Package ‘modelsummary’

April 19, 2021

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 0.7.0

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

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

Depends R (>= 3.5.0)

Imports broom,
   checkmate,
   generics,
   glue,
   insight,
   kableExtra (>= 1.2.1),
   parameters,
   performance,
   tables,
   tidyr,
   tidyselect

Suggests broom.mixed,
   estimatr,
   flextable,
   fixest,
   gamlss,
   ggplot2,
   gt,
   huxtable,
   knitr,
   lme4,
   lmtest,
   nnet,
R topics documented:

- coef_rename
- datasummary
- datasummary_balance
- datasummary_correlation
- datasummary_correlation_format
- datasummary_df
- datasummary_skim
- dvnames
- get_estimates
- get_gof
- glance.modelsummary_list
- glance_custom
- gof_map
- Histogram
- Max
- Mean
- Median
- Min
- modelplot
- modelsummary
- modelsummary_wide
- N
- Ncol
- NPercent
- NUnique
- P0
- P100
- P25
- P50
- P75
- PercentMissing
- SD
- supported_models
- tidy.modelsummary_list
- tidy_custom
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

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

Description

data_summary 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.
Usage
datasummary(
  formula,
  data,
  output = "default",
  fmt = 2,
  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 (character string)
  • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
  • Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
  • To change the default output format, type options(modelsummary_default = "latex"), where latex can be any of the valid object types listed above.
  • Warning: the output argument cannot be used when customizing tables with external packages.
    • See the 'Details' section below for more information.
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., \)
  • function: returns a formatted character string.
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 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.'
**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**

```r
## 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)
```
dataset_summary_balance

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

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

dataset_summary(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

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

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

dataset_summary(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
dataset_summary(f, data = mtcars, output = 'table.html')
dataset_summary(f, data = mtcars, output = 'table.tex')
dataset_summary(f, data = mtcars, output = 'table.md')
dataset_summary(f, data = mtcars, output = 'table.docx')
dataset_summary(f, data = mtcars, output = 'table.pptx')
dataset_summary(f, data = mtcars, output = 'table.jpg')
dataset_summary(f, data = mtcars, output = 'table.png')

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

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

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

Usage

datasummary_balance(
  formula,  
data,  
  output = "default",  
fmt = 1,  
title = NULL,  
notes = NULL,  
align = NULL,  
add_columns = NULL,  
add_rows = NULL,  
dinn = TRUE,  
dinn_statistic = "std.error",  
...  
)

Arguments

formula a one-sided formula with the "condition" or "column" variable on the right-hand side.
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.
output filename or object type (character string)
  • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
  • Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
  • To change the default output format, type options(modelsummary_default = "latex"), where latex can be any of the valid object types listed above.
  • Warning: the output argument cannot be used when customizing tables with external packages.
  • See the 'Details' section below for more information.
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., \"
  • function: returns a formatted character string.
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 section below.
**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.

**Usage**

```r
datasummary_correlation(
  data,
  output = "default",
  fmt = 2,
  title = NULL,
  notes = NULL,
  method = "pearson",
  ...
)
```

**Arguments**

- **data**  
  A data.frame (or tibble)

- **output**  
  filename or object type (character string)
  - Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.

**Examples**

```r
## Not run:
datasummary_balance(~am, mtcars)
## End(Not run)
```
• Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
• To change the default output format, type `options(modelsummary_default = "latex")`, where latex can be any of the valid object types listed above.
• Warning: the output argument cannot be used when customizing tables with external packages.
• See the 'Details' section below for more information.

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., `\`)
• function: returns a formatted character string.

title
string

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

method
character or function
• character: "pearson", "kendall", "spearman", or "pearspear" (Pearson correlations above and Spearman correlations below the diagonal)
• 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.

Examples

```r
## 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
  • integer: the number of digits to keep after the period format(round(x,fmt),nsmall=fmt)
  • character: passed to the sprintf function (e.g., '
  • function: returns a formatted character string.
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_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,
  ...
)

Arguments

data A data.frame (or tibble)
output filename or object type (character string)
  • Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
  • Supported object types: "default", "html", "markdown", "latex", "latex_tabular",
    "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
  • To change the default output format, type options(modelsummary_default = "latex"),
    where latex can be any of the valid object types listed above.
  • Warning: the output argument cannot be used when customizing tables
    with external packages.
  • See the 'Details' section below for more information.
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., \\)
  • function: returns a formatted character string.
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.
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.
...

all other arguments are passed through to the extractor and table-making functions.
This allows users to pass arguments directly to modelsummary in order to
affect the behavior of other functions behind the scenes. Examples include:
  • broom::tidy(exponentiate=TRUE) to exponentiate logistic regression
  • kableExtra::kbl(escape=FALSE) to avoid escaping math characters in
    kableExtra tables.
• performance::model_performance(metrics="RMSE") to select goodness-of-fit statistics to extract using the performance package (must have set options(modelsummary_get="easystats") first).

### Description

This function was inspired by the excellent skimr package for R.

### Usage

```r
datasummary_skim(
data,  
type = "numeric",  
output = "default",  
fmt = "%.1f",  
histogram = TRUE,  
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 (character string)
  - Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
  - Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
  - To change the default output format, type options(modelsummary_default = "latex"), where latex can be any of the valid object types listed above.
  - Warning: the output argument cannot be used when customizing tables with external packages.
  - See the ‘Details’ section below for more information.
- **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., \\
  - function: returns a formatted character string.
- **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.
### 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")`
get_estimates

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. A mostly internal function with some potential uses outside.

Description

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

Usage

generate_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 five types of input (see the 'Details' and 'Examples' sections below):

• string, vector, or list of strings: "robust", "HC", "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "stata", or "classical" (alias "constant" or "iid").

• formula or list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).

• function or 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).

• list of length(models) variance-covariance matrices with row and column names equal to the names of your coefficient estimates.
get_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. This allows users to pass arguments directly to modelsummary in order to affect the behavior of other functions behind the scenes. Examples include:

- `broom::tidy(exponentiate=TRUE)` to exponentiate logistic regression
- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.
- `performance::model_performance(metrics="RMSE")` to select goodness-of-fit statistics to extract using the performance package (must have set `options(modelsummary_get="easystats")` first).
glance.modelsummary_list

Extract goodness-of-fit statistics from a modelsummary_list object.

Description

A modelsummary_list is a simple list which holds two data.frames names "glance" and "tidy", and to which we have attached the class "modelsummary_list". This allows us to manually create objects that will be printed nicely by modelsummary (see Examples section).

Usage

```r
## S3 method for class 'modelsummary_list'
glance(x, ...)
```

Arguments

- `x` a list of class modelsummary_list (see Examples section)
- `...` other parameters are accepted by ignored

Examples

```r
## Not run:
mod <- list(
tidy = data.frame(term = c("A", "B"),
estimate = 1:2,
std.error = 3:4),
glance = data.frame(nobs = 10))
class(mod) <- c("modelsummary_list", class(mod))
modelsummary(mod)
## End(Not run)
```

---

glance_custom

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

Description

To customize the output of a model of class lm, you can define a new method called glance_custom.lm which returns a one-row data.frame.

Usage

```r
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 '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

datsumary statistic shortcut

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

Mean
datsumary statistic shortcut

Description
datsumary 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
## Median

**datasummary statistic shortcut**

### Description

datasummary statistic shortcut

### Usage

```r
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

```r
## 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

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

### Examples

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

## End(Not run)
```
modelplot

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:
datsummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
Min + Max + SD + Var,
data = mtcars)

## End(Not run)

---

Model Summary Plots with Estimates and Confidence Intervals

Description

Model Summary Plots with Estimates and Confidence Intervals

Usage

modelplot(
  models,
  conf_level = 0.95,
  coef_map = NULL,
  coef_omit = NULL,
  coef_rename = NULL,
  vcov = NULL,
  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 (see the Examples section below). 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")`. 
modelplot

coef_omit string regular expression. Omits all matching coefficients from the table using
`grepl(perl=TRUE)`.

coef_rename named character vector or function which returns a named vector. Values of
the vector 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. c("hp:mpg"="hp X mpg")
for an interaction term.

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

  • string, vector, or list of strings: "robust", "HC", "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "stata", or "classical" (alias "constant" or "iid").
  • formula or list of formulas with the cluster variable(s) on the right-hand
    side (e.g., ~clusterid).
  • function or 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`).
  • 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 coef-
    ficient estimates. See 'Examples' section below. Warning: since this list of
    vectors can include arbitrary strings or numbers, modelsummary cannot au-
    tomatically calculate p values. The stars argument may thus use incorrect
    significance thresholds when vcov is a list of vectors.

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 func-
ations. This allows users to pass arguments directly to modelsummary in order to
affect the behavior of other functions behind the scenes. Examples include:

  • broom::tidy(exponentiate=TRUE) to exponentiate logistic regression
  • kableExtra::kbl(escape=FALSE) to avoid escaping math characters in
    kableExtra tables.
  • performance::model_performance(metrics="RMSE") to select goodness-
    of-fit statistics to extract using the performance package (must have set
    options(modelsummary_get="easystats") first).

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 = c("Var", 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)
Description

The content of the tables can be altered with the function’s arguments, or by calling `options`, as described in the `Details` section below. The look of the tables can be customized by specifying the `output` argument, and by using functions from one of the supported table customization packages: `kableExtra`, `gt`, `flextable`, `huxtable`.

Usage

```r
modelsummary(
  models,  # a model or (optionally named) list of models
  output = "default",  # filename or object type (character string)
  fmt = 3,  # integer: the number of digits to keep after the period
  estimate = "estimate",  # estimate
  statistic = "std.error",  # statistic
  vcov = NULL,  # vcov
  conf_level = 0.95,  # conf_level
  stars = FALSE,  # stars
  coef_map = NULL,  # coef_map
  coef_omit = NULL,  # coef_omit
  coef_rename = NULL,  # coef_rename
  gof_map = NULL,  # gof_map
  gof_omit = NULL,  # gof_omit
  group = term ~ model,  # group
  group_map = NULL,  # group_map
  add_rows = NULL,  # add_rows
  align = NULL,  # align
  notes = NULL,  # notes
  title = NULL,  # title
  ...  # ...
)
```

Arguments

- **models**: a model or (optionally named) list of models
- **output**: filename or object type (character string)
  - Supported filename extensions: `.html`, `.tex`, `.md`, `.txt`, `.png`, `.jpg`.
  - Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
  - To change the default output format, type `options(modelsummary_default = "latex")`, where `latex` can be any of the valid object types listed above.
  - Warning: the `output` argument *cannot* be used when customizing tables with external packages.
  - See the 'Details' section below for more information.
- **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., `\`)
• function: returns a formatted character string.

**estimate**

string or glue string of the estimate to display (or a vector with one string per model). Valid entries include any column name of the data.frame produced by `get_estimates(model)`. 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, such as:
  – "{p.value} [{conf.low},{conf.high}]"
  – "Std.Error: {std.error}"

• Note: Parentheses are added automatically unless the string includes glue curly braces `{}`.
• Note: To report uncertainty statistics next to coefficients, you can `#` supply a glue string to the `estimate` argument.

**vcov**

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

• string, vector, or list of strings: "robust", "HC", "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "stata", or "classical" (alias "constant" or "iid").
• formula or list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).
• function or 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).
• 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. 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

**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)
• Note: a legend will not be inserted at the bottom of the table when the `estimate` or `statistic` arguments use "glue strings" with `{stars}`.

**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 (see the Examples section below). 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").
coef_omit
string regular expression. Omits all matching coefficients from the table using `grepl(perl=TRUE)`.

c coef_rename
named character vector or function which returns a named vector. Values of the vector 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. `c("hp:mpg"="hp X mpg")` for an interaction term.

gof_map
- NULL (default): the `modelsummary::gof_map` dictionary is used for formatting, and all unknown statistic are included.
- 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. Omits all matching gof statistics from the table (using `grepl(perl=TRUE)`).

group
a two-sided formula with two or three components which describes how groups of parameters should be displayed. The formula must include both a "term" and a "model" component. In addition, a component can be used to identify groups of parameters (e.g., outcome levels of a multinomial logit model). This group identifier must be the name of a column in the data.frame produced by `get_estimates(model)`.

- `term ~ model` displays coefficients as rows and models as columns
- `model ~ term` displays models as rows and coefficients as columns
- `response + term ~ model` displays response levels and coefficients as rows and models as columns.

group_map
named or unnamed character vector. Subset, rename, and reorder coefficient groups specified in the group argument. See 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 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.

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

title
string

... all other arguments are passed through to the extractor and table-making functions. This allows users to pass arguments directly to `modelsummary` in order to affect the behavior of other functions behind the scenes. Examples include:

- `broom::tidy(exponentiate=TRUE)` to exponentiate logistic regression
- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.
- `performance::model_performance(metrics="RMSE")` to select goodness-of-fit statistics to extract using the `performance` package (must have set `options(modelsummary_get="easystats")` first).

Details

options
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:

```r
options(modelsummary_html = 'kableExtra') options(modelsummary_latex = 'gt') options(modelsummary_word = 'huxtable') options(modelsummary_png = 'gt')
```

modelsummary can use two sets of packages to extract information from statistical models: broom and the easystats family (performance and parameters). By default, it uses broom first and easystats as a fallback if broom fails. You can change the order of priorities or include goodness-of-fit extracted by both packages by setting:

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

output argument:
The `modelsummary_list` output type is a lightweight representation of the model results. The modelsummary function can export to this format by setting the output argument, and it can accept objects of this format as input models to create a table. This can be useful to save raw results, in order to print a table later, without having to save and extract from the entire model object. Note that the confidence intervals are only stored in a `modelsummary_list` if explicitly requested:

```r
backup <- modelsummary(models, output = "modelsummary_list" statistic = "conf.int")
modelsummary(backup)
```

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 argument:
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.

"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.
Examples

## Not run:

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

library(modelsummary)

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,
              statistic = NULL,
              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"="!!")))

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

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

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

# add_rows
rows <- tibble::tribble(~term, ~Bivariate, ~Multivariate,
```

modelsummary_wide

'Summary Tables for Models with Grouped Coefficients'

**Description**

modelsummary_wide summarizes models with grouped coefficients. For example, these groups could correspond to levels of a multinomial logit outcome variable, or to parameters of a GAMLSS model. This function's arguments are the same as in modelsummary, except for the coef_group and the stacking arguments.

**Usage**

```r
modelsummary_wide(models, 
output = "default", 
fmt = 3, 
estimate = "estimate", 
statistic = "std.error", 
vcov = NULL, 
conf_level = 0.95, 
stars = FALSE, 
coef_group = NULL, 
coef_map = NULL, 
coef_omit = NULL, 
coef_rename = NULL,
```

---

# End(Not run)
Arguments

models: a model or (optionally named) list of models

output: filename or object type (character string)
- Supported filename extensions: .html, .tex, .md, .txt, .png, .jpg.
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "data.frame", "modelsummary_list", "gt", "kableExtra", "huxtable", "flextable".
- To change the default output format, type `options(modelsummary_default = "latex")`, where `latex` can be any of the valid object types listed above.
- Warning: the `output` argument cannot be used when customizing tables with external packages.
- See the 'Details' section below for more information.

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., `{\}
- function: returns a formatted character string.

estimate: string or glue string of the estimate to display (or a vector with one string per model). Valid entries include any column name of the data.frame produced by `get_estimates(model)`. 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, such as:
  - "{{p.value}} [{conf.low},{conf.high}]"
  - "Std.Error: {std.error}"
- Note: Parentheses are added automatically unless the string includes glue curly braces {}.
- Note: To report uncertainty statistics next to coefficients, you can supply a glue string to the `estimate` argument.

vcov: robust standard errors and other manual statistics. The `vcov` argument accepts five types of input (see the 'Details' and 'Examples' sections below):
- string, vector, or list of strings: "robust", "HC", "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "stata", or "classical" (alias "constant" or "iid").
• formula or list of formulas with the cluster variable(s) on the right-hand side (e.g., ~clusterid).
• function or 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).
• 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. 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

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)
• Note: a legend will not be inserted at the bottom of the table when the estimate or statistic arguments use "glue strings" with {stars}.

coeff_group the name of the coefficient groups to use as columns (NULL or character). If coeff_group is NULL, modelsummary tries to guess the correct coefficient group identifier. To be valid, this identifier must be a column in the data.frame produced by get_estimates(model).

coeff_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. coeff_map can be a named or an unnamed character vector (see the Examples section below). If coeff_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").

coeff_omit string regular expression. Omits all matching coefficients from the table using grepl(perl=TRUE).

coeff_rename named character vector or function which returns a named vector. Values of the vector 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. c("hp:mpg"="hp X mpg") for an interaction term.

gof_map • NULL (default): the modelsummary::gof_map dictionary is used for formatting, and all unknown statistic are included.
• 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. Omits all matching gof statistics from the table (using grepl(perl=TRUE)).

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 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.

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

title: string

stacking: direction in which models are stacked: "horizontal" or "vertical"

... all other arguments are passed through to the extractor and table-making functions. This allows users to pass arguments directly to `modelsummary` in order to affect the behavior of other functions behind the scenes. Examples include:

- `broom::tidy(exponentiate=TRUE)` to exponentiate logistic regression
- `kableExtra::kbl(escape=FALSE)` to avoid escaping math characters in `kableExtra` tables.
- `performance::model_performance(metrics="RMSE")` to select goodness-of-fit statistics to extract using the `performance` package (must have set `options(modelsummary_get="easystats")` first).

Value

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

---

N(x) | datasummary statistic shortcut

Description

datasummary statistic shortcut

Usage

N(x)

Arguments

x: variable to summarize

Examples

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

*datasummary statistic shortcut*

**Description**

datasummary statistic shortcut

**Usage**

\[\text{Ncol}(x, \ldots)\]

**Arguments**

- \(x\) variable to summarize
- \(\ldots\) unused

**NPercent**

*datasummary statistic shortcut*

**Description**

datasummary statistic shortcut

**Usage**

\[\text{NPercent}(x, y)\]

**Arguments**

- \(x\) variable to summarize
- \(y\) denominator variable

**NUnique**

*datasummary statistic shortcut*

**Description**

datasummary statistic shortcut

**Usage**

\[\text{NUnique}(x, \ldots)\]

**Arguments**

- \(x\) variable to summarize
- \(\ldots\) unused
Examples

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

---

**P0**  
*datasummary statistic shortcut*

**Description**

datasummary statistic shortcut

**Usage**

```r
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**

```r
## 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**

```r
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

**Description**

datasummary statistic shortcut

**Usage**

```r
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**

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

P75

**Description**

datasummary statistic shortcut

**Usage**

```r
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

```r
## 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

### Usage

```r
PercentMissing(x)
```

#### Arguments

- `x`: variable to summarize

## SD
datasummary statistic shortcut

### Usage

```r
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

```r
## Not run:
datasummary(mpg + hp ~ Mean + Median + P0 + P25 + P50 + P75 + P100 +
            Min + Max + SD + Var,
            data = mtcars)
## End(Not run)
```
supported_models  List of model objects from which modelsummary can extract estimates and statistics

Description
List of model objects from which modelsummary can extract estimates and statistics

Usage

supported_models()


tidy.modelsummary_list
Extract parameter estimates from a modelsummary_list object.

Description
A modelsummary_list is a simple list which holds two data.frames names "glance" and "tidy", and to which we have attached the class "modelsummary_list". This allows us to manually create objects that will be printed nicely by modelsummary (see Examples section).

Usage

## S3 method for class 'modelsummary_list'
tidy(x, ...)

Arguments

x  a list of class modelsummary_list (see Examples section)
...
other parameters are accepted by ignored

Examples

## Not run:
mod <- list(
  tidy = data.frame(term = c("A", "B"),
                   estimate = 1:2,
                   std.error = 3:4),
  glance = data.frame(nobs = 10))
class(mod) <- c("modelsummary_list", class(mod))
modelsummary(mod)

## End(Not run)
tidy_custom

**Description**

To customize the output of a model of class `lm`, you can define a method called `tidy_custom.lm` which returns a data.frame with a column called "term", and the other columns you want to use as "estimate" or "statistic" in your `modelsummary()` call. The output of this method must be similar to the result of `tidy(model)`.

**Usage**

```r
tidy_custom(x)
```

**Arguments**

- `x` An object to be converted into a tidy data.frame or tibble.

**Value**

A data.frame or tibble with information about model components.

---

**Var**

datasummary statistic shortcut

**Description**

datasummary statistic shortcut

**Usage**

```r
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**

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

* datasets
  - gof_map, 18
  - coef_rename, 3
  - datasummary, 3
  - datasummary_balance, 6
  - datasummary_correlation, 8
  - datasummary_correlation_format, 10
  - datasummary_df, 11
  - datasummary_skim, 13
  - dvnames, 14
  - get_estimates, 15
  - get_gof, 16
  - glance.modelsummary_list, 17
  - glance_custom, 17
  - gof_map, 18
  - Histogram, 18
  - Max, 19
  - Mean, 19
  - Median, 20
  - Min, 20
  - modelplot, 21
  - modelsummary, 24
  - modelsummary_wide, 29
  - N, 32
  - Ncol, 33
  - NPercent, 33
  - NUnique, 33
  - P0, 34
  - P100, 34
  - P25, 35
  - P50, 36
  - P75, 36
  - PercentMissing, 37
  - SD, 37
  - supported_models, 38
  - tidy.modelsummary_list, 38
  - tidy_custom, 39
  - Var, 39