Package 'modelsummary'

February 13, 2026

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. Details can be found in Arel-Bundock (2022) <doi:10.18637/jss.v103.i01>.

Version 2.6.0

URL https://modelsummary.com

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

Depends R (>= 4.1.0)

Imports checkmate (>= 2.3.1), data.table (>= 1.17.8), generics, glue, insight (>= 1.4.0), methods, parameters (>= 0.28.0), performance (>= 0.15.0), tables (>= 0.9.31), tinytable (>= 0.16.0)

Suggests AER, altdoc, Amelia, betareg, bookdown, brms, broom, broom.mixed, car, clubSandwich, correlation, covr, did, digest, DT, estimatr, fixest, flextable, future, future.apply, gamlss, ggdist, ggplot2, glmmTMB, gh, gt (>= 0.8.0), gtExtras, haven, huxtable, labelled, IRdisplay, ivreg, kableExtra, knitr, lavaan, lfe, lme4, lmtest, magick, magrittr, marginaleffects, MASS, mgcv, mice, nlme, nnet, officer, openxlsx, pandoc, parallel, pscl, psych, randomizr, Rdatasets, remotes, rmarkdown, rstanarm, rsvg, sandwich, spelling, survey, survival, tibble, tictoc, tidyselect, tidyverse, tinysnapshot (>= 0.2.0), tinytest, tinytex, webshot2, wesanderson

License GPL-3

Encoding UTF-8

Config/testthat/edition 3

Language en-US

RoxygenNote 7.3.3

Collate 'bind_est_gof.R' 'coef_rename.R' 'config_modelsummary.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_DT.R' 'factory_dataframe.R'
'factory_flextable.R' 'factory_gt.R' 'factory_huxtable.R'
'factory_kableExtra.R' 'factory_markdown.R'
'factory_tinytable.R' 'factory_typst.R' 'fmt_factory.R'
'format_estimates.R' 'format_gof.R' 'format_msg.R'
'get_estimates.R' 'get_gof.R' 'get_vcov.R' 'glance_custom.R'
'gof_map.R' 'hush.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_cbind.R' 'modelsummary_list.R'
'modelsummary_rbind.R' 'poorman.R' 'reexport.R'
'rename_statistics.R' 'sanitize_conf_level.R' 'sanitize_fmt.R'
'sanitize_gof_map.R' 'sanitize_models.R' 'sanitize_output.R'
'sanitize_shape.R' 'sanitize_statistic.R' 'sanitize_vcov.R'
'sanity_checks.R' 'settings.R' 'shape_estimates.R' 'span.R'
'stars.R' 'supported_models.R' 'themes.R' 'tidy_custom.R'
'update_modelsummary.R' 'utils_labels.R' 'utils_pad.R'
'utils_print.R' 'utils_replace.R' 'utils_stats.R'
'utils_warn.R' 'zzz.R'

NeedsCompilation no

Author Vincent Arel-Bundock [aut, cre] (ORCID:
<https://orcid.org/0000-0003-2042-7063>),
Joachim Gassen [ctb] (ORCID: <https://orcid.org/0000-0003-4364-2911>),
Nathan Eastwood [ctb],
Nick Huntington-Klein [ctb] (ORCID:
<https://orcid.org/0000-0002-7352-3991>),
Moritz Schwarz [ctb] (ORCID: <https://orcid.org/0000-0003-0340-3780>),
Benjamin Elbers [ctb] (ORCID: <https://orcid.org/0000-0001-5392-3448>),
Grant McDermott [ctb] (ORCID: <https://orcid.org/0000-0001-7883-8573>),
Lukas Wallrich [ctb] (ORCID: <https://orcid.org/0000-0003-2121-5177>)

Maintainer Vincent Arel-Bundock <vincent.arel-bundock@umontreal.ca>

Repository CRAN

Date/Publication 2026-02-13 12:20:02 UTC

Contents

`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,
  poly = 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
poly	boolean remove the "poly()" label and function arguments
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

```
library(modelsummary)
dat <- mtcars
dat$horse_power <- dat$hp
mod <- lm(mpg ~ horse_power + factor(cyl), dat)
modelsummary(mod, coef_rename = coef_rename)
```

---

config_modelsummary *Persistent user settings for the* modelsummary *package*

Description

Persistent user settings for the modelsummary package

Usage

```
config_modelsummary(
  factory_default,
  factory_latex,
  factory_html,
  factory_markdown,
  startup_message,
  reset = FALSE
)
```

Arguments

factory_default	Default output format: "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
factory_latex	Name of package used to generate LaTeX output when output="latex".
factory_html	Name of package used to generate LaTeX output when output="html".

`factory_markdown`
: Name of package used to generate LaTeX output when `output="markdown"`.

`startup_message`
: TRUE or FALSE to show warnings at startup

`reset`
: TRUE to return to default settings.

`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.

`modelsummary` also supplies several shortcut summary functions which can be used in `datasummary()` formulas: 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://modelsummary.com/
- https://modelsummary.com/vignettes/datasummary.html

Usage

```
datasummary(
  formula,
  data,
  output = getOption("modelsummary_output", default = "default"),
  fmt = 2,
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  align = getOption("modelsummary_align", default = NULL),
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  sparse_header = getOption("modelsummary_sparse_header", default = TRUE),
  escape = getOption("modelsummary_escape", default = 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. Grouping/nesting variables can appear on both sides of the formula, but all summary functions must be on one side.

`data`
: A data.frame (or tibble)

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_identity()`: unformatted raw values

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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. `TRUE` escapes all cells, captions, and notes. Users can have more fine-grained control by setting `escape=FALSE` and using an external command such as: `modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)`

... — all other arguments are passed through to the table-making functions tinytable::tt, kableExtra::kbl, gt::gt, DT::datatable, etc. depending on the `output` argument. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes.

Details

Visit the ’modelsummary’ website for more usage examples: https://modelsummary.com

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: tinytable, kableExtra, gt, html, rtf, and LaTeX. When saving tables to other formats, nested labels will be combined to a "flat" header.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, `modelsummary` uses `tinytable` as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to `kableExtra` for one session:

```
options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex
= 'kableExtra') options(modelsummary_factory_html = 'kableExtra')
```

Global Options

The behavior of `modelsummary` can be modified by setting global options. In particular, most of the arguments for most of the package’s functions cna be set using global options. For example:

- `options(modelsummary_output = "modelsummary_list")`
- `options(modelsummary_statistic = '({conf.low}, {conf.high})')`

- `options(modelsummary_stars = TRUE)`

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in `datasummary_balance()`:

- `options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))`

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a `weights` column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- `options(modelsummary_model_labels = "roman")`
- `options(modelsummary_model_labels_term = "Parameter")`
- `options(modelsummary_model_labels_group = "Group")`
- `options(modelsummary_model_labels_model = "Model")`

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "easystats")`

`options(modelsummary_get = "broom")`

`options(modelsummary_get = "all")`

The "all" option (default) means easystats then broom.

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")
```

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

Examples

```
library(modelsummary)

# 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 = 'tinytable')
#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)
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

`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://modelsummary.com/
- https://modelsummary.com/vignettes/datasummary.html

Usage

```
datasummary_balance(
  formula,
  data,
  output = getOption("modelsummary_output", default = "default"),
  fmt = fmt_decimal(digits = 1, pdigits = 3),
```

```
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  align = getOption("modelsummary_align", default = NULL),
  stars = getOption("modelsummary_stars", default = FALSE),
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  dinm = getOption("modelsummary_dinm", default = TRUE),
 dinm_statistic = getOption("modelsummary_dinm_statistic", default = "std.error"),
  fn = getOption("modelsummary_balance_fn", default = NULL),
  escape = getOption("modelsummary_escape", default = TRUE),
  ...
)
```

Arguments

`formula`

- `~1`: show summary statistics for the full dataset
- one-sided formula: with the "condition" or "column" variable on the right-hand side.
- two-side formula: with the subset of variables to summarize on the left-hand side and the condition 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. If there is a `weights` column, the reported mean and standard errors will also be weighted.

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits

- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_identity()`: unformatted raw values

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

`stars` to indicate statistical significance

- FALSE (default): no significance stars.
- TRUE: `c("+" = .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}`.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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"

`fn` Named list of functions used to summarize numeric variables. Each function should accept a numeric vector and return a single value. Defaults to mean and standard deviation with `na.rm = TRUE`. If a `weights` column is present and `fn` is supplied, weights are ignored with a warning.

escape — boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. TRUE escapes all cells, captions, and notes. Users can have more fine-grained control by setting escape=FALSE and using an external command such as: modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)

... — all other arguments are passed through to the table-making functions tinytable::tt, kableExtra::kbl, gt::gt, DT::datatable, etc. depending on the output argument. This allows users to pass arguments directly to datasummary in order to affect the behavior of other functions behind the scenes.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, modelsummary uses tinytable as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to kableExtra for one session:

```
options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex
= 'kableExtra') options(modelsummary_factory_html = 'kableExtra')
```

Global Options

The behavior of modelsummary can be modified by setting global options. In particular, most of the arguments for most of the package's functions cna be set using global options. For example:

- options(modelsummary_output = "modelsummary_list")
- options(modelsummary_statistic = '({conf.low}, {conf.high})')
- options(modelsummary_stars = TRUE)

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in datasummary_balance():

- options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a weights column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- options(modelsummary_model_labels = "roman")
- options(modelsummary_model_labels_term = "Parameter")
- options(modelsummary_model_labels_group = "Group")
- options(modelsummary_model_labels_model = "Model")

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "easystats")`

`options(modelsummary_get = "broom")`

`options(modelsummary_get = "all")`

The "all" option (default) means easystats then broom.

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")`

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
```

```
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

Examples

```
library(modelsummary)
datasummary_balance(~am, mtcars)

# custom summary functions
fun <- list(
  "Median" = function(x) median(x, na.rm = TRUE),
  "Variance" = function(x) var(x, na.rm = TRUE)
)
datasummary_balance(~am, mtcars, fn = fun)
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

`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://modelsummary.com/
- https://modelsummary.com/vignettes/datasummary.html

Usage

```
datasummary_correlation(
  data,
  output = getOption("modelsummary_output", default = "default"),
  method = getOption("modelsummary_method", default = "pearson"),
  fmt = 2,
  align = getOption("modelsummary_align", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
```

```
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  escape = getOption("modelsummary_escape", default = TRUE),
  stars = getOption("modelsummary_stars", default = FALSE),
  ...
)
```

Arguments

`data` A data.frame (or tibble)

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)

- fmt = fmt_sprintf("%.3f"): See ?sprintf
- fmt = fmt_identity(): unformatted raw values

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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

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. Positions can be defined using integers. In the modelsummary() function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- attr(new_rows, 1:2)
- attr(new_rows, "gof_start") See Examples section below.

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

title — string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as \\label{tab:mytable} directly to the title string, while also specifying escape=FALSE.

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. TRUE escapes all cells, captions, and notes. Users can have more fine-grained control by setting escape=FALSE and using an external command such as: modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)

stars — to indicate statistical significance

- FALSE (default): no significance stars.
- TRUE: c("+" = .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}.

... — other parameters are passed through to the table-making packages.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, modelsummary uses tinytable as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to kableExtra for one session:

options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex = 'kableExtra') options(modelsummary_factory_html = 'kableExtra')

Global Options

The behavior of `modelsummary` can be modified by setting global options. In particular, most of the arguments for most of the package's functions cna be set using global options. For example:

- `options(modelsummary_output = "modelsummary_list")`
- `options(modelsummary_statistic = '({conf.low}, {conf.high})')`
- `options(modelsummary_stars = TRUE)`

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in `datasummary_balance()`:

- `options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))`

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a `weights` column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- `options(modelsummary_model_labels = "roman")`
- `options(modelsummary_model_labels_term = "Parameter")`
- `options(modelsummary_model_labels_group = "Group")`
- `options(modelsummary_model_labels_model = "Model")`

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

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

The "all" option (default) means easystats then broom.

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")
```

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

Examples

```
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)

# `correlation` package objects
if (requireNamespace("correlation", quietly = TRUE)) {
  co <- correlation::correlation(mtcars[, 1:4])
  datasummary_correlation(co)

  # add stars to easycorrelation objects
  datasummary_correlation(co, stars = TRUE)
}

# 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)
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

`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,
  stars = FALSE
)
```

Arguments

x	square numeric matrix
fmt	how to format numeric values: integer, user-supplied function, or `modelsummary` function. • Integer: Number of decimal digits • User-supplied functions: – Any function which accepts a numeric vector and returns a character vector of the same length. • `modelsummary` functions: – `fmt = fmt_significant(2)`: Two significant digits (at the term-level) – `fmt = fmt_sprintf("%.3f")`: See `?sprintf` – `fmt = fmt_identity()`: unformatted raw values
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 ".").
stars	to indicate statistical significance • FALSE (default): no significance stars. • TRUE: `c("+" = .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}`.

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://modelsummary.com/
- https://modelsummary.com/vignettes/datasummary.html

Usage

```
datasummary_crosstab(
  formula,
  statistic = 1 ~ 1 + N + Percent("row"),
  data,
  output = getOption("modelsummary_output", default = "default"),
  fmt = 1,
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  align = getOption("modelsummary_align", default = NULL),
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  sparse_header = getOption("modelsummary_sparse_header", default = TRUE),
  escape = getOption("modelsummary_escape", default = 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) • Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg

- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_identity()`: unformatted raw values

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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. `TRUE` escapes all cells, captions, and notes. Users can have more fine-grained control by setting `escape=FALSE` and using an external command such as: `modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)`

`...` all other arguments are passed through to the table-making functions tinytable::tt, kableExtra::kbl, gt::gt, DT::datatable, etc. depending on the `output` argument. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes.

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()`.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, `modelsummary` uses `tinytable` as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to `kableExtra` for one session:

`options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex = 'kableExtra') options(modelsummary_factory_html = 'kableExtra')`

Global Options

The behavior of `modelsummary` can be modified by setting global options. In particular, most of the arguments for most of the package's functions cna be set using global options. For example:

- `options(modelsummary_output = "modelsummary_list")`
- `options(modelsummary_statistic = '({conf.low}, {conf.high})')`
- `options(modelsummary_stars = TRUE)`

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in `datasummary_balance()`:

- `options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))`

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a `weights` column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- `options(modelsummary_model_labels = "roman")`
- `options(modelsummary_model_labels_term = "Parameter")`
- `options(modelsummary_model_labels_group = "Group")`
- `options(modelsummary_model_labels_model = "Model")`

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "easystats")`

`options(modelsummary_get = "broom")`

`options(modelsummary_get = "all")`

The "all" option (default) means easystats then broom.

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")
```

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

Examples

```
library(modelsummary)

# 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 = ~ Percent("col"), data = mtcars)
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

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

Description

Draw a table from a data.frame

Usage

```
datasummary_df(
  data,
  output = getOption("modelsummary_output", default = "default"),
  fmt = 2,
  align = getOption("modelsummary_align", default = NULL),
  hrule = getOption("modelsummary_hrule", default = NULL),
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  escape = getOption("modelsummary_escape", default = TRUE),
  ...
)
```

Arguments

`data` A data.frame (or tibble)

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_identity()`: unformatted raw values

`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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

`hrule` position of horizontal rules (integer vector)

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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. `TRUE` escapes all cells, captions, and notes. Users can have more fine-grained control by setting `escape=FALSE` and using an external command such as: `modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)`

`...` all other arguments are passed through to the table-making functions tinytable::tt, kableExtra::kbl, gt::gt, DT::datatable, etc. depending on the `output` argument. This allows users to pass arguments directly to `datasummary` in order to affect the behavior of other functions behind the scenes.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, `modelsummary` uses `tinytable` as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to `kableExtra` for one session:

```
options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex
= 'kableExtra') options(modelsummary_factory_html = 'kableExtra')
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

---

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

Usage

```
datasummary_skim(
  data,
  output = getOption("modelsummary_output", default = "default"),
  type = getOption("modelsummary_type", default = "all"),
  fmt = 1,
  title = getOption("modelsummary_title", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  align = getOption("modelsummary_align", default = NULL),
  escape = getOption("modelsummary_escape", default = TRUE),
  by = getOption("modelsummary_by", default = NULL),
 fun_numeric = getOption("modelsummary_fun_numeric", default = list(Unique = NUnique,
  `Missing Pct.` = PercentMissing, Mean = Mean, SD = SD, Min = Min, Median = Median,
   Max = Max, Histogram = function(x) "")),
  ...
)
```

Arguments

`data`	A data.frame (or tibble)
`output`	filename or object type (character string) • Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg

- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`
- 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.

`type` String. Variables to summarize: "all", "numeric", "categorical", "dataset"

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_identity()`: unformatted raw values

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. `TRUE` escapes all cells, captions, and notes. Users can have more fine-grained control by setting `escape=FALSE` and using an external command such as: `modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)`

by	Character vector of grouping variables to compute statistics over.
fun_numeric	Named list of funtions to apply to each numeric column of data. If fun_numeric includes "Histogram" or "Density", inline plots are inserted. This argument is only used when type="numeric" or "all".
...	all other arguments are passed through to the table-making functions tinytable::tt, kableExtra::kbl, gt::gt, DT::datatable, etc. depending on the output argument. This allows users to pass arguments directly to datasummary in order to affect the behavior of other functions behind the scenes.

Details

modelsummary website:

- https://modelsummary.com/
- https://modelsummary.com/vignettes/datasummary.html

Note that the default escape=TRUE may not be best for all formats, such as LaTeX/PDF.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, modelsummary uses tinytable as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to kableExtra for one session:

```
options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex = 'kableExtra') options(modelsummary_factory_html = 'kableExtra')
```

Global Options

The behavior of modelsummary can be modified by setting global options. In particular, most of the arguments for most of the package's functions cna be set using global options. For example:

- `options(modelsummary_output = "modelsummary_list")`
- `options(modelsummary_statistic = '({conf.low}, {conf.high})')`
- `options(modelsummary_stars = TRUE)`

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in datasummary_balance():

- `options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))`

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a weights column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- `options(modelsummary_model_labels = "roman")`
- `options(modelsummary_model_labels_term = "Parameter")`

- `options(modelsummary_model_labels_group = "Group")`
- `options(modelsummary_model_labels_model = "Model")`

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "easystats")`

`options(modelsummary_get = "broom")`

`options(modelsummary_get = "all")`

The "all" option (default) means easystats then broom.

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")`

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

Examples

```
dat <- mtcars
dat$vs <- as.logical(dat$vs)
dat$cyl <- as.factor(dat$cyl)
datasummary_skim(dat)
datasummary_skim(dat, type = "categorical")
```

---

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. If the dependent variables are labelled, the labels are used as names. Pass your list of models to `dvnames` before sending to `modelsummary` to automatically get dependent variable-titled columns.

Usage

```
dvnames(models, number = FALSE, strip = 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?
strip	Boolean toggle for how dependent variable names are extracted. When strip is TRUE, the function calls insight::find_response(), typically returning the raw variable name stripped of any transformation notation (e.g., log(y)). When strip is FALSE, it first tries to parse the formula stored in the model object and return the characters before the first ~, keeping any transformation text; it falls back to insight::find_response() if the formula cannot be parsed. In both cases, if insight::find_response() and labels fail, the name is replaced by fill.
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 (1) and (2)
modelsummary(list(m1, m2))

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

---

fmt_decimal *Rounding with decimal digits in the* fmt *argument*

Description

Rounding with decimal digits in the fmt argument

Usage

```
fmt_decimal(digits = 3, pdigits = NULL, ...)
```

Arguments

digits	Number of decimal digits to keep, including trailing zeros.
pdigits	Number of decimal digits to keep for p values. If NULL, the value of digits is used.
...	Additional arguments are passed to the format() function (e.g., big.marks, scientific). See ?format

`fmt_equivalence` *Rounding with number of digits determined by an equivalence test*

Description

This function implements the suggestions of Astier & Wolak for the number of decimal digits to keep for coefficient estimates. The other statistics are rounded by `fmt_significant()`.

Usage

```
fmt_equivalence(conf_level = 0.95, digits = 3, pdigits = NULL, ...)
```

Arguments

`conf_level`	Confidence level to use for the equivalence test (1 - alpha).
`digits`	Number of significant digits to keep.
`pdigits`	Number of decimal digits to keep for p values. If `NULL`, the value of `digits` is used.
`...`	Additional arguments are passed to the `format()` function (e.g., `big.marks`, `scientific`). See `?format`

References

Astier, Nicolas, and Frank A. Wolak. Credible Numbers: A Procedure for Reporting Statistical Precision in Parameter Estimates. No. w32124. National Bureau of Economic Research, 2024.

Examples

```
library(modelsummary)
mod <- lm(mpg ~ hp, mtcars)

# Default equivalence-based formatting
modelsummary(mod, fmt = fmt_equivalence())

# alpha = 0.2
modelsummary(mod, fmt = fmt_equivalence(conf_level = .8))

# default equivalence, but with alternative significant digits for other statistics
modelsummary(mod, fmt = fmt_equivalence(digits = 5))
```

fmt_sci *Rounding using scientific notation*

Description

Rounding using scientific notation

Usage

```
fmt_sci(digits = 3, ...)
```

Arguments

digits	a positive integer indicating how many significant digits are to be used for numeric and complex x.
...	additional arguments passed to format().

fmt_significant *Rounding with significant digits in the* fmt *argument*

Description

The number of decimal digits to keep after the decimal is assessed

Usage

```
fmt_significant(digits = 3, ...)
```

Arguments

digits	Number of significant digits to keep.
...	Additional arguments are passed to the format() function (e.g., big.marks, scientific). See ?format

fmt_sprintf *Rounding with the* sprintf() *function in the* fmt *argument.*

Description

Rounding with the sprintf() function in the fmt argument.

Usage

```
fmt_sprintf(fmt)
```

Arguments

fmt	A string to control sprintf(), such as "%.3f" to keep 3 decimal digits. See ?sprintf.

Value

A function that takes an object and returns a string with the object formatted using sprintf.

fmt_statistic *Rounding with decimal digits on a per-statistic basis in the* fmt *argument for* modelsummary()

Description

Rounding with decimal digits on a per-statistic basis in the fmt argument for modelsummary()

Usage

```
fmt_statistic(..., default = 3)
```

Arguments

...	Statistic names and fmt value
default	Number of decimal digits to keep for unspecified terms

fmt_term — *Rounding with decimal digits on a per-term basis in the* `fmt` *argument for* `modelsummary()`

Description

Rounding with decimal digits on a per-term basis in the `fmt` argument for `modelsummary()`

Usage

```
fmt_term(..., default = 3)
```

Arguments

`...`	Term names and `fmt` value
`default`	Number of decimal digits to keep for unspecified terms

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

Description

A unified approach to extract results from a wide variety of models.

Usage

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

Arguments

`model`	a single model object
`conf_level`	numeric value between 0 and 1. confidence level to use for confidence intervals. Setting this argument to `NULL` does not extract confidence intervals, which can be faster for some models.
`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. "BS", "bootstrap", "residual", "mammen", "webb", "xy", "wild" use the `sandwich::vcovBS()`. 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.

shape — `NULL`, formula, or string which determines the shape of a table.

- `NULL`: Default shape with terms in rows and models in columns.
- Formula: The left side determines what appears on rows, and the right side determines what appears on columns. The formula can include one or more group identifier(s) to display related terms together, which can be useful for models with multivariate outcomes or grouped coefficients (See examples section below). The group identifier(s) must be column names produced by: `get_estimates(model)`. The group identifier(s) can be combined with the term identifier in a single column by using the colon to represent an interaction. If an incomplete formula is supplied (e.g., `~statistic`), `modelsummary` tries to complete it automatically. Goodness-of-fit statistics are only appended to the bottom of the table when `model` is on the right hand side of the formula (i.e., columns). 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 and group id in separate columns
  - `term : response + statistic ~ model`: term and group id in a single column
  - `term ~ response`
- String: "cbind", "rbind", "rcollapse"

- "cbind": side-by-side models with autmoatic spanning column headers to group models (`tinytable` only feature).
- "rbind" or "rcollapse": "panels" or "stacks" of regression models.
- the `models` argument must be a (potentially named) nested list of models.
- Unnamed nested list with 2 panels: `list(list(model1, model2), list(model3, model4))`
- Named nested list with 2 panels: `list("Panel A" = list(model1, model2), "Panel B" = list(model3, model4))`
- Named panels and named models: `list("Panel A" = list("(I)" = model1, "(II)" = model2), "Panel B" = list("(I)" = model3, "(II)" = model4))`
- "rbind": Bind the rows of independent regression tables
- "rcollapse": Bind the rows of regression tables and create a panel at the bottom where we "collapse" goodness-of-fit statistics which are identical across models.

`coef_rename` logical, named or unnamed character vector, or function

- Logical: TRUE renames variables based on the "label" attribute of each column. See the Example section below. Note: renaming is done by the `parameters` package at the extraction stage, before other arguments are applied like `coef_omit`. Therefore, this only works for models with builtin support and not for custom models.
- Unnamed character vector of length equal to the number of coefficients in the final table, after `coef_omit` is applied.
- 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)`

`...` all other arguments are passed through to three functions. See the documentation of these functions for lists of available arguments.

- parameters::model_parameters extracts parameter estimates. Available arguments depend on model type, but include:
  - `standardize`, `include_reference`, `centrality`, `dispersion`, `test`, `ci_method`, `prior`, `diagnostic`, `rope_range`, `power`, `cluster`, etc.
- performance::model_performance extracts goodness-of-fit statistics. Available arguments depend on model type, but include:
  - `metrics`, `estimator`, etc.
- tinytable::tt, kableExtra::kbl or gt::gt draw tables, depending on the value of the `output` argument. For example, by default `modelsummary` creates tables with tinytable::tt, which accepts a `width` and `theme` arguments.

Value

A dataframe with model estimates. The backend attribute of the returned object contains the backend that was used to extract parameters from the model. Moreover, 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))`.

`get_gof` *Extract goodness-of-fit statistics a tidy format.*

Description

A unified approach to extract results from a wide variety of models.

Usage

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

Arguments

`model`	a single model object
`gof_function`	function which accepts a model object in the `model` argument and returns a 1-row `data.frame` with one custom goodness-of-fit statistic per column.
`vcov_type`	string vcov type to add at the bottom of the table
`...`	all other arguments are passed through to three functions. See the documentation of these functions for lists of available arguments. • parameters::model_parameters extracts parameter estimates. Available arguments depend on model type, but include: – `standardize`, `include_reference`, `centrality`, `dispersion`, `test`, `ci_method`, `prior`, `diagnostic`, `rope_range`, `power`, `cluster`, etc. • performance::model_performance extracts goodness-of-fit statistics. Available arguments depend on model type, but include: – `metrics`, `estimator`, etc. • tinytable::tt, kableExtra::kbl or gt::gt draw tables, depending on the value of the `output` argument. For example, by default `modelsummary` creates tables with tinytable::tt, which accepts a `width` and `theme` arguments.

Value

A dataframe with the goodness-o-fit statistics. The backend attribute indicates the backend used to extract them. Moreover, for some models `get_gof` attaches useful attributes to the output. You can access this information by calling the `attributes` function `attributes(get_estimates(model))`.

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

```
if (identical(Sys.getenv("pkgdown"),  "true")) {

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)
}
```

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://modelsummary.com/

- modelplot Vignette.

Usage

```
modelplot(
  models,
  conf_level = getOption("modelsummary_conf_level", default = 0.95),
  coef_map = getOption("modelsummary_coef_map", default = NULL),
  coef_omit = getOption("modelsummary_coef_omit", default = NULL),
  coef_rename = getOption("modelsummary_coef_rename", default = NULL),
  vcov = getOption("modelsummary_vcov", default = NULL),
  exponentiate = getOption("modelsummary_exponentiate", default = FALSE),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  facet = getOption("modelsummary_facet", default = FALSE),
  draw = getOption("modelsummary_draw", default = TRUE),
  background = getOption("modelsummary_background", default = NULL),
  ...
)
```

Arguments

`models` a model, (named) list of models, or nested list of models.

- Single model: `modelsummary(model)`
- Unnamed list of models: `modelsummary(list(model1, model2))`
  - Models are labelled automatically. The default label style can be altered by setting a global option. See below.
- Named list of models: `modelsummary(list("A"=model1, "B"=model2))`
  - Models are labelled using the list names.
- Nested list of models:
  - When using the `shape` argument with "rbind", "rcollapse", or "cbind" values, `models` can be a nested list of models to display "panels" or "stacks" of regression models. See the `shape` argument documentation and examples below.

`conf_level` numeric value between 0 and 1. confidence level to use for confidence intervals. Setting this argument to `NULL` does not extract confidence intervals, which can be faster for some models.

`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")`. If `coef_map` is an unnamed vector, its values must be raw variable names if `coef_rename=FALSE` and variable labels if `coef_rename=TRUE`. See `modelsummary::get_estimates` to get the coefficient out of a model. See Examples section below.

`coef_omit` integer vector or regular expression to identify which coefficients to omit (or keep) from the table. Positive integers determine which coefficients to omit. Negative integers determine which coefficients to keep. A regular expression can be used to omit coefficients, and perl-compatible "negative lookaheads" can be used to specify which coefficients to *keep* in the table. Examples:

- c(2, 3, 5): omits the second, third, and fifth coefficients.
- c(-2, -3, -5): negative values keep the second, third, and fifth coefficients.
- `"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` logical, named or unnamed character vector, or function

- Logical: TRUE renames variables based on the "label" attribute of each column. See the Example section below. Note: renaming is done by the `parameters` package at the extraction stage, before other arguments are applied like `coef_omit`. Therefore, this only works for models with builtin support and not for custom models.
- Unnamed character vector of length equal to the number of coefficients in the final table, after `coef_omit` is applied.
- 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)`

`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. "BS", "bootstrap", "residual", "mammen", "webb", "xy", "wild" use the `sandwich::vcovBS()`. 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`. The `exponentiate` argument is ignored for distributional random effects parameters (SD and Cor) and dispersions parameters.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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 three functions. See the documentation of these functions for lists of available arguments.

- parameters::model_parameters extracts parameter estimates. Available arguments depend on model type, but include:
  - `standardize`, `include_reference`, `centrality`, `dispersion`, `test`, `ci_method`, `prior`, `diagnostic`, `rope_range`, `power`, `cluster`, etc.
- performance::model_performance extracts goodness-of-fit statistics. Available arguments depend on model type, but include:
  - `metrics`, `estimator`, etc.
- tinytable::tt, kableExtra::kbl or gt::gt draw tables, depending on the value of the `output` argument. For example, by default `modelsummary` creates tables with tinytable::tt, which accepts a `width` and `theme` arguments.

Examples

```
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 = "(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)
```

```
# logistic regression example
df <- as.data.frame(Titanic)
mod_titanic <- glm(
  Survived ~ Class + Sex,
  family = binomial,
  weight = Freq,
  data = df
)

# displaying odds ratio using a log scale
modelplot(mod_titanic, exponentiate = TRUE) +
  scale_x_log10() +
  xlab("Odds Ratios and 95% confidence intervals")
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

`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, Typst, 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: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, `DT`. For more information, see the Details and Examples sections below, and the vignettes on the `modelsummary` website: https://modelsummary.com/

- 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 = getOption("modelsummary_output", default = "default"),
  fmt = getOption("modelsummary_fmt", default = 3),
  estimate = getOption("modelsummary_estimate", default = "estimate"),
  statistic = getOption("modelsummary_statistic", default = "std.error"),
  vcov = getOption("modelsummary_vcov", default = NULL),
  conf_level = getOption("modelsummary_conf_level", default = 0.95),
  exponentiate = getOption("modelsummary_exponentiate", default = FALSE),
```

```
  stars = getOption("modelsummary_stars", default = FALSE),
  shape = getOption("modelsummary_shape", default = term + statistic ~ model),
  coef_map = getOption("modelsummary_coef_map", default = NULL),
  coef_omit = getOption("modelsummary_coef_omit", default = NULL),
  coef_rename = getOption("modelsummary_coef_rename", default = FALSE),
  gof_map = getOption("modelsummary_gof_map", default = NULL),
  gof_omit = getOption("modelsummary_gof_omit", default = NULL),
  gof_function = getOption("modelsummary_gof_function", default = NULL),
  group_map = getOption("modelsummary_group_map", default = NULL),
  add_columns = getOption("modelsummary_add_columns", default = NULL),
  add_rows = getOption("modelsummary_add_rows", default = NULL),
  align = getOption("modelsummary_align", default = NULL),
  notes = getOption("modelsummary_notes", default = NULL),
  title = getOption("modelsummary_title", default = NULL),
  escape = getOption("modelsummary_escape", default = TRUE),
  ...
)
```

Arguments

`models` a model, (named) list of models, or nested list of models.

- Single model: `modelsummary(model)`
- Unnamed list of models: `modelsummary(list(model1, model2))`
  - Models are labelled automatically. The default label style can be altered by setting a global option. See below.
- Named list of models: `modelsummary(list("A"=model1, "B"=model2))`
  - Models are labelled using the list names.
- Nested list of models:
  - When using the `shape` argument with "rbind", "rcollapse", or "cbind" values, `models` can be a nested list of models to display "panels" or "stacks" of regression models. See the `shape` argument documentation and examples below.

`output` filename or object type (character string)

- Supported filename extensions: .docx, .html, .tex, .md, .txt, .csv, .xlsx, .png, .jpg
- Supported object types: "default", "html", "markdown", "latex", "latex_tabular", "typst", "data.frame", "tinytable", "gt", "kableExtra", "huxtable", "flextable", "DT", "jupyter". The "modelsummary_list" value produces a lightweight object which can be saved and fed back to the `modelsummary` function.
- The "default" output format can be set to "tinytable", "kableExtra", "gt", "flextable", "huxtable", "DT", or "markdown"
  - If the user does not choose a default value, the packages listed above are tried in sequence.
  - Session-specific configuration: `options("modelsummary_factory_default" = "gt")`
  - Persistent configuration: `config_modelsummary(output = "markdown")`

- 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.

`fmt` how to format numeric values: integer, user-supplied function, or `modelsummary` function.

- Integer: Number of decimal digits
- User-supplied functions:
  - Any function which accepts a numeric vector and returns a character vector of the same length.
- `modelsummary` functions:
  - `fmt = fmt_significant(2)`: Two significant digits (at the term-level)
  - `fmt = fmt_decimal(digits = 2, pdigits = 3)`: Decimal digits for estimate and p values
  - `fmt = fmt_sprintf("%.3f")`: See `?sprintf`
  - `fmt = fmt_term("(Intercept)" = 1, "X" = 2)`: Format terms differently
  - `fmt = fmt_statistic("estimate" = 1, "r.squared" = 6)`: Format statistics differently.
  - `fmt = fmt_identity()`: unformatted raw values
- string: Passing the string `s` is equivalent to passing `fmt_sprintf(s)`
- Note on 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}]"`
- Numbers are automatically rounded and converted to strings. To let glue apply functions to numeric values, users must set `fmt=NULL`. For more complex formatting, users are encouraged to use the `fmt` argument, which accepts custom functions.

`statistic` vector of strings or `glue` strings which select uncertainty statistics to report vertically below the estimate (ex: standard errors, confidence intervals, p values). 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}"`
- Notes:
  - The names of the `statistic` are used a column names when using the `shape` argument to display statistics as columns:
    * `statistic=c("p"="p.value", "["="conf.low", "]"="conf.high")`
  - Some statistics are not supported for all models. See column names in `get_estimates(model)`, and visit the website to learn how to add custom statistics.
  - 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. "BS", "bootstrap", "residual", "mammen", "webb", "xy", "wild" use the `sandwich::vcovBS()`. 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` numeric value between 0 and 1. confidence level to use for confidence intervals. Setting this argument to `NULL` does not extract confidence intervals, which can be faster for some models.

`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`. The `exponentiate` argument is ignored for distributional random effects parameters (SD and Cor) and dispersions parameters.

stars — to indicate statistical significance

- FALSE (default): no significance stars.
- TRUE: `c("+" = .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 — `NULL`, formula, or string which determines the shape of a table.

- `NULL`: Default shape with terms in rows and models in columns.
- Formula: The left side determines what appears on rows, and the right side determines what appears on columns. The formula can include one or more group identifier(s) to display related terms together, which can be useful for models with multivariate outcomes or grouped coefficients (See examples section below). The group identifier(s) must be column names produced by: `get_estimates(model)`. The group identifier(s) can be combined with the term identifier in a single column by using the colon to represent an interaction. If an incomplete formula is supplied (e.g., `~statistic`), `modelsummary` tries to complete it automatically. Goodness-of-fit statistics are only appended to the bottom of the table when `model` is on the right hand side of the formula (i.e., columns). 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 and group id in separate columns
  - `term : response + statistic ~ model`: term and group id in a single column
  - `term ~ response`
- String: "cbind", "rbind", "rcollapse"
  - "cbind": side-by-side models with autmoatic spanning column headers to group models (`tinytable` only feature).
  - "rbind" or "rcollapse": "panels" or "stacks" of regression models.
  - the `models` argument must be a (potentially named) nested list of models.
  - Unnamed nested list with 2 panels: `list(list(model1, model2), list(model3, model4))`
  - Named nested list with 2 panels: `list("Panel A" = list(model1, model2), "Panel B" = list(model3, model4))`
  - Named panels and named models: `list("Panel A" = list("(I)" = model1, "(II)" = model2), "Panel B" = list("(I)" = model3, "(II)" = model4))`
  - "rbind": Bind the rows of independent regression tables
  - "rcollapse": Bind the rows of regression tables and create a panel at the bottom where we "collapse" goodness-of-fit statistics which are identical across models.

`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")`. If `coef_map` is an unnamed vector, its values must be raw variable names if `coef_rename=FALSE` and variable labels if `coef_rename=TRUE`. See `modelsummary::get_estimates` to get the coefficient out of a model. See Examples section below.

`coef_omit` integer vector or regular expression to identify which coefficients to omit (or keep) from the table. Positive integers determine which coefficients to omit. Negative integers determine which coefficients to keep. A regular expression can be used to omit coefficients, and perl-compatible "negative lookaheads" can be used to specify which coefficients to *keep* in the table. Examples:

- c(2, 3, 5): omits the second, third, and fifth coefficients.
- c(-2, -3, -5): negative values keep the second, third, and fifth coefficients.
- `"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` logical, named or unnamed character vector, or function

- Logical: TRUE renames variables based on the "label" attribute of each column. See the Example section below. Note: renaming is done by the `parameters` package at the extraction stage, before other arguments are applied like `coef_omit`. Therefore, this only works for models with builtin support and not for custom models.
- Unnamed character vector of length equal to the number of coefficients in the final table, after `coef_omit` is applied.
- 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.

- character vector: "all", "none", or a vector of statistics such as `c("rmse", "nobs", "r.squared")`. Elements correspond to colnames in the data.frame produced by `get_gof(model)`. The `modelsummary::gof_map` default dictionary is used to format and rename statistics.
- NA: excludes all statistics from the bottom part of the table.
- data.frame with 3 columns named "raw", "clean", "fmt". Unknown statistics are omitted. See the 'Examples' section below. The `fmt` column in this data frame only accepts integers. For more flexibility, use a list of lists, as described in the next bullet.
- list of lists, each of which includes 3 elements named "raw", "clean", "fmt". Unknown statistics are omitted. The `fmt` element can be a string (`?fmt_sprintf`), numeric value (`?fmt_decimal`), or function which will be used to round/format the string in question. 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.

`gof_function` function which accepts a model object in the `model` argument and returns a 1-row `data.frame` with one custom goodness-of-fit statistic per column.

`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_columns` a data.frame (or tibble) with the same number of rows 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 columns positions. See Examples section below.

`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. Positions can be defined using integers. In the `modelsummary()` function (only), you can also use string shortcuts: "coef_start", "coef_end", "gof_start", "gof_end"

- `attr(new_rows, 1:2)`
- `attr(new_rows, "gof_start")` 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. For LaTeX/PDF output, this option requires at least version 3.0.25 of the siunitx LaTeX package. See the LaTeX preamble help section below for commands to insert in your LaTeX preamble.

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

`title` string. Cross-reference labels should be added with Quarto or Rmarkdown chunk options when applicable. When saving standalone LaTeX files, users can

add a label such as `\\label{tab:mytable}` directly to the title string, while also specifying `escape=FALSE`.

`escape` boolean TRUE escapes or substitutes LaTeX/HTML characters which could prevent the file from compiling/displaying. `TRUE` escapes all cells, captions, and notes. Users can have more fine-grained control by setting `escape=FALSE` and using an external command such as: `modelsummary(model, "latex") |> tinytable::format_tt(tab, j=1:5, escape=TRUE)`

`...` all other arguments are passed through to three functions. See the documentation of these functions for lists of available arguments.

- parameters::model_parameters extracts parameter estimates. Available arguments depend on model type, but include:
  - `standardize`, `include_reference`, `centrality`, `dispersion`, `test`, `ci_method`, `prior`, `diagnostic`, `rope_range`, `power`, `cluster`, etc.
- performance::model_performance extracts goodness-of-fit statistics. Available arguments depend on model type, but include:
  - `metrics`, `estimator`, etc.
- tinytable::tt, kableExtra::kbl or gt::gt draw tables, depending on the value of the `output` argument. For example, by default `modelsummary` creates tables with tinytable::tt, which accepts a `width` and `theme` arguments.

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:

- `tinytable`: set `output="tinytable"`, post-process your table, and use the `tinytable::save_tt` function.
- `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. The `backend` attribute includes the backend used to extract estimates and goodness-of-fit measure.

Version 2.0.0, kableExtra, and tinytable

Since version 2.0.0, `modelsummary` uses `tinytable` as its default table-drawing backend. Learn more at: https://vincentarelbundock.github.io/tinytable/",

Revert to `kableExtra` for one session:

```
options(modelsummary_factory_default = 'kableExtra') options(modelsummary_factory_latex
= 'kableExtra') options(modelsummary_factory_html = 'kableExtra')
```

Global Options

The behavior of `modelsummary` can be modified by setting global options. In particular, most of the arguments for most of the package's functions cna be set using global options. For example:

- `options(modelsummary_output = "modelsummary_list")`
- `options(modelsummary_statistic = '({conf.low}, {conf.high})')`
- `options(modelsummary_stars = TRUE)`

Options not specific to given arguments are listed below.

Balance tables: summary functions:

You can customize the summary statistics shown in `datasummary_balance()`:

- `options(modelsummary_balance_fn = list(Mean = Mean, "Std. Dev." = SD))`

The list must be named, and each function should accept a numeric vector and return a single value. When this option is set and a `weights` column is present, weights are ignored with a warning.

Model labels: default column names:

These global option changes the style of the default column headers:

- `options(modelsummary_model_labels = "roman")`
- `options(modelsummary_model_labels_term = "Parameter")`
- `options(modelsummary_model_labels_group = "Group")`
- `options(modelsummary_model_labels_model = "Model")`

The options above control the label shown in the stub columns for terms, groups, and model names. By default, those headers are blank. Setting any of these options to a character string inserts that label as the column header.

The supported styles are: "model", "arabic", "letters", "roman", "(arabic)", "(letters)", "(roman)"

Table-making packages:

`modelsummary` supports 6 table-making packages: `tinytable`, `kableExtra`, `gt`, `flextable`, `huxtable`, and `DT`. Some of these packages have overlapping functionalities. 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_word = 'huxtable')` `options(modelsummary_factory_png = 'gt')` `options(modelsummary_factory_latex = 'gt')` `options(modelsummary_factory_latex_tabular = 'kableExtra')`

Table themes:

Change the look of tables in an automated and replicable way, using the `modelsummary` theming functionality. See the vignette: https://modelsummary.com/vignettes/appearance.html

- `modelsummary_theme_gt`
- `modelsummary_theme_kableExtra`
- `modelsummary_theme_huxtable`
- `modelsummary_theme_flextable`
- `modelsummary_theme_dataframe`

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 case of failure. You can change the order of priorities or include goodness-of-fit extracted by *both* packages by setting:

`options(modelsummary_get = "easystats")`

`options(modelsummary_get = "broom")`

`options(modelsummary_get = "all")`

The "all" option (default) means easystats then broom.

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")`

LaTeX preamble

When creating LaTeX via the `tinytable` backend (default in version 2.0.0 and later), it is useful to include the following commands in the LaTeX preamble of your documents. These commands are automatically added to the preamble when compiling Rmarkdown or Quarto documents, except when the `modelsummary()` calls are cached.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
```

```
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

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. Users on mac or linux can launch parallel computation using the built-in `parallel` package. All they need to do is supply a `mc.cores` argument which will be pushed forward to the `parallel::mclapply` function:

```
modelsummary(model_list, mc.cores = 5)
```

All users can also use the `future.apply` package to parallelize model summaries. For example, to use 4 cores to extract results:

```
library(future.apply)
plan(multicore, workers = 4)
options("modelsummary_future" = TRUE)
modelsummary(model_list)
```

Note that the "multicore" plan only parallelizes under mac or linux. Windows users can use `plan(multisession)` instead. However, note that the first time `modelsummary()` is called under multisession can be a fair bit longer, because of extra costs in passing data to and loading required packages on to workers. Subsequent calls to `modelsummary()` will often be much faster.

Some users have reported difficult to reproduce errors when using the `future` package with some packages. The `future` parallelization in `modelsummary` can be disabled by calling:

```
options("modelsummary_future" = FALSE)
```

References

Arel-Bundock V (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software*, *103*(1), 1-23. doi:10.18637/jss.v103.i01.'

Examples

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

library(modelsummary)

# load data and estimate models
utils::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"))

# fmt
mod <- lm(mpg ~ hp + drat + qsec, data = mtcars)
```

```
modelsummary(mod, fmt = 3)

modelsummary(mod, fmt = fmt_significant(3))

modelsummary(mod, fmt = NULL)

modelsummary(mod, fmt = fmt_decimal(4))

modelsummary(mod, fmt = fmt_sprintf("%.5f"))

modelsummary(mod, fmt = fmt_statistic(estimate = 4, conf.int = 1), statistic = "conf.int")

modelsummary(mod, fmt = fmt_term(hp = 4, drat = 1, default = 2))

m <- lm(mpg ~ I(hp * 1000) + drat, data = mtcars)
f <- function(x) format(x, digits = 3, nsmall = 2, scientific = FALSE, trim = TRUE)
modelsummary(m, fmt = f, gof_map = NA)

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

modelsummary(models, coef_rename = toupper)

modelsummary(models, coef_rename = coef_rename)

# coef_rename = TRUE for variable labels
datlab <- mtcars
datlab$cyl <- factor(datlab$cyl)
attr(datlab$hp, "label") <- "Horsepower"
attr(datlab$cyl, "label") <- "Cylinders"
modlab <- lm(mpg ~ hp * drat + cyl, data = datlab)
modelsummary(modlab, coef_rename = TRUE)

# coef_rename: unnamed vector of length equal to the number of terms in the final table
m <- lm(hp ~ mpg + factor(cyl), data = mtcars)
modelsummary(m, coef_omit = -(3:4), coef_rename = c("Cyl 6", "Cyl 8"))

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

modelsummary(models, coef_map = c("Volume", "Height"))

# coef_omit: omit the first and second coefficients
modelsummary(models, coef_omit = 1:2)

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

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

# coef_omit: omit coefficients matching either one of two substring
```

```
# modelsummary(models, coef_omit = "ei|rc", gof_omit = ".*")

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

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

# shape: multinomial model
library(nnet)
multi <- multinom(factor(cyl) ~ mpg + hp, data = mtcars, trace = FALSE)

# shape: term names and group ids in rows, models in columns
modelsummary(multi, shape = response ~ model)

# shape: term names and group ids in rows in a single column
modelsummary(multi, shape = term:response ~ model)

# shape: term names in rows and group ids in columns
modelsummary(multi, shape = term ~ response:model)

# shape = "rcollapse"
panels <- list(
  "Panel A: MPG" = list(
    "A" = lm(mpg ~ hp, data = mtcars),
    "B" = lm(mpg ~ hp + factor(gear), data = mtcars)),
  "Panel B: Displacement" = list(
    "A" = lm(disp ~ hp, data = mtcars),
    "C" = lm(disp ~ hp + factor(gear), data = mtcars))
)

# shape = "cbind"
modelsummary(panels, shape = "cbind")

modelsummary(
  panels,
  shape = "rbind",
  gof_map = c("nobs", "r.squared"))

# 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}", escape = FALSE)

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

```
attr(rows, "position") <- "gof_start"
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: list of lists
f <- function(x) format(round(x, 3), big.mark = ",")
gm <- list(
  list("raw" = "nobs", "clean" = "N", "fmt" = f),
  list("raw" = "AIC", "clean" = "aic", "fmt" = f))
modelsummary(models, gof_map = gm)
```

`update_modelsummary` *Update* `modelsummary` *and its dependencies*

Description

Update `modelsummary` and its dependencies to the latest R-Universe or CRAN versions. The R session needs to be restarted after install.

Usage

```
update_modelsummary(source = "development")
```

Arguments

`source`	one of two strings: "development" or "cran"

Index