# Package 'modelsummary'

June 5, 2022

**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** 0.11.0

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

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

**Depends** R (>= 3.5.0)

**Imports** broom,
checkmate,
data.table,
generics,
glue,
insight (>= 0.16.1),
kableExtra (>= 1.2.1),
parameters (>= 0.17.0),
performance,
tables

**Suggests** betareg,
bookdown,
broom.mixed,
covr,
digest,
DT,
estimatr,
fixest (> 0.10.3),
flextable,

future,
future.apply,
gamlss,
ggplot2,
gt (>= 0.3.0),
huxtable,
IRdisplay,
knitr,
lfe,
lme4,
lmtest,
magick,
officer,
rmarkdown,
sandwich,
spelling,
survey,
testthat,
tibble,
MASS,
mice,
nnet,
pscl,
vdiffr

**License** GPL-3

**Encoding** UTF-8

**Config/testthat/edition** 3

**LazyData** false

**Roxygen** list(markdown = TRUE)

**Language** en-US

**RoxygenNote** 7.2.0

**Collate** 'bind_est_gof.R'
'coef_rename.R'
'convenience.R'
'datasummary.R'
'datasummary_balance.R'
'datasummary_correlation.R'
'datasummary_crosstab.R'
'datasummary_df.R'
'datasummary_extract.R'
'datasummary_functions.R'
'datasummary_skim.R'
'dvnames.R'
'escape.R'
'factory.R'
'factory_dataframe.R'

'factory_flextable.R'
'factory_gt.R'
'factory_huxtable.R'
'factory_kableExtra.R'
'format_estimates.R'
'format_gof.R'
'get_estimates.R'
'get_gof.R'
'get_vcov.R'
'glance_custom.R'
'gof_map.R'
'map_estimates.R'
'map_gof.R'
'methods_did.R'
'methods_estimatr.R'
'methods_fixest.R'
'methods_lfe.R'
'methods_stats.R'
'modelplot.R'
'modelsummary.R'
'modelsummary_list.R'
'modelsummary_wide.R'
'poorman.R'
'reexport.R'
'rename_statistics.R'
'sanitize_fmt.R'
'sanitize_gof_map.R'
'sanitize_models.R'
'sanitize_output.R'
'sanitize_shape.R'
'sanitize_vcov.R'
'sanity_checks.R'
'settings.R'
'shape_estimates.R'
'span.R'
'stars.R'
'supported_models.R'
'themes.R'
'tidy_custom.R'
'utils_pad.R'
'utils_print.R'
'utils_replace.R'
'utils_rounding.R'
'utils_stats.R'
'utils_warn.R'

# R topics documented:



---

`coef_rename` *Rename model terms*

---

## Description

A convenience function which can be passed to the `coef_rename` argument of the `modelsummary` function.

### Usage

```
coef_rename(
  x,
  factor = TRUE,
  factor_name = TRUE,
  backticks = TRUE,
  titlecase = TRUE,
  underscore = TRUE,
  asis = TRUE
)
```

### Arguments

| | |
|---|---|
| x | character vector of term names to transform |
| factor | boolean remove the "factor()" label |
| factor_name | boolean remove the "factor()" label and the name of the variable |
| backticks | boolean remove backticks |
| titlecase | boolean convert to title case |
| underscore | boolean replace underscores by spaces |
| asis | boolean remove the I from as-is formula calls |

### Examples

```
## Not run:
library(modelsummary)
dat <- mtcars
dat$horse_power <- dat$hp
mod <- lm(mpg ~ horse_power + factor(cyl), dat)
modelsummary(mod, coef_rename = coef_rename)

## End(Not run)
```

---

## datasummary *Summary tables using 2-sided formulae: crosstabs, frequencies, table 1s and more.*

---

### Description

datasummary can use any summary function which produces one numeric or character value per variable. The examples section of this documentation shows how to define custom summary functions. The package also ships with several shortcut summary functions: Min, Max, Mean, Median, Var, SD, NPercent, NUnique, Ncol, P0, P25, P50, P75, P100. See the Details and Examples sections below, and the vignettes on the modelsummary website: https://vincentarelbundock.github.io/modelsummary/
* datasummary Vignette.

## Usage

```
datasummary(
  formula,
  data,
  output = "default",
  fmt = 2,
  title = NULL,
  notes = NULL,
  align = NULL,
  add_columns = NULL,
  add_rows = NULL,
  sparse_header = TRUE,
  escape = 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 (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.<br>• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.<br>• Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.<br>• LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.<br>• The default output formats and table-making packages can be modified with global options. See the 'Details' section. |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`<br>• character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`<br>• function: returns a formatted character string.<br>• NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments. |
| title | string |
| notes | list or vector of notes to append to the bottom of the table. |

`align` A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.

- "l": left-aligned column
- "c": centered column
- "r": right-aligned column
- "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF):
  - `\usepackage{booktabs}`
  - `\usepackage{siunitx}`
  - `\newcolumntype{d}{S[input-symbols = ()]}`

`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 section below.

`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.‘

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

`...` all other arguments are passed through to the table-making functions. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes, for instance:

- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.

## 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.

## Global Options

The behavior of `modelsummary` can be affected by setting global options:

- `modelsummary_factory_default`

- modelsummary_factory_latex
- modelsummary_factory_html
- modelsummary_factory_png
- modelsummary_get
- modelsummary_format_numeric_latex
- modelsummary_format_numeric_html

**Table-making packages:**

modelsummary supports 4 table-making packages: kableExtra, gt, flextable, and huxtable. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the options function:

```
options(modelsummary_factory_html = 'kableExtra') options(modelsummary_factory_latex
= 'gt') options(modelsummary_factory_word = 'huxtable') options(modelsummary_factory_png
= 'gt')
```

**Model extraction functions:**

modelsummary can use two sets of packages to extract information from statistical models: the easystats family (performance and parameters) and broom. By default, it uses easystats first and then falls back on broom in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

```
options(modelsummary_get = "broom") options(modelsummary_get = "easystats") options(modelsummary_get
= "all")
```

**Formatting numeric entries:**

By default, LaTeX tables enclose all numeric entries in the \num{} command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

```
options(modelsummary_format_numeric_latex = "plain") options(modelsummary_format_numeric_latex
= "mathmode")
```

A similar option can be used to display numerical entries using MathJax in HTML tables:

```
options(modelsummary_format_numeric_html = "mathjax")
```

## 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_balance` *Balance table: Summary statistics for different subsets of the data (e.g., control and treatment groups)*

---

## Description

Creates balance tables with summary statistics for different subsets of the data (e.g., control and treatment groups). It can also be used to create summary tables for full data sets. See the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://vincentarelbundock.github.io/modelsumma
* datasummary Vignette.

## Usage

```
datasummary_balance(
  formula,
  data,
  output = "default",
  fmt = 1,
  title = NULL,
  notes = NULL,
  align = NULL,
  add_columns = NULL,
  add_rows = NULL,
```

```
  dinm = TRUE,
  dinm_statistic = "std.error",
  escape = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| formula | a one-sided formula with the "condition" or "column" variable on the right-hand side. ~1 can be used to show summary statistics for the full data set |
| data | A data.frame (or tibble). If this data includes columns called "blocks", "clusters", and/or "weights", the "estimatr" package will consider them when calculating the difference in means. If there is a `weights` column, the reported mean and standard errors will also be weighted. |
| output | filename or object type (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.<br>• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.<br>• Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.<br>• LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.<br>• The default output formats and table-making packages can be modified with global options. See the 'Details' section. |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`<br>• character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`<br>• function: returns a formatted character string.<br>• NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments. |
| title | string |
| notes | list or vector of notes to append to the bottom of the table. |
| align | A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.<br>• "l": left-aligned column<br>• "c": centered column<br>• "r": right-aligned column<br>• "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF): |

- `\usepackage{booktabs}`
- `\usepackage{siunitx}`
- `\newcolumntype{d}{S[input-symbols = ()]}`

`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 section below.

`dinm` TRUE calculates a difference in means with uncertainty estimates. This option is only available if the `estimatr` package is installed. If `data` includes columns named "blocks", "clusters", or "weights", this information will be taken into account automatically by `estimatr::difference_in_means`.

`dinm_statistic` string: "std.error" or "p.value"

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

`...` all other arguments are passed through to the table-making functions. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes, for instance:

- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.

## Global Options

The behavior of `modelsummary` can be affected by setting global options:

- `modelsummary_factory_default`
- `modelsummary_factory_latex`
- `modelsummary_factory_html`
- `modelsummary_factory_png`
- `modelsummary_get`
- `modelsummary_format_numeric_latex`
- `modelsummary_format_numeric_html`

### Table-making packages:

`modelsummary` supports 4 table-making packages: `kableExtra`, `gt`, `flextable`, and `huxtable`. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the `options` function:

`options(modelsummary_factory_html = 'kableExtra')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')`

**Model extraction functions:**

`modelsummary` can use two sets of packages to extract information from statistical models: the `easystats` family (`performance` and `parameters`) and `broom`. By default, it uses `easystats` first and then falls back on `broom` in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "broom")` `options(modelsummary_get = "easystats")` `options(modelsummary_get = "all")`

**Formatting numeric entries:**

By default, LaTeX tables enclose all numeric entries in the `\num{}` command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

`options(modelsummary_format_numeric_latex = "plain")` `options(modelsummary_format_numeric_latex = "mathmode")`

A similar option can be used to display numerical entries using MathJax in HTML tables:

`options(modelsummary_format_numeric_html = "mathjax")`

## Examples

```
## Not run:
datasummary_balance(~am, mtcars)

## End(Not run)
```

---

# datasummary_correlation

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

---

## Description

The names of the variables displayed in the correlation table are the names of the columns in the `data`. You can rename those columns (with or without spaces) to produce a table of human-readable variables. See the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://vincentarelbundock.github.io/modelsummary/ * datasummary Vignette.

## Usage

```
datasummary_correlation(
  data,
  output = "default",
  method = "pearson",
  fmt = 2,
  align = NULL,
  add_rows = NULL,
  add_columns = NULL,
  title = NULL,
  notes = NULL,
```

```
  escape = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| data | A data.frame (or tibble) |
| output | filename or object type (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.<br>• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the modelsummary function.<br>• Warning: Users should not supply a file name to the output argument if they intend to customize the table with external packages. See the 'Details' section.<br>• LaTeX compilation requires the booktabs and siunitx packages, but siunitx can be disabled or replaced with global options. See the 'Details' section.<br>• The default output formats and table-making packages can be modified with global options. See the 'Details' section. |
| method | character or function<br>• character: "pearson", "kendall", "spearman", or "pearspear" (Pearson correlations above and Spearman correlations below the diagonal)<br>• function: takes a data.frame with numeric columns and returns a square matrix or data.frame with unique row.names and colnames corresponding to variable names. Note that the datasummary_correlation_format can often be useful for formatting the output of custom correlation functions. |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period format(round(x, fmt), nsmall=fmt)<br>• character: passed to the sprintf function (e.g., '%.3f' keeps 3 digits with trailing zero). See ?sprintf<br>• function: returns a formatted character string.<br>• NULL: does not format numbers, which allows users to include function in the "glue" strings in the estimate and statistic arguments. |
| align | A string with a number of characters equal to the number of columns in the table (e.g., align = "lcc"). Valid characters: l, c, r, d.<br>• "l": left-aligned column<br>• "c": centered column<br>• "r": right-aligned column<br>• "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by kableExtra. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF): |

- \usepackage{booktabs}
- \usepackage{siunitx}
- \newcolumntype{d}{S[input-symbols = ()]}

| | |
|---|---|
| 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 section below. |
| add_columns | a data.frame (or tibble) with the same number of rows as your main table. |
| title | string |
| notes | list or vector of notes to append to the bottom of the table. |
| escape | boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes. |
| ... | other parameters are passed through to the table-making packages. |

## Global Options

The behavior of modelsummary can be affected by setting global options:

- modelsummary_factory_default
- modelsummary_factory_latex
- modelsummary_factory_html
- modelsummary_factory_png
- modelsummary_get
- modelsummary_format_numeric_latex
- modelsummary_format_numeric_html

### Table-making packages:

modelsummary supports 4 table-making packages: kableExtra, gt, flextable, and huxtable. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the options function:

```
options(modelsummary_factory_html = 'kableExtra') options(modelsummary_factory_latex
= 'gt') options(modelsummary_factory_word = 'huxtable') options(modelsummary_factory_png
= 'gt')
```

### Model extraction functions:

modelsummary can use two sets of packages to extract information from statistical models: the easystats family (performance and parameters) and broom. By default, it uses easystats first and then falls back on broom in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

```
options(modelsummary_get = "broom") options(modelsummary_get = "easystats") options(modelsummary_get
= "all")
```

**Formatting numeric entries:**

By default, LaTeX tables enclose all numeric entries in the `\num{}` command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

`options(modelsummary_format_numeric_latex = "plain")` `options(modelsummary_format_numeric_latex = "mathmode")`

A similar option can be used to display numerical entries using MathJax in HTML tables:

`options(modelsummary_format_numeric_html = "mathjax")`

## Examples

```
## Not run:
library(modelsummary)

# clean variable names (base R)
dat <- mtcars[, c("mpg", "hp")]
colnames(dat) <- c("Miles / Gallon", "Horse Power")
datasummary_correlation(dat)

# clean variable names (tidyverse)
library(tidyverse)
dat <- mtcars %>%
  select(`Miles / Gallon` = mpg,
         `Horse Power` = hp)
datasummary_correlation(dat)

# alternative methods
datasummary_correlation(dat, method = "pearspear")

# custom function
cor_fun <- function(x) cor(x, method = "kendall")
datasummary_correlation(dat, method = cor_fun)

# rename columns alphabetically and include a footnote for reference
note <- sprintf("(%s) %s", letters[1:ncol(dat)], colnames(dat))
note <- paste(note, collapse = "; ")

colnames(dat) <- sprintf("(%s)", letters[1:ncol(dat)])

datasummary_correlation(dat, notes = note)

# `datasummary_correlation_format`: custom function with formatting
dat <- mtcars[, c("mpg", "hp", "disp")]

cor_fun <- function(x) {
  out <- cor(x, method = "kendall")
  datasummary_correlation_format(
    out,
    fmt = 2,
    upper_triangle = "x",
    diagonal = ".")
}
```

```
datasummary_correlation(dat, method = cor_fun)

# use kableExtra and psych to color significant cells
library(psych)
library(kableExtra)

dat <- mtcars[, c("vs", "hp", "gear")]

cor_fun <- function(dat) {
  # compute correlations and format them
  correlations <- data.frame(cor(dat))
  correlations <- datasummary_correlation_format(correlations, fmt = 2)

  # calculate pvalues using the `psych` package
  pvalues <- psych::corr.test(dat)$p

  # use `kableExtra::cell_spec` to color significant cells
  for (i in 1:nrow(correlations)) {
    for (j in 1:ncol(correlations)) {
      if (pvalues[i, j] < 0.05 && i != j) {
        correlations[i, j] <- cell_spec(correlations[i, j], background = "pink")
      }
    }
  }
  return(correlations)
}

# The `escape=FALSE` is important here!
datasummary_correlation(dat, method = cor_fun, escape = FALSE)

## End(Not run)
```

---

`datasummary_correlation_format` *Format the content of a correlation table*

---

## Description

Mostly for internal use, but can be useful when users supply a function to the `method` argument of `datasummary_correlation`.

## Usage

```
datasummary_correlation_format(
  x,
  fmt,
  leading_zero = FALSE,
  diagonal = NULL,
  upper_triangle = NULL
)
```

### Arguments

| | |
|---|---|
| x | square numeric matrix |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`<br>• character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`<br>• function: returns a formatted character string.<br>• NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments. |
| leading_zero | boolean. If `FALSE`, leading zeros are removed |
| diagonal | character or NULL. If character, all elements of the diagonal are replaced by the same character (e.g., "1"). |
| upper_triangle | character or NULL. If character, all elements of the upper triangle are replaced by the same character (e.g., "" or "."). |

### Examples

```
library(modelsummary)

dat <- mtcars[, c("mpg", "hp", "disp")]

cor_fun <- function(x) {
  out <- cor(x, method = "kendall")
  datasummary_correlation_format(
    out,
    fmt = 2,
    upper_triangle = "x",
    diagonal = ".")
}

datasummary_correlation(dat, method = cor_fun)
```

---

## datasummary_crosstab *Cross tabulations for categorical variables*

---

### Description

Convenience function to tabulate counts, cell percentages, and row/column percentages for categorical variables. See the Details section for a description of the internal design. For more complex cross tabulations, use datasummary directly. See the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://vincentarelbundock.github.io/modelsummary/
* datasummary Vignette.

## Usage

```
datasummary_crosstab(
  formula,
  statistic = 1 ~ 1 + N + Percent("row"),
  data,
  output = "default",
  fmt = 1,
  title = NULL,
  notes = NULL,
  align = NULL,
  add_columns = NULL,
  add_rows = NULL,
  sparse_header = TRUE,
  escape = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| formula | A two-sided formula to describe the table: rows ~ columns, where rows and columns are variables in the data. Rows and columns may contain interactions, e.g., `var1 * var2 ~ var3`. |
| statistic | A formula of the form `1 ~ 1 + N + Percent("row")`. The left-hand side may only be empty or contain a `1` to include row totals. The right-hand side may contain: `1` for column totals, `N` for counts, `Percent()` for cell percentages, `Percent("row")` for row percentages, `Percent("col")` for column percentages. |
| data | A data.frame (or tibble) |
| output | filename or object type (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.<br>• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.<br>• Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.<br>• LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.<br>• The default output formats and table-making packages can be modified with global options. See the 'Details' section. |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`<br>• character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf` |

- function: returns a formatted character string.
- NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments.

`title` string

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

`align` A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.

- "l": left-aligned column
- "c": centered column
- "r": right-aligned column
- "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF):
  - `\usepackage{booktabs}`
  - `\usepackage{siunitx}`
  - `\newcolumntype{d}{S[input-symbols = ()]}`

`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 section below.

`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.‘

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

`...` all other arguments are passed through to the table-making functions. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes, for instance:

- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.

## Details

`datasummary_crosstab` is a wrapper around the datasummary function. This wrapper works by creating a customized formula and by feeding it to `datasummary`. The customized formula comes in two parts.

First, we take a two-sided formula supplied by the `formula` argument. All variables of that formula are wrapped in a `Factor()` call to ensure that the variables are treated as categorical.

Second, the `statistic` argument gives a two-sided formula which specifies the statistics to include in the table. `datasummary_crosstab` modifies this formula automatically to include "clean" labels.

Finally, the `formula` and `statistic` formulas are combined into a single formula which is fed directly to the `datasummary` function to produce the table.

Variables in `formula` are automatically wrapped in `Factor()`.

## Global Options

The behavior of `modelsummary` can be affected by setting global options:

- `modelsummary_factory_default`
- `modelsummary_factory_latex`
- `modelsummary_factory_html`
- `modelsummary_factory_png`
- `modelsummary_get`
- `modelsummary_format_numeric_latex`
- `modelsummary_format_numeric_html`

### Table-making packages:

`modelsummary` supports 4 table-making packages: `kableExtra`, `gt`, `flextable`, and `huxtable`. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the `options` function:

```
options(modelsummary_factory_html = 'kableExtra') options(modelsummary_factory_latex
= 'gt') options(modelsummary_factory_word = 'huxtable') options(modelsummary_factory_png
= 'gt')
```

### Model extraction functions:

`modelsummary` can use two sets of packages to extract information from statistical models: the `easystats` family (`performance` and `parameters`) and `broom`. By default, it uses `easystats` first and then falls back on `broom` in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

```
options(modelsummary_get = "broom") options(modelsummary_get = "easystats") options(modelsummary_get
= "all")
```

### Formatting numeric entries:

By default, LaTeX tables enclose all numeric entries in the `\num{}` command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

```
options(modelsummary_format_numeric_latex = "plain") options(modelsummary_format_numeric_latex
= "mathmode")
```

A similar option can be used to display numerical entries using MathJax in HTML tables:

```
options(modelsummary_format_numeric_html = "mathjax")
```

## Examples

```
## Not run:
  # crosstab of two variables, showing counts, row percentages, and row/column totals
  datasummary_crosstab(cyl ~ gear, data = mtcars)

  # crosstab of two variables, showing counts only and no totals
  datasummary_crosstab(cyl ~ gear, statistic = ~ N, data = mtcars)

  # crosstab of three variables
  datasummary_crosstab(am * cyl ~ gear, data = mtcars)

  # crosstab with two variables and column percentages
  datasummary_crosstab(am ~ gear, statistic = ~ Percentage("col"), data = mtcars)

## End(Not run)
```

---

`datasummary_df` *Draw a table from a data.frame*

---

## Description

Draw a table from a data.frame

## Usage

```
datasummary_df(
  data,
  output = "default",
  fmt = 2,
  align = NULL,
  hrule = NULL,
  title = NULL,
  notes = NULL,
  add_rows = NULL,
  add_columns = NULL,
  escape = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame (or tibble) |
| `output` | filename or object type (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg. |

- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.
- LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.
- The default output formats and table-making packages can be modified with global options. See the 'Details' section.

`fmt` determines how to format numeric values

- integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`
- character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`
- function: returns a formatted character string.
- NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments.

`align` A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.

- "l": left-aligned column
- "c": centered column
- "r": right-aligned column
- "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF):
  - `\usepackage{booktabs}`
  - `\usepackage{siunitx}`
  - `\newcolumntype{d}{S[input-symbols = ()]}`

`hrule` position of horizontal rules (integer vector)

`title` string

`notes` list or vector of notes to append to the bottom of the 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 section below.

`add_columns` a data.frame (or tibble) with the same number of rows as your main table.

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

`...` all other arguments are passed through to the table-making functions. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes, for instance:

- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.

---

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

---

## Description

This function was inspired by the excellent `skimr` package for R. See the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://vincentarelbundock.github.io/modelsummary/ * datasummary Vignette.

## Usage

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

## Arguments

`data` A data.frame (or tibble)

`type` of variables to summarize: "numeric" or "categorical" (character)

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.
- LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.

- The default output formats and table-making packages can be modified with global options. See the 'Details' section.

`fmt` determines how to format numeric values

- integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`
- character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`
- function: returns a formatted character string.
- NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments.

`histogram` include a histogram (TRUE/FALSE). Supported for:

- type = "numeric"
- output is "html", "default", "jpg", "png", or "kableExtra"
- PDF and HTML documents compiled via Rmarkdown or knitr
- See the examples section below for an example of how to use `datasummary` to include histograms in other formats such as markdown.

`title` string

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

`align` A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.

- "l": left-aligned column
- "c": centered column
- "r": right-aligned column
- "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF):
  - `\usepackage{booktabs}`
  - `\usepackage{siunitx}`
  - `\newcolumntype{d}{S[input-symbols = ()]}`

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

`...` all other arguments are passed through to the table-making functions. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes, for instance:

- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.

## Global Options

The behavior of `modelsummary` can be affected by setting global options:

- `modelsummary_factory_default`

- modelsummary_factory_latex
- modelsummary_factory_html
- modelsummary_factory_png
- modelsummary_get
- modelsummary_format_numeric_latex
- modelsummary_format_numeric_html

**Table-making packages:**

modelsummary supports 4 table-making packages: kableExtra, gt, flextable, and huxtable. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the options function:

options(modelsummary_factory_html = 'kableExtra') options(modelsummary_factory_latex = 'gt') options(modelsummary_factory_word = 'huxtable') options(modelsummary_factory_png = 'gt')

**Model extraction functions:**

modelsummary can use two sets of packages to extract information from statistical models: the easystats family (performance and parameters) and broom. By default, it uses easystats first and then falls back on broom in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

options(modelsummary_get = "broom") options(modelsummary_get = "easystats") options(modelsummary_get = "all")

**Formatting numeric entries:**

By default, LaTeX tables enclose all numeric entries in the \num{} command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

options(modelsummary_format_numeric_latex = "plain") options(modelsummary_format_numeric_latex = "mathmode")

A similar option can be used to display numerical entries using MathJax in HTML tables:

options(modelsummary_format_numeric_html = "mathjax")

## Examples

```
## Not run:
dat <- mtcars
dat$vs <- as.logical(dat$vs)
dat$cyl <- as.factor(dat$cyl)
datasummary_skim(dat)
datasummary_skim(dat, "categorical")

# You can use `datasummary` to produce a similar table in different formats.
# Note that the `Histogram` function relies on unicode characters. These
# characters will only display correctly in some operating systems, under some
# locales, using some fonts. Displaying such histograms on Windows computers
# is notoriously tricky. The `modelsummary` authors cannot provide support to
# display these unicode histograms.
```

```
f <- All(mtcars) ~ Mean + SD + Min + Median + Max + Histogram
datasummary(f, mtcars, output="markdown")


## End(Not run)
```

---

dvnames *Title models with their dependent variables*

---

## Description

A convenience function for use with a regression model or list of regression models. Returns a named list of models, where the names are the models' respective dependent variables. Pass your list of models to `dvnames` before sending to `modelsummary` to automatically get dependent variable-titled columns.

## Usage

```
dvnames(models, number = FALSE, fill = "Model")
```

## Arguments

| | |
|---|---|
| `models` | A regression model or list of regression models |
| `number` | Should the models be numbered (1), (2), etc., in addition to their dependent variable names? |
| `fill` | If `insight::find_response()` cannot find a response, the column title to use in its place. Set to `' '` to leave blank. |

## Examples

```
m1 <- lm(mpg ~ hp, data = mtcars)
m2 <- lm(mpg ~ hp + wt, data = mtcars)

# Without dvnames, column names are Model 1 and Model 2
modelsummary(list(m1, m2))

# With dvnames, they are "mpg" and "mpg"
modelsummary(dvnames(list(m1,m2)))
```

`get_estimates` *Extract model estimates in a tidy format.*

## Description

This is a mostly internal function which could be useful to users who want a unified approach to extract results from a wide variety of models. For some models `get_estimates` attaches useful attributes to the output. You can access this information by calling the `attributes` function: `attributes(get_estimates(model))`

## Usage

```
get_estimates(model, conf_level = 0.95, vcov = NULL, ...)
```

## Arguments

| | |
|---|---|
| `model` | a single model object |
| `conf_level` | confidence level to use for confidence intervals |
| `vcov` | robust standard errors and other manual statistics. The `vcov` argument accepts six types of input (see the 'Details' and 'Examples' sections below): |

- NULL returns the default uncertainty estimates of the model object
- string, vector, or (named) list of strings. "iid", "classical", and "constant" are aliases for `NULL`, which returns the model's default uncertainty estimates. The strings "HC", "HC0", "HC1" (alias: "stata"), "HC2", "HC3" (alias: "robust"), "HC4", "HC4m", "HC5", "HAC", "NeweyWest", "Andrews", "panel-corrected", "outer-product", and "weave" use variance-covariance matrices computed using functions from the `sandwich` package, or equivalent method. The behavior of those functions can (and sometimes *must*) be altered by passing arguments to `sandwich` directly from `modelsummary` through the ellipsis (`...`), but it is safer to define your own custom functions as described in the next bullet.
- function or (named) list of functions which return variance-covariance matrices with row and column names equal to the names of your coefficient estimates (e.g., `stats::vcov`, `sandwich::vcovHC`, `function(x) vcovPC(x, cluster="country")`).
- formula or (named) list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).
- named list of `length(models)` variance-covariance matrices with row and column names equal to the names of your coefficient estimates.
- a named list of length(models) vectors with names equal to the names of your coefficient estimates. See 'Examples' section below. Warning: since this list of vectors can include arbitrary strings or numbers, `modelsummary` cannot automatically calculate p values. The `stars` argument may thus use incorrect significance thresholds when `vcov` is a list of vectors.

`...` all other arguments are passed through to the extractor and table-making functions (by default `broom::tidy` and `kableExtra::kbl`, but this can be customized). This allows users to pass arguments directly to `modelsummary` in order to affect the behavior of other functions behind the scenes. For example,

- `metrics="none"`, `metrics="all"`, or `metrics=c("R2", "RMSE")` to select the goodness-of-fit extracted by the `performance` package (must have set `options(modelsummary_get="easystats")` first). This can be useful for some models when statistics take a long time to compute. See `?performance::performance`

---

`get_gof` *Extract model gof A mostly internal function with some potential uses outside.*

---

## Description

Extract model gof A mostly internal function with some potential uses outside.

## Usage

```
get_gof(model, vcov_type = NULL, ...)
```

## Arguments

| | |
|---|---|
| `model` | a single model object |
| `vcov_type` | string vcov type to add at the bottom of the table |
| `...` | all other arguments are passed through to the extractor and table-making functions (by default `broom::tidy` and `kableExtra::kbl`, but this can be customized). This allows users to pass arguments directly to `modelsummary` in order to affect the behavior of other functions behind the scenes. For example, |

- `metrics="none"`, `metrics="all"`, or `metrics=c("R2", "RMSE")` to select the goodness-of-fit extracted by the `performance` package (must have set `options(modelsummary_get="easystats")` first). This can be useful for some models when statistics take a long time to compute. See `?performance::performance`

## 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 'modelsummary' function. 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"
modelsummary(mod, gof_map = gm)

## End(Not run)
```

## Histogram *datasummary statistic shortcut*

### Description

This function uses Unicode characters to create a histogram. This can sometimes be useful, but is generally discouraged. Unicode characters can only display a limited number of heights for bars, and the accuracy of output is highly dependent on the platform (typeface, output type, windows vs. mac, etc.). We recommend you use the `kableExtra::spec_hist` function instead.

### Usage

```
Histogram(x, bins = 10)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| bins | number of histogram bars |

---

`Max` *datasummary statistic shortcut*

---

### Description

datasummary statistic shortcut

### Usage

```
Max(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the `modelsummary:::rounding` function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

### Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

---

`Mean` *datasummary statistic shortcut*

---

### Description

datasummary statistic shortcut

### Usage

```
Mean(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

### Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

---

## Median *datasummary statistic shortcut*

---

### Description

datasummary statistic shortcut

### Usage

```
Median(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

### Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

## Min *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

```
Min(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

### Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

## modelplot *Model Summary Plots with Estimates and Confidence Intervals*

### Description

Dot-Whisker plot of coefficient estimates with confidence intervals. For more information, see the Details and Examples sections below, and the vignettes on the modelsummary website: https://vincentarelbundock.github.io/m

- modelplot Vignette.

## Usage

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

## Arguments

| | |
|---|---|
| `models` | a model or (optionally named) list of models |
| `conf_level` | confidence level to use for confidence intervals |
| `coef_map` | character vector. Subset, rename, and reorder coefficients. Coefficients omitted from this vector are omitted from the table. The order of the vector determines the order of the table. `coef_map` can be a named or an unnamed character vector. If `coef_map` is a named vector, its values define the labels that must appear in the table, and its names identify the original term names stored in the model object: `c("hp:mpg"="HPxM/G")`. See Examples section below. |
| `coef_omit` | string regular expression (perl-compatible) used to determine which coefficients to omit from the table. A "negative lookahead" can be used to specify which coefficients to *keep* in the table. Examples:<br>• `"ei"`: omit coefficients matching the "ei" substring.<br>• `"^Volume$"`: omit the "Volume" coefficient.<br>• `"ei\|rc"`: omit coefficients matching either the "ei" or the "rc" substrings.<br>• `"^(?!Vol)"`: keep coefficients starting with "Vol" (inverse match using a negative lookahead).<br>• `"^(?!.*ei)"`: keep coefficients matching the "ei" substring.<br>• `"^(?!.*ei\|.*pt)"`: keep coefficients matching either the "ei" or the "pt" substrings.<br>• See the Examples section below for complete code. |
| `coef_rename` | named character vector or function<br>• 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. Ex: c("hp:mpg"="hp X mpg")<br>• Function: Accepts a character vector of the model's term names and returns a named vector like the one described above. The `modelsummary` package supplies a `coef_rename()` function which can do common cleaning tasks: `modelsummary(model, coef_rename = coef_rename)` |

vcov — robust standard errors and other manual statistics. The vcov argument accepts six types of input (see the 'Details' and 'Examples' sections below):

- NULL returns the default uncertainty estimates of the model object
- string, vector, or (named) list of strings. "iid", "classical", and "constant" are aliases for NULL, which returns the model's default uncertainty estimates. The strings "HC", "HC0", "HC1" (alias: "stata"), "HC2", "HC3" (alias: "robust"), "HC4", "HC4m", "HC5", "HAC", "NeweyWest", "Andrews", "panel-corrected", "outer-product", and "weave" use variance-covariance matrices computed using functions from the sandwich package, or equivalent method. The behavior of those functions can (and sometimes *must*) be altered by passing arguments to sandwich directly from modelsummary through the ellipsis (...), but it is safer to define your own custom functions as described in the next bullet.
- function or (named) list of functions which return variance-covariance matrices with row and column names equal to the names of your coefficient estimates (e.g., stats::vcov, sandwich::vcovHC, function(x) vcovPC(x, cluster="country")).
- formula or (named) list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).
- named list of length(models) variance-covariance matrices with row and column names equal to the names of your coefficient estimates.
- a named list of length(models) vectors with names equal to the names of your coefficient estimates. See 'Examples' section below. Warning: since this list of vectors can include arbitrary strings or numbers, modelsummary cannot automatically calculate p values. The stars argument may thus use incorrect significance thresholds when vcov is a list of vectors.

exponentiate — TRUE, FALSE, or logical vector of length equal to the number of models. If TRUE, the estimate, conf.low, and conf.high statistics are exponentiated, and the std.error is transformed to exp(estimate)*std.error.

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 section below.

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 through to the extractor and table-making functions (by default broom::tidy and kableExtra::kbl, but this can be customized). This allows users to pass arguments directly to modelsummary in order to affect the behavior of other functions behind the scenes. For example,

- `metrics="none"`, `metrics="all"`, or `metrics=c("R2", "RMSE")` to select the goodness-of-fit extracted by the `performance` package (must have set `options(modelsummary_get="easystats")` first). This can be useful for some models when statistics take a long time to compute. See `?performance::performance`

## 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)

# add_rows: add an empty reference category

mod <- lm(hp ~ factor(cyl), mtcars)

add_rows = data.frame(
  term = "factory(cyl)4",
  model = "Model 1",
  estimate = NA)
attr(add_rows, "position") = 3
modelplot(mod, add_rows = add_rows)


# 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` *Model Summary Tables*

---

## Description

Create beautiful and customizable tables to summarize several statistical models side-by-side. This function supports dozens of statistical models, and it can produce tables in HTML, LaTeX, Word, Markdown, PDF, PowerPoint, Excel, RTF, JPG, or PNG. The appearance of the tables can be customized extensively by specifying the `output` argument, and by using functions from one of the supported table customization packages: `kableExtra`, `gt`, `flextable`, `huxtable`. For more information, see the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://vincentarelbundock.github.io/modelsummary/

- The `modelsummary` Vignette includes dozens of examples of tables with extensive customizations.
- The Appearance Vignette shows how to modify the look of tables.

## Usage

```
modelsummary(
  models,
  output = "default",
  fmt = 3,
  estimate = "estimate",
  statistic = "std.error",
  vcov = NULL,
  conf_level = 0.95,
  exponentiate = FALSE,
  stars = FALSE,
  shape = term + statistic ~ model,
  coef_map = NULL,
  coef_omit = NULL,
  coef_rename = NULL,
  gof_map = NULL,
  gof_omit = NULL,
```

```
  group_map = NULL,
  add_rows = NULL,
  align = NULL,
  notes = NULL,
  title = NULL,
  escape = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| models | a model or (optionally named) list of models |
| output | filename or object type (character string)<br>• Supported filename extensions: .docx, .html, .tex, .md, .txt, .png, .jpg.<br>• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "gt", "kableExtra", "huxtable", "flextable", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.<br>• Warning: Users should not supply a file name to the `output` argument if they intend to customize the table with external packages. See the 'Details' section.<br>• LaTeX compilation requires the `booktabs` and `siunitx` packages, but `siunitx` can be disabled or replaced with global options. See the 'Details' section.<br>• The default output formats and table-making packages can be modified with global options. See the 'Details' section. |
| fmt | determines how to format numeric values<br>• integer: the number of digits to keep after the period `format(round(x, fmt), nsmall=fmt)`<br>• character: passed to the `sprintf` function (e.g., '%.3f' keeps 3 digits with trailing zero). See `?sprintf`<br>• function: returns a formatted character string.<br>• NULL: does not format numbers, which allows users to include function in the "glue" strings in the `estimate` and `statistic` arguments.<br>• A named list to format distinct elements of the table differently. Names correspond to column names produced by `get_estimates(model)` or `get_gof(model)`. Values are integers, characters, or functions, as described above. The `fmt` element is used as default for unspecified elements Ex: `fmt=list("estimate"=2, "std.error"=1, "r.squared"=4, "fmt"=3)`<br>• LaTeX output: To ensure proper typography, all numeric entries are enclosed in the `\num{}` command, which requires the `siunitx` package to be loaded in the LaTeX preamble. This behavior can be altered with global options. See the 'Details' section. |
| estimate | a single string or a character vector of length equal to the number of models. Valid entries include any column name of the data.frame produced by `get_estimates(model)`, and strings with curly braces compatible with the `glue` package format. Examples: |

- "estimate"
- "{estimate} ({std.error}){stars}"
- "{estimate} [{conf.low}, {conf.high}]"

statistic — vector of strings or glue strings which select uncertainty statistics to report vertically below the estimate. NULL omits all uncertainty statistics.

- "conf.int", "std.error", "statistic", "p.value", "conf.low", "conf.high", or any column name produced by: get_estimates(model)
- glue package strings with braces, with or without R functions, such as:
  - "{p.value} [{conf.low}, {conf.high}]"
  - "Std.Error: {std.error}"
  - ‘"exp(estimate) * std.error"
- Numbers are automatically rounded and converted to strings. To apply functions to their numeric values, as in the last glue example, users must set fmt=NULL.
- Parentheses are added automatically unless the string includes glue curly braces {}.

vcov — robust standard errors and other manual statistics. The vcov argument accepts six types of input (see the 'Details' and 'Examples' sections below):

- NULL returns the default uncertainty estimates of the model object
- string, vector, or (named) list of strings. "iid", "classical", and "constant" are aliases for NULL, which returns the model's default uncertainty estimates. The strings "HC", "HC0", "HC1" (alias: "stata"), "HC2", "HC3" (alias: "robust"), "HC4", "HC4m", "HC5", "HAC", "NeweyWest", "Andrews", "panel-corrected", "outer-product", and "weave" use variance-covariance matrices computed using functions from the sandwich package, or equivalent method. The behavior of those functions can (and sometimes *must*) be altered by passing arguments to sandwich directly from modelsummary through the ellipsis (. . . ), but it is safer to define your own custom functions as described in the next bullet.
- function or (named) list of functions which return variance-covariance matrices with row and column names equal to the names of your coefficient estimates (e.g., stats::vcov, sandwich::vcovHC, function(x) vcovPC(x, cluster="country")).
- formula or (named) list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).
- named list of length(models) variance-covariance matrices with row and column names equal to the names of your coefficient estimates.
- a named list of length(models) vectors with names equal to the names of your coefficient estimates. See 'Examples' section below. Warning: since this list of vectors can include arbitrary strings or numbers, modelsummary cannot automatically calculate p values. The stars argument may thus use incorrect significance thresholds when vcov is a list of vectors.

conf_level — confidence level to use for confidence intervals

exponentiate — TRUE, FALSE, or logical vector of length equal to the number of models. If TRUE, the estimate, conf.low, and conf.high statistics are exponentiated, and the std.error is transformed to exp(estimate)*std.error.

`stars` to indicate statistical significance

- FALSE (default): no significance stars.
- TRUE: +=.1, *=.05, **=.01, ***=0.001
- Named numeric vector for custom stars such as `c('*' = .1, '+' = .05)`
- Note: a legend will not be inserted at the bottom of the table when the `estimate` or `statistic` arguments use "glue strings" with `{stars}`.

`shape` formula which determines the shape of the table. The left side determines what appears on rows, and the right side determines what appears on columns. The formula can include a group identifier to display related terms together, which can be useful for models with multivariate outcomes or grouped coefficients (See examples section below). This identifier must be one of the column names produced by: `get_estimates(model)`. If an incomplete formula is supplied (e.g., `~statistic`), `modelsummary` tries to complete it automatically. Potential `shape` values include:

- `term + statistic ~ model`: default
- `term ~ model + statistic`: statistics in separate columns
- `model + statistic ~ term`: models in rows and terms in columns
- `term + response + statistic ~ model`:
- `term ~ response`

`coef_map` character vector. Subset, rename, and reorder coefficients. Coefficients omitted from this vector are omitted from the table. The order of the vector determines the order of the table. `coef_map` can be a named or an unnamed character vector. If `coef_map` is a named vector, its values define the labels that must appear in the table, and its names identify the original term names stored in the model object: `c("hp:mpg"="HPxM/G")`. See Examples section below.

`coef_omit` string regular expression (perl-compatible) used to determine which coefficients to omit from the table. A "negative lookahead" can be used to specify which coefficients to *keep* in the table. Examples:

- `"ei"`: omit coefficients matching the "ei" substring.
- `"^Volume$"`: omit the "Volume" coefficient.
- `"ei|rc"`: omit coefficients matching either the "ei" or the "rc" substrings.
- `"^(?!Vol)"`: keep coefficients starting with "Vol" (inverse match using a negative lookahead).
- `"^(?!.*ei)"`: keep coefficients matching the "ei" substring.
- `"^(?!.*ei|.*pt)"`: keep coefficients matching either the "ei" or the "pt" substrings.
- See the Examples section below for complete code.

`coef_rename` named character vector or function

- 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. Ex: c("hp:mpg"="hp X mpg")
- Function: Accepts a character vector of the model's term names and returns a named vector like the one described above. The `modelsummary` package supplies a `coef_rename()` function which can do common cleaning tasks: `modelsummary(model, coef_rename = coef_rename)`

`gof_map` rename, reorder, and omit goodness-of-fit statistics and other model information. This argument accepts 4 types of values:

- NULL (default): the `modelsummary::gof_map` dictionary is used for formatting, and all unknown statistic are included.
- NA: excludes all statistics from the bottom part of the table.
- character vector such as `c("rmse", "nobs", "r.squared")`. Elements correspond to colnames in the data.frame produced by `get_gof(model)`. The default dictionary is used to format and rename statistics.
- data.frame with 3 columns named "raw", "clean", "fmt". Unknown statistics are omitted. See the 'Examples' section below.
- list of lists, each of which includes 3 elements named "raw", "clean", "fmt". Unknown statistics are omitted. See the 'Examples section below'.

`gof_omit` string regular expression (perl-compatible) used to determine which statistics to omit from the bottom section of the table. A "negative lookahead" can be used to specify which statistics to *keep* in the table. Examples:

- `"IC"`: omit statistics matching the "IC" substring.
- `"BIC|AIC"`: omit statistics matching the "AIC" or "BIC" substrings.
- `"^(?!.*IC)"`: keep statistics matching the "IC" substring.

`group_map` named or unnamed character vector. Subset, rename, and reorder coefficient groups specified a grouping variable specified in the `shape` argument formula. This argument behaves like `coef_map`.

`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 section below.

`align` A string with a number of characters equal to the number of columns in the table (e.g., `align = "lcc"`). Valid characters: l, c, r, d.

- "l": left-aligned column
- "c": centered column
- "r": right-aligned column
- "d": dot-aligned column. Only supported for LaTeX/PDF tables produced by `kableExtra`. These commands must appear in the LaTeX preamble (they are added automatically when compiling Rmarkdown documents to PDF):
  - `\usepackage{booktabs}`
  - `\usepackage{siunitx}`
  - `\newcolumntype{d}{S[input-symbols = ()]}`

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

`title` string

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. This setting does not affect captions or notes.

... all other arguments are passed through to the extractor and table-making functions (by default `broom::tidy` and `kableExtra::kbl`, but this can be customized). This allows users to pass arguments directly to `modelsummary` in order to affect the behavior of other functions behind the scenes. For example,

- `metrics="none"`, `metrics="all"`, or `metrics=c("R2", "RMSE")` to select the goodness-of-fit extracted by the `performance` package (must have set `options(modelsummary_get="easystats")` first). This can be useful for some models when statistics take a long time to compute. See `?performance::performance`

## Details

`output`:

The `modelsummary_list` output is a lightweight format which can be used to save model results, so they can be fed back to `modelsummary` later to avoid extracting results again.

When a file name with a valid extension 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 an external package, you need to choose a different output format and saving mechanism. Unfortunately, the approach differs from package to package:

- `gt`: set `output="gt"`, post-process your table, and use the `gt::gtsave` function.
- `kableExtra`: set `output` to your destination format (e.g., "latex", "html", "markdown"), post-process your table, and use `kableExtra::save_kable` function.

`vcov`:

To use a string such as "robust" or "HC0", your model must be supported by the `sandwich` package. This includes objects such as: lm, glm, survreg, coxph, mlogit, polr, hurdle, zeroinfl, and more.

NULL, "classical", "iid", and "constant" are aliases which do not modify uncertainty estimates and simply report the default standard errors stored in the model object.

One-sided formulas such as `~clusterid` are passed to the `sandwich::vcovCL` function.

Matrices and functions producing variance-covariance matrices are first passed to `lmtest`. If this does not work, `modelsummary` attempts to take the square root of the diagonal to adjust "std.error", but the other uncertainty estimates are not be adjusted.

Numeric vectors are formatted according to `fmt` and placed in brackets. Character vectors printed as given, without parentheses.

If your model type is supported by the `lmtest` package, the `vcov` argument will try to use that package to adjust all the uncertainty estimates, including "std.error", "statistic", "p.value", and "conf.int". If your model is not supported by `lmtest`, only the "std.error" will be adjusted by, for example, taking the square root of the matrix's diagonal.

## Value

a regression table in a format determined by the `output` argument.

## Global Options

The behavior of `modelsummary` can be affected by setting global options:

- `modelsummary_factory_default`
- `modelsummary_factory_latex`
- `modelsummary_factory_html`
- `modelsummary_factory_png`
- `modelsummary_get`
- `modelsummary_format_numeric_latex`
- `modelsummary_format_numeric_html`

**Table-making packages:**

`modelsummary` supports 4 table-making packages: `kableExtra`, `gt`, `flextable`, and `huxtable`. Some of these packages have overlapping functionalities. For example, 3 of those packages can export to LaTeX. To change the default backend used for a specific file format, you can use the `options` function:

`options(modelsummary_factory_html = 'kableExtra')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')`

**Model extraction functions:**

`modelsummary` can use two sets of packages to extract information from statistical models: the `easystats` family (`performance` and `parameters`) and `broom`. By default, it uses `easystats` first and then falls back on `broom` in ase of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "broom")` `options(modelsummary_get = "easystats")` `options(modelsummary_get = "all")`

**Formatting numeric entries:**

By default, LaTeX tables enclose all numeric entries in the `\num{}` command from the siunitx package. To prevent this behavior, or to enclose numbers in dollar signs (for LaTeX math mode), users can call:

`options(modelsummary_format_numeric_latex = "plain")` `options(modelsummary_format_numeric_latex = "mathmode")`

A similar option can be used to display numerical entries using MathJax in HTML tables:

`options(modelsummary_format_numeric_html = "mathjax")`

## Parallel computation

It can take a long time to compute and extract summary statistics from certain models (e.g., Bayesian). In those cases, users can parallelize the process. Since parallelization occurs at the model level, no speedup is available for tables with a single model. To use parallel computation, all users have to do is load the `future.apply` package and call the `plan()` function. Example:

```
library(future.apply)
plan("multisession")
modelsummary(model_list)
```

## Examples

```
## Not run:

# The `modelsummary` website includes \emph{many} examples and tutorials:
# https://vincentarelbundock.github.io/modelsummary

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
modelsummary(models)

# statistic
modelsummary(models, statistic = NULL)
modelsummary(models, statistic = 'p.value')
modelsummary(models, statistic = 'statistic')
modelsummary(models, statistic = 'conf.int', conf_level = 0.99)
modelsummary(models, statistic = c("t = {statistic}",
                                   "se = {std.error}",
                                   "conf.int"))

# estimate
modelsummary(models,
  statistic = NULL,
  estimate = "{estimate} [{conf.low}, {conf.high}]")
modelsummary(models,
  estimate = c("{estimate}{stars}",
               "{estimate} ({std.error})"))

# vcov
modelsummary(models, vcov = "robust")
modelsummary(models, vcov = list("classical", "stata"))
modelsummary(models, vcov = sandwich::vcovHC)
modelsummary(models,
  vcov = list(stats::vcov, sandwich::vcovHC))
modelsummary(models,
  vcov = list(c("(Intercept)"="", "Height"="!"),
              c("(Intercept)"="", "Height"="!", "Volume"="!!")))

# vcov with custom names
modelsummary(
  models,
  vcov = list("Stata Corp" = "stata",
              "Newey Lewis & the News" = "NeweyWest"))

# coef_rename
modelsummary(models, coef_rename = c('Volume' = 'Large', 'Height' = 'Tall'))
```

```
modelsummary(models, coef_rename = toupper)
modelsummary(models, coef_rename = coef_rename)

# coef_map
modelsummary(models, coef_map = c('Volume' = 'Large', 'Height' = 'Tall'))
modelsummary(models, coef_map = c('Volume', 'Height'))

# coef_omit: omit coefficients matching one substring
modelsummary(models, coef_omit = "ei", omit = ".*")

# coef_omit: omit a specific coefficient
modelsummary(models, coef_omit = "^Volume$", gof_omit = ".*", output = "markdown")

# coef_omit: omit coefficients matching either one of two substring
modelsummary(models, coef_omit = "ei|rc", omit = ".*")

# coef_omit: keep coefficients starting with a substring (using a negative lookahead)
modelsummary(models, coef_omit = "^(?!Vol)", omit = ".*")

# coef_omit: keep coefficients matching a substring
modelsummary(models, coef_omit = "^(?!.*ei|.*pt)", omit = ".*")

# shape
library(nnet)
multi <- multinom(factor(cyl) ~ mpg + hp, data = mtcars, trace = FALSE)
modelsummary(multi, shape = y.level ~ model)
modelsummary(multi, shape = term ~ y.level)

# title
modelsummary(models, title = 'This is the title')

# title with LaTeX label (for numbering and referencing)
modelsummary(models, title = 'This is the title \\label{tab:description}')

# add_rows
rows <- tibble::tribble(~term, ~Bivariate, ~Multivariate,
  'Empty row', '-', '-',
  'Another empty row', '?', '?')
attr(rows, 'position') <- c(1, 3)
modelsummary(models, add_rows = rows)

# notes
modelsummary(models, notes = list('A first note', 'A second note'))

# gof_map: tribble
library(tibble)
gm <- tribble(
  ~raw,        ~clean,      ~fmt,
  "r.squared", "R Squared", 5)
modelsummary(models, gof_map = gm)

# gof_map: data.frame
gm <- modelsummary::gof_map
```

```
gm$omit[gm$raw == 'deviance'] <- FALSE
gm$fmt[gm$raw == 'r.squared'] <- "%.5f"
modelsummary(models, gof_map = gm)

# gof_map: list of lists
f1 <- function(x) format(round(x, 3), big.mark=",")
f2 <- function(x) format(round(x, 0), big.mark=",")
gm <- list(
  list("raw" = "nobs", "clean" = "N", "fmt" = f2),
  list("raw" = "AIC", "clean" = "aic", "fmt" = f1))
modelsummary(models,
  fmt = f1,
  gof_map = gm)


## End(Not run)
```

---

N *datasummary statistic shortcut*

---

## Description

datasummary statistic shortcut

## Usage

```
N(x)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |

## Examples

```
## Not run:
datasummary(Factor(cyl) ~ N, data = mtcars)

## End(Not run)
```

## Ncol *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

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

### Arguments

| | |
|---|---|
| x | variable to summarize |
| ... | unused |

## NPercent *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

```
NPercent(x, y)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| y | denominator variable |

## NUnique *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

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

### Arguments

| | |
|---|---|
| x | variable to summarize |
| ... | unused |

### Examples

```
## Not run:
datasummary(cyl + hp ~ NUnique, data = mtcars)

## End(Not run)
```

## P0 *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

```
P0(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the `modelsummary:::rounding` function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

---

P100 *datasummary statistic shortcut*

---

## Description

datasummary statistic shortcut

## Usage

```
P100(x, fmt = NULL, na.rm = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

P25 *datasummary statistic shortcut*

## Description

datasummary statistic shortcut

## Usage

```
P25(x, fmt = NULL, na.rm = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

P50 *datasummary statistic shortcut*

## Description

datasummary statistic shortcut

## Usage

```
P50(x, fmt = NULL, na.rm = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

---

P75 *datasummary statistic shortcut*

---

## Description

datasummary statistic shortcut

## Usage

```
P75(x, fmt = NULL, na.rm = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

## PercentMissing *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

```
PercentMissing(x)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |

## SD *datasummary statistic shortcut*

### Description

datasummary statistic shortcut

### Usage

```
SD(x, fmt = NULL, na.rm = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the `modelsummary:::rounding` function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

### Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

Var *datasummary statistic shortcut*

## Description

datasummary statistic shortcut

## Usage

```
Var(x, fmt = NULL, na.rm = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | variable to summarize |
| fmt | passed to the modelsummary:::rounding function |
| na.rm | a logical value indicating whether 'NA' values should be stripped before the computation proceeds. |
| ... | unused |

## Examples

```
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)

## End(Not run)
```

# Index