',
    'Some Labels','You Good?'))
vtable(subset(df,select=c(1,2,5)),
    labels=c('Number 1','Number 2','You Good?'),class=FALSE,values=FALSE)
vtable(subset(df,select=c('var1','var4')),
    labels=c('Number 1','Some Labels'),
    factor.limit=1,col.width=c(10,10,40,35))

#Different methods of applying variable labels:
labelsmethod2 <- data.frame(var1='Number 1',var2='Number 2',
    var3='Some Letters',var4='Some Labels',var5='You Good?')
vtable(df,labels=labelsmethod2)
labelsmethod3 <- data.frame(a =c("var1","var2","var3","var4","var5"),
    b=c('Number 1','Number 2','Some Letters','Some Labels','You Good?'))
vtable(df,labels=labelsmethod3)

#Using value labels and pre-labeled data:
library(sjlabelled)
df <- set_label(df,c('Number 1','Number 2','Some Letters',
    'Some Labels','You Good?'))
df$var1 <- set_labels(df$var1,labels=c('A little','Some more',
'Even more','A lot'))
vtable(df)

#efc is data with embedded variable and value labels from the sjlabelled package
library(sjlabelled)
data(efc)
vtable(efc)

#Displaying the values of a character vector
data(USJudgeRatings)
USJudgeRatings$Judge <- row.names(USJudgeRatings)
vtable(USJudgeRatings,char.values=c('Judge'))

#Adding summary statistics for variable mean and proportion of data that is missing.
vtable(efc,summ=c('mean(x)','propNA(x)'))

}
```

`weighted.sd` *Weighted standard deviation*

Description

This is a basic weighted standard deviation function, mainly for internal use with `sumtable`. For a more fully-fledged weighted SD function, see `Hmisc::wtd.var`, although it uses a slightly differend degree-of-freedom correction.

Usage

```
weighted.sd(x, w, na.rm = TRUE)
```

Arguments

`x`	A numeric vector.
`w`	A vector of weights. Negative weights are not allowed.
`na.rm`	Set to `TRUE` to remove indices with missing values in `x` or `w`.

Examples

```
x <- c(1, 1, 2, 3, 4, 4, 4)
w <- c(4, 1, 3, 7, 0, 2, 5)
weighted.sd(x, w)
```

Index