Package ‘modelsummary’

September 4, 2020

Type Package

Title Summary Tables and Plots for Statistical Models and Data: Beautiful, Customizable, and Publication-Ready

Description Create beautiful and customizable tables to summarize several statistical models side-by-side. Draw coefficient plots, multi-level cross-tabs, dataset summaries, balance tables (a.k.a. ``Table 1s''), and correlation matrices. This package supports dozens of statistical models, and it can produce tables in HTML, LaTeX, Word, Markdown, PDF, PowerPoint, Excel, RTF, JPG, or PNG. Tables can easily be embedded in 'Rmarkdown' or 'knitr' dynamic documents.

Version 0.6.1

URL `https://vincentarelbundock.github.io/modelsummary/`

BugReports `https://github.com/vincentarelbundock/modelsummary/issues/`

Depends R (>= 3.4.0)

Imports broom,
checkmate,
dplyr (>= 1.0.0),
generics,
ggplot2,
kableExtra (>= 1.2.1),
knitr,
magrittr,
rmarkdown,
tables,
tibble,
tidyr (>= 1.0.0)

Suggests broom.mixed,
estimatr,
flextable,
gt (>= 0.2.0),
huxtable,
lmtest,
officer,
testthat,
vdiffr

Enhances fixest

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

R topics documented:

`All` *Include all columsn of a dataframe.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `All`

`AllObs` *Display all observations in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `AllObs`

Arguments — *'Arguments' pseudo-function*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Arguments`

datasummary — *Create summary tables using 2-sided formulae: crosstabs, frequencies, table 1s and more.*

Description

Create summary tables using 2-sided formulae: crosstabs, frequencies, table 1s and more.

Usage

```
datasummary(
  formula,
  data,
  output = "default",
  fmt = "%.2f",
  title = NULL,
  notes = NULL,
  align = NULL,
  add_columns = NULL,
  add_rows = NULL,
  sparse_header = TRUE
)
```

Arguments

`formula`	A two-sided formula to describe the table: rows ~ columns. See the Examples section for a mini-tutorial and the Details section for more resources.
`data`	A data.frame (or tibble)
`output`	filename or object type (string) • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg. • Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable". • When a file name is supplied to the 'output' argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the 'gt' or 'kableExtra' packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the 'gt::gtsave', 'kable::save_kable', or 'cat' functions.

`fmt`	string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. ’%.3f’ will keep 3 digits after the decimal point with trailing zero. ’%.5f’ will keep 5 digits. ’%.3e’ will use exponential notation. See ‘?sprintf‘ for more options.
`title`	string
`notes`	list or vector of notes to append to the bottom of the table.
`align`	A character string of length equal to the number of columns in the table. "lcr" means that the first column will be left-aligned, the 2nd column center-aligned, and the 3rd column right-aligned.
`add_columns`	a data.frame (or tibble) with the same number of rows as your main table.
`add_rows`	a data.frame (or tibble) with the same number of columns as your main table. By default, rows are appended to the bottom of the table. You can define a "position" attribute of integers to set the row positions. See examples.
`sparse_header`	TRUE or FALSE. TRUE eliminates column headers which have a unique label across all columns, except for the row immediately above the data. FALSE keeps all headers. The order in which terms are entered in the formula determines the order in which headers appear. For example, ‘x~mean*z‘ will print the ‘mean‘-related header above the ‘z‘-related header.‘

Details

Visit the ’modelsummary’ website for more usage examples: https://vincentarelbundock.github.io/modelsummary

The ’datasummary’ function is a thin wrapper around the ’tabular’ function from the ’tables’ package. More details about table-making formulas can be found in the ’tables’ package documentation: ?tables::tabular

Hierarchical or "nested" column labels are only available for these output formats: kableExtra, gt, html, rtf, and LaTeX. When saving tables to other formats, nested labels will be combined to a "flat" header.

Examples

```
## Not run:

# The left-hand side of the formula describes rows, and the right-hand side
# describes columns. This table uses the "mpg" variable as a row and the "mean"
# function as a column:

datasummary(mpg ~ mean, data = mtcars)

# This table uses the "mean" function as a row and the "mpg" variable as a column:

datasummary(mean ~ mpg, data = mtcars)

# Display several variables or functions of the data using the "+"
# concatenation operator. This table has 2 rows and 2 columns:

datasummary(hp + mpg ~ mean + sd, data = mtcars)

# Nest variables or statistics inside a "factor" variable using the "*" nesting
# operator. This table shows the mean of "hp" and "mpg" for each value of
# "cyl":
```

```
mtcars$cyl <- as.factor(mtcars$cyl)
datasummary(hp + mpg ~ cyl * mean, data = mtcars)

# If you don't want to convert your original data
to factors, you can use the 'Factor()'
# function inside 'datasummary' to obtain an identical result:

datasummary(hp + mpg ~ Factor(cyl) * mean, data = mtcars)

# You can nest several variables or statistics inside a factor by using
# parentheses. This table shows the mean and the standard deviation for each
# subset of "cyl":

datasummary(hp + mpg ~ cyl * (mean + sd), data = mtcars)

# Summarize all numeric variables with 'All()'
datasummary(All(mtcars) ~ mean + sd, data = mtcars)

# Define custom summary statistics. Your custom function should accept a vector
# of numeric values and return a single numeric or string value:

minmax <- function(x) sprintf("[%.2f, %.2f]", min(x), max(x))
mean_na <- function(x) mean(x, na.rm = TRUE)

datasummary(hp + mpg ~ minmax + mean_na, data = mtcars)

# To handle missing values, you can pass arguments to your functions using
# '*Arguments()'

datasummary(hp + mpg ~ mean * Arguments(na.rm = TRUE), data = mtcars)

# For convenience, 'modelsummary' supplies several convenience functions
# with the argument `na.rm=TRUE` by default: Mean, Median, Min, Max, SD, Var,
# P0, P25, P50, P75, P100, NUnique, Histogram

datasummary(hp + mpg ~ Mean + SD + Histogram, data = mtcars)

# These functions also accept a 'fmt' argument which allows you to
# round/format the results

datasummary(hp + mpg ~ Mean * Arguments(fmt = "%.3f") + SD * Arguments(fmt = "%.1f"), data = mtcars)

# Save your tables to a variety of output formats:
f <- hp + mpg ~ Mean + SD
datasummary(f, data = mtcars, output = 'table.html')
datasummary(f, data = mtcars, output = 'table.tex')
datasummary(f, data = mtcars, output = 'table.md')
datasummary(f, data = mtcars, output = 'table.docx')
datasummary(f, data = mtcars, output = 'table.pptx')
datasummary(f, data = mtcars, output = 'table.jpg')
datasummary(f, data = mtcars, output = 'table.png')

# Display human-readable code
datasummary(f, data = mtcars, output = 'html')
datasummary(f, data = mtcars, output = 'markdown')
datasummary(f, data = mtcars, output = 'latex')
```

```
# Return a table object to customize using a table-making package
datasummary(f, data = mtcars, output = 'gt')
datasummary(f, data = mtcars, output = 'kableExtra')
datasummary(f, data = mtcars, output = 'flextable')
datasummary(f, data = mtcars, output = 'huxtable')

# add_rows
new_rows <- data.frame(a = 1:2, b = 2:3, c = 4:5)
attr(new_rows, 'position') <- c(1, 3)
datasummary(mpg + hp ~ mean + sd, data = mtcars, add_rows = new_rows)


## End(Not run)
```

datasummary_correlation

Generate a correlation table for all numeric variables in your dataset.

Description

Generate a correlation table for all numeric variables in your dataset.

Usage

```
datasummary_correlation(
  data,
  output = "default",
  fmt = "%.2f",
  title = NULL,
  notes = NULL
)
```

Arguments

`data`	A data.frame (or tibble)
`output`	filename or object type (string) • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg. • Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable". • When a file name is supplied to the ‘output‘ argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the ‘gt‘ or ‘kableExtra‘ packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the ‘gt::gtsave‘, ‘kable::save_kable‘, or ‘cat‘ functions.
`fmt`	string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. ’%.3f’ will keep 3 digits after the decimal point with trailing zero. ’%.5f’ will keep 5 digits. ’%.3e’ will use exponential notation. See ‘?sprintf‘ for more options.

`title`	string
`notes`	list or vector of notes to append to the bottom of the table.

`datasummary_skim` *Quick overview of numeric or categorical variables*

Description

Quick overview of numeric or categorical variables

Usage

```
datasummary_skim(
  data,
  type = "numeric",
  output = "default",
  fmt = "%.1f",
  histogram = FALSE,
  title = NULL,
  notes = NULL,
  align = NULL,
  ...
)
```

Arguments

`data`	A data.frame (or tibble)
`type`	of variables to summarize: "numeric" or "categorical" (character)
`output`	filename or object type (string) • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg. • Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable". • When a file name is supplied to the ‘output‘ argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the ‘gt‘ or ‘kableExtra‘ packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the ‘gt::gtsave‘, ‘kable::save_kable‘, or ‘cat‘ functions.
`fmt`	string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. ’%.3f’ will keep 3 digits after the decimal point with trailing zero. ’%.5f’ will keep 5 digits. ’%.3e’ will use exponential notation. See ‘?sprintf‘ for more options.
`histogram`	TRUE to include a unicode character histogram (boolean)
`title`	string
`notes`	list or vector of notes to append to the bottom of the table.
`align`	A character string of length equal to the number of columns in the table. "lcr" means that the first column will be left-aligned, the 2nd column center-aligned, and the 3rd column right-aligned.

`...`	all other arguments are passed to the 'tidy' method used to extract estimates from the model. For example, this allows users to set 'exponentiate=TRUE' to exponentiate logistic regression coefficients.

DropEmpty *'DropEmpty' pseudo-function*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `DropEmpty`

Factor *Use a variable as a factor to give rows in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Factor`

factory *Factory to create tables in different output formats using standardized inputs.*

Description

Factory to create tables in different output formats using standardized inputs.

Usage

```
factory(
  tab,
  align = NULL,
  fmt = "%.3f",
  hrule = NULL,
  notes = NULL,
  output = NULL,
  title = NULL,
  add_rows = NULL,
  add_columns = NULL,
  ...
)
```

Arguments

`tab`	table body (data.frame)
`align`	A character string of length equal to the number of columns in the table. "lcr" means that the first column will be left-aligned, the 2nd column center-aligned, and the 3rd column right-aligned.
`fmt`	string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. ’%.3f’ will keep 3 digits after the decimal point with trailing zero. ’%.5f’ will keep 5 digits. ’%.3e’ will use exponential notation. See ‘?sprintf‘ for more options.
`hrule`	position of horizontal rules (integer vector)
`notes`	list or vector of notes to append to the bottom of the table.
`output`	filename or object type (string) • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg. • Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable". • When a file name is supplied to the ‘output‘ argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the ‘gt‘ or ‘kableExtra‘ packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the ‘gt::gtsave‘, ‘kable::save_kable‘, or ‘cat‘ functions.
`title`	string
`add_rows`	a data.frame (or tibble) with the same number of columns as your main table. By default, rows are appended to the bottom of the table. You can define a "position" attribute of integers to set the row positions. See examples.
`add_columns`	a data.frame (or tibble) with the same number of rows as your main table.
`...`	all other arguments are passed to the ‘tidy‘ method used to extract estimates from the model. For example, this allows users to set ‘exponentiate=TRUE‘ to exponentiate logistic regression coefficients.

`Format` *Use a variable as a factor to give rows in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Format`

glance_custom

Extract custom information from a model object and turn it into a tidy tibble with a single row.

Description

glance_custom methods always return either a one-row data frame (except on ‘NULL‘, which returns an empty data frame). This

Usage

```
glance_custom(x)
```

Arguments

x	model or other R object to convert to single-row data frame

Methods

No methods found in currently loaded packages.

gof_map

Data.frame used to clean up and format goodness-of-fit statistics

Description

By default, this data frame is passed to the ’gof_map’ argument of the ’msummary’ or ’modelsummary’ functions. Users can modify this data frame to customize the list of statistics to display and their format. See example below.

Usage

```
gof_map
```

Format

data.frame with 4 columns of character data: raw, clean, fmt, omit

Examples

```
## Not run:

library(modelsummary)
mod <- lm(wt ~ drat, data = mtcars)
gm <- modelsummary::gof_map
gm$omit[gm$raw == 'deviance'] <- FALSE
gm$fmt[gm$raw == 'r.squared'] <- "%.5f"
msummary(mod, gof_map = gm)


## End(Not run)
```

`Heading` *'Heading' pseudo-function*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Heading`

`modelplot` *Plot model coefficients using points or point-ranges*

Description

Plot model coefficients using points or point-ranges

Usage

```
modelplot(
  models,
  conf_level = 0.95,
  coef_map = NULL,
  coef_omit = NULL,
  facet = FALSE,
  draw = TRUE,
  background = NULL,
  ...
)
```

Arguments

`models`	a single model object or a (potentially named) list of models to summarize
`conf_level`	confidence level to use for confidence intervals
`coef_map`	named character vector. Values refer to the variable names that will appear in the table. Names refer to the original term names stored in the model object (e.g., "hp:mpg" for an interaction term). Coefficients that are omitted from this vector will be omitted from the table. The table will be ordered in the same order as this vector.
`coef_omit`	string regular expression. Omits all matching coefficients from the table (using 'grepl').
`facet`	TRUE or FALSE. When the 'models' argument includes several model objects, TRUE draws terms in separate facets, and FALSE draws terms side-by-side (dodged).
`draw`	TRUE returns a 'ggplot2' object, FALSE returns the data.frame used to draw the plot.
`background`	A list of 'ggplot2' geoms to add to the background of the plot. This is especially useful to display annotations "behind" the 'geom_pointrange' that 'modelplot' draws.

`...` all other arguments are passed to the ‘tidy‘ method used to extract estimates from the model. For example, this allows users to set ‘exponentiate=TRUE‘ to exponentiate logistic regression coefficients.

Examples

```
## Not run:

library(modelsummary)

# single model
mod <- lm(hp ~ vs + drat, mtcars)
modelplot(mod)

# omit terms with string matches or regexes
modelplot(mod, coef_omit = 'Interc')

# rename, reorder and subset with 'coef_map'
cm <- c('vs' = 'V-shape engine',
        'drat' = 'Rear axle ratio')
modelplot(mod, coef_map = cm)

# several models
models <- list()
models[['Small model']] <- lm(hp ~ vs, mtcars)
models[['Medium model']] <- lm(hp ~ vs + factor(cyl) , mtcars)
models[['Large model']] <- lm(hp ~ vs + drat + factor(cyl), mtcars)
modelplot(models)

# customize your plots with 'ggplot2' functions
library(ggplot2)

modelplot(models) +
    scale_color_brewer(type = 'qual') +
    theme_classic()

# pass arguments to 'geom_pointrange' through the ... ellipsis
modelplot(mod, color = 'red', size = 1, fatten = .5)

# add geoms to the background, behind geom_pointrange
b <- list(geom_vline(xintercept = 0, color = 'orange'),
          annotate("rect", alpha = .1,
                   xmin = -.5, xmax = .5,
                   ymin = -Inf, ymax = Inf),
          geom_point(aes(y = term, x = estimate), alpha = .3,
                     size = 10, color = 'red', shape = 'square'))
modelplot(mod, background = b)


## End(Not run)
```

modelsummary *Beautiful, customizable summaries of statistical models*

Description

Beautiful, customizable summaries of statistical models

Usage

```
modelsummary(
  models,
  output = "default",
  fmt = "%.3f",
  statistic = "std.error",
  statistic_override = NULL,
  statistic_vertical = TRUE,
  conf_level = 0.95,
  stars = FALSE,
  coef_map = NULL,
  coef_omit = NULL,
  gof_map = NULL,
  gof_omit = NULL,
  add_rows = NULL,
  title = NULL,
  notes = NULL,
  estimate = "estimate",
  ...
)
```

Arguments

`models` a single model object or a (potentially named) list of models to summarize

`output` filename or object type (string)

- Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
- Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable".
- When a file name is supplied to the ‘output‘ argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the ‘gt‘ or ‘kableExtra‘ packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the ‘gt::gtsave‘, ‘kable::save_kable‘, or ‘cat‘ functions.

`fmt` string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. ’%.3f’ will keep 3 digits after the decimal point with trailing zero. ’%.5f’ will keep 5 digits. ’%.3e’ will use exponential notation. See ‘?sprintf‘ for more options.

`statistic` string name of the statistic to include in parentheses

- Typical values: "conf.int", "std.error", "statistic", "p.value"
- Alternative values: any column name produced by ‘broom::tidy(model)‘

`statistic_override`

manually override the uncertainy estimates. This argument accepts three types of input:

- a function or list of functions of length(models) which produce variance-covariance matrices with row and column names equal to the names of your coefficient estimates. For example, ‘R‘ supplies the ‘vcov‘ function, and the ‘sandwich‘ package supplies ‘vcovHC‘, ‘vcovHAC‘, etc.

- a list of length(models) variance-covariance matrices with row and column names equal to the names of your coefficient estimates.
- a list of length(models) vectors with names equal to the names of your coefficient estimates. Numeric vectors are formatted according to ‘fmt‘ and placed in brackets, character vectors printed as given.

`statistic_vertical`	TRUE if statistics should be printed below estimates. FALSE if statistics should be printed beside estimates.
`conf_level`	confidence level to use for confidence intervals
`stars`	to indicate statistical significance • FALSE (default): no significance stars. • TRUE: *=.1, **=.05, ***=.01 • Named numeric vector for custom stars such as ‘c(’*’ = .1, ’+’ = .05)‘
`coef_map`	named character vector. Values refer to the variable names that will appear in the table. Names refer to the original term names stored in the model object (e.g., "hp:mpg" for an interaction term). Coefficients that are omitted from this vector will be omitted from the table. The table will be ordered in the same order as this vector.
`coef_omit`	string regular expression. Omits all matching coefficients from the table (using ‘grepl‘).
`gof_map`	data.frame with four columns: ‘raw‘, ‘clean‘, ‘fmt‘, and ‘omit‘. If ‘gof_map‘ is NULL, then ‘modelsummary‘ will use this data frame by default: ‘modelsummary::gof_map‘
`gof_omit`	string regular expression. Omits all matching gof statistics from the table (using ‘grepl‘).
`add_rows`	a data.frame (or tibble) with the same number of columns as your main table. By default, rows are appended to the bottom of the table. You can define a "position" attribute of integers to set the row positions. See examples.
`title`	string
`notes`	list or vector of notes to append to the bottom of the table.
`estimate`	character name of the estimate to display. Must be a column name in the data.frame produced by ‘tidy(model)‘. In the vast majority of cases, the default value of this argument should not be changed.
`...`	all other arguments are passed to the ‘tidy‘ method used to extract estimates from the model. For example, this allows users to set ‘exponentiate=TRUE‘ to exponentiate logistic regression coefficients.

Value

a ’gt’ table object.

Examples

```
## Not run:

library(modelsummary)
```

```
# load data and estimate models
data(trees)
models <- list()
models[['Bivariate']] <- lm(Girth ~ Height, data = trees)
models[['Multivariate']] <- lm(Girth ~ Height + Volume, data = trees)

# simple table
msummary(models)

# confidence intervals, p values, or t-stats instead of standard errors
msummary(models, statistic = 'conf.int', conf_level = 0.99)
msummary(models, statistic = 'p.value', conf_level = 0.99)
msummary(models, statistic = 'statistic', conf_level = 0.99)

# rename and re-order coefficients
msummary(models, coef_map = c('Volume' = 'Large', 'Height' = 'Tall'))

# titles
msummary(models, title = 'This is the title')

# title with italicized text
msummary(models, title = gt::md('This is *the* title'))

# add_rows: we use `tribble` from the `tibble` package to build a data.frame
# more easily. Then, we assign an attribute to determine each row's position.
rows <- tibble::tribble(~term, ~Bivariate, ~Multivariate,
                        'Empty row', '-', '-',
                        'Another empty row', '?', '?')
attr(rows, 'position') <- c(1, 3)
msummary(models, add_rows = rows)

# notes at the bottom of the table (here, the second note includes markdown bold characters)
msummary(models, notes = list('A first note', gt::md('A **bold** note')))

# modify list of GOF statistics and their format using the built-in
# 'gof_map' data frame as a starting point
gof_custom <- modelsummary::gof_map
gof_custom$omit[gof_custom$raw == 'deviance'] <- FALSE
gof_custom$fmt[gof_custom$raw == 'r.squared'] <- "%.5f"
msummary(models, gof_map = gof_custom)


## End(Not run)
```

`msummary` *Beautiful, customizable summaries of statistical models*

Description

‘msummary()‘ is a shortcut to ‘modelsummary()‘

Usage

```
msummary(
  models,
  output = "default",
  fmt = "%.3f",
  statistic = "std.error",
  statistic_override = NULL,
  statistic_vertical = TRUE,
  conf_level = 0.95,
  stars = FALSE,
  coef_map = NULL,
  coef_omit = NULL,
  gof_map = NULL,
  gof_omit = NULL,
  add_rows = NULL,
  title = NULL,
  notes = NULL,
  estimate = "estimate",
  ...
)
```

Arguments

`models`
: a single model object or a (potentially named) list of models to summarize

`output`
: filename or object type (string)

- Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
- Supported object types: "default", "html", "markdown", "latex", "data.frame", "gt", "kableExtra", "huxtable", "flextable".
- When a file name is supplied to the ‘output‘ argument, the table is written immediately to file. If you want to customize your table by post-processing it with functions provided by the ‘gt‘ or ‘kableExtra‘ packages, you need to choose a different output format (e.g., "gt", "latex", "html", "markdown"), and you need to save the table after post-processing using the ‘gt::gtsave‘, ‘kable::save_kable‘, or ‘cat‘ functions.

`fmt`
: string which specifies how numeric values will be rounded. This string is passed to the ‘sprintf‘ function. '%.3f' will keep 3 digits after the decimal point with trailing zero. '%.5f' will keep 5 digits. '%.3e' will use exponential notation. See ‘?sprintf‘ for more options.

`statistic`
: string name of the statistic to include in parentheses

- Typical values: "conf.int", "std.error", "statistic", "p.value"
- Alternative values: any column name produced by ‘broom::tidy(model)‘

`statistic_override`
: manually override the uncertainy estimates. This argument accepts three types of input:

- a function or list of functions of length(models) which produce variance-covariance matrices with row and column names equal to the names of your coefficient estimates. For example, ‘R‘ supplies the ‘vcov‘ function, and the ‘sandwich‘ package supplies ‘vcovHC‘, ‘vcovHAC‘, etc.
- a list of length(models) variance-covariance matrices with row and column names equal to the names of your coefficient estimates.

	• a list of length(models) vectors with names equal to the names of your coefficient estimates. Numeric vectors are formatted according to ‘fmt‘ and placed in brackets, character vectors printed as given.
`statistic_vertical`	TRUE if statistics should be printed below estimates. FALSE if statistics should be printed beside estimates.
`conf_level`	confidence level to use for confidence intervals
`stars`	to indicate statistical significance • FALSE (default): no significance stars. • TRUE: *=.1, **=.05, ***=.01 • Named numeric vector for custom stars such as ‘c('*' = .1, '+' = .05)‘
`coef_map`	named character vector. Values refer to the variable names that will appear in the table. Names refer to the original term names stored in the model object (e.g., "hp:mpg" for an interaction term). Coefficients that are omitted from this vector will be omitted from the table. The table will be ordered in the same order as this vector.
`coef_omit`	string regular expression. Omits all matching coefficients from the table (using ‘grepl‘).
`gof_map`	data.frame with four columns: ‘raw‘, ‘clean‘, ‘fmt‘, and ‘omit‘. If ‘gof_map‘ is NULL, then ‘modelsummary‘ will use this data frame by default: ‘modelsummary::gof_map‘
`gof_omit`	string regular expression. Omits all matching gof statistics from the table (using ‘grepl‘).
`add_rows`	a data.frame (or tibble) with the same number of columns as your main table. By default, rows are appended to the bottom of the table. You can define a "position" attribute of integers to set the row positions. See examples.
`title`	string
`notes`	list or vector of notes to append to the bottom of the table.
`estimate`	character name of the estimate to display. Must be a column name in the data.frame produced by ‘tidy(model)‘. In the vast majority of cases, the default value of this argument should not be changed.
`...`	all other arguments are passed to the ‘tidy‘ method used to extract estimates from the model. For example, this allows users to set ‘exponentiate=TRUE‘ to exponentiate logistic regression coefficients.

Value

a 'gt' table object.

Examples

```
## Not run:

library(modelsummary)

# load data and estimate models
data(trees)
models <- list()
models[['Bivariate']] <- lm(Girth ~ Height, data = trees)
```

```
models[['Multivariate']] <- lm(Girth ~ Height + Volume, data = trees)

# simple table
msummary(models)

# confidence intervals, p values, or t-stats instead of standard errors
msummary(models, statistic = 'conf.int', conf_level = 0.99)
msummary(models, statistic = 'p.value', conf_level = 0.99)
msummary(models, statistic = 'statistic', conf_level = 0.99)

# rename and re-order coefficients
msummary(models, coef_map = c('Volume' = 'Large', 'Height' = 'Tall'))

# titles
msummary(models, title = 'This is the title')

# title with italicized text
msummary(models, title = gt::md('This is *the* title'))

# add_rows: we use `tribble` from the `tibble` package to build a data.frame
# more easily. Then, we assign an attribute to determine each row's position.
rows <- tibble::tribble(~term, ~Bivariate, ~Multivariate,
                        'Empty row', '-', '-',
                        'Another empty row', '?', '?')
attr(rows, 'position') <- c(1, 3)
msummary(models, add_rows = rows)

# notes at the bottom of the table (here, the second note includes markdown bold characters)
msummary(models, notes = list('A first note', gt::md('A **bold** note')))

# modify list of GOF statistics and their format using the built-in
# 'gof_map' data frame as a starting point
gof_custom <- modelsummary::gof_map
gof_custom$omit[gof_custom$raw == 'deviance'] <- FALSE
gof_custom$fmt[gof_custom$raw == 'r.squared'] <- "%.5f"
msummary(models, gof_map = gof_custom)


## End(Not run)
```

`Multicolumn` *Use a variable as a factor to give rows in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Multicolumn`

`Paste` *Generate terms to paste values together in table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Paste`

`Percent` *Pseudo-function to compute a statistic relative to a reference set.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `Percent`

`PlusMinus` *Generate 'x +/- y' terms in table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `PlusMinus`

`RowFactor` *Use a variable as a factor to give rows in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `RowFactor`

`RowNum` *Display all observations in a table.*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

tables `RowNum`

`sanity_ds_right_handed_formula` *sanity check: datasummary_table1*

Description

sanity check: datasummary_table1

Usage

```
sanity_ds_right_handed_formula(formula)
```

Arguments

`formula`	right-handed formulae only

`tidy_custom.default` *Extract custom information from a model object and turn it into a tidy tibble*

Description

Extract custom information from a model object and turn it into a tidy tibble

Usage

```
## Default S3 method:
tidy_custom(x)
```

Arguments

`x`	An object to be converted into a tidy [tibble::tibble()].

Value

A [tibble::tibble()] with information about model components.

Index