Package 'panelsummary'

March 20, 2026

Type Package

Title Create Publication-Ready Regression Tables with Panels

Version 0.1.3

Maintainer Michael Topper <miketopper123@gmail.com>

Description Create an automated regression table that is well-suited for models that are estimated with multiple dependent variables. 'panelsummary' extends 'modelsummary' (Arel-Bundock, V. (2022) <doi:10.18637/jss.v103.i01>) by allowing regression tables to be split into multiple sections with a simple function call. Utilize familiar arguments such as fmt, estimate, statistic, vcov, conf_level, stars, coef_map, coef_omit, coef_rename, gof_map, and gof_omit from 'modelsummary' to clean the table, and additionally, add a row for the mean of the dependent variable without external manipulation.

License GPL (>= 3)

URL https://github.com/michaeltopper1/panelsummary, https://michaeltopper1.github.io/panelsummary/

BugReports https://github.com/michaeltopper1/panelsummary/issues

Imports dplyr (>= 1.0.9), fixest (>= 0.10.4), kableExtra (>= 1.3.4), methods (>= 4.1.3), modelsummary (>= 1.3.0), rlang (>= 1.0.6), stringr (>= 1.4.1), tidyselect (>= 1.2.0)

Suggests covr, gt, knitr, parameters, performance, rmarkdown, testthat (>= 3.0.0), tibble (>= 3.1.8), utils (>= 4.1.3)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Michael Topper [aut, cre], Vincent Arel-Bundock [cph] (Some documentation is inherited from 'modelsummary')

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2026-03-20 08:50:02 UTC

Contents

`clean_raw` *Pass a panelsummary::panelsummary_raw dataframe into kableExtra::kbl() with typical defaults*

Description

'clean_raw' Passes a panelsummary::panelsummary_raw dataframe that has (or has not) been edited further into kableExtra::kbl() with default settings that look publication-ready. This includes changing the column names and fixing the alignment.

Usage

```
clean_raw(
  data.frame,
  alignment = NULL,
  colnames = NULL,
  format = NULL,
  caption = NULL,
  pretty_num = FALSE,
  booktabs = FALSE,
  linesep = if (booktabs) c("", "", "", "", "\\addlinespace") else "\\hline"
)
```

Arguments

`data.frame`	The data.frame (or tibble) from panelsummary::panelsummary_raw() that has been manipulated.
`alignment`	A character string. By default, it is set to left adjusting the first column, and centering the rest of the columns. For example, a model with three columns will have adjustment of "lcc".
`colnames`	An optional vector of strings. The vector of strings should have the same length as the number columns of the table. * 'NULL' (the default): colnames are defaulted to a whitespace, followed by (1), (2), ...etc.

format	A character string. Possible values are latex, html, pipe (Pandoc's pipe tables), simple (Pandoc's simple tables), and rst. The value of this argument will be automatically determined if the function is called within a knitr document. The format value can also be set in the global option knitr.table.format. If format is a function, it must return a character string.
caption	A string. The table caption.
pretty_num	A logical. If TRUE, then numbers over 999 have a comma printing format.
booktabs	T/F for whether to enable the booktabs format for tables. I personally would recommend you turn this on for every latex table except some special cases.
linesep	By default, in booktabs tables, kable insert an extra space every five rows for clear display. If you don't want this feature or if you want to do it in a different pattern, you can consider change this option. The default is c(", ", ", ", '\addlinespace'). Also, if you are not using booktabs, but you want a cleaner display, you can change this to ".

Value

A raw data frame that is ready for further manipulation.

Examples

```
## Cleaning a panelsummary_raw dataframe with clean_raw

ols_1 <- lm(mpg ~ hp + cyl, data = mtcars)

panelsummary_raw(ols_1, ols_1) |> clean_raw()
```

---

connect_means *Merges the means to the modelsummary output dataframe*

Description

'connect_means' connects the means to the modelsummary dataframe.

Usage

```
connect_means(panel_df, means)
```

Arguments

panel_df	The modelsummary dataframe supplied from modelsummary::modelsummary(x, output = "data.frame")
means	The list of named vectors of means which correspond to each model.

Value

A data.frame with the attached means.

`models_supported` *all models supported by panelsummary*

Description

all models supported by panelsummary

Usage

```
models_supported()
```

Value

a list of all modeltypes supported by panelsummary

Examples

```
models_supported()
```

`panelsummary` *Create a regression table with multiple panels*

Description

'panelsummary' Creates a beautiful and customizable regression table with panels. This function is best used to summarize multiple dependent variables that are passed through the same regression models. This function returns a kableExtra object which can then be edited using kableExtra's suite of functions.

Usage

```
panelsummary(
  ...,
  panel_labels = NULL,
  mean_dependent = FALSE,
  colnames = NULL,
  caption = NULL,
  format = NULL,
  pretty_num = FALSE,
  collapse_fe = FALSE,
  bold = FALSE,
```

```
  italic = FALSE,
  hline_after = FALSE,
  hline_before_fe = TRUE,
  fmt = 3,
  estimate = "estimate",
  statistic = "std.error",
  vcov = NULL,
  conf_level = 0.95,
  stars = FALSE,
  coef_map = NULL,
  coef_omit = NULL,
  coef_rename = NULL,
  gof_map = NULL,
  gof_omit = NULL
)
```

Arguments

`...`	A regression model or models (see panelsummary::models_supported for classes that are supported). * The regression model can be a list of models or a singular object. * If a list is passed in, one column for each list is created. Each argument will correspond to a panel. * If only one object is passed in, there will be no panels and the output will be similar to evaluating modelsummary::modelsummary() followed by kableExtra::kbl()
`panel_labels`	A character vector. How to label each panel in the table. * 'NULL' (the default): the panels will be labeled "Panel A:", "Panel B:",...etc.
`mean_dependent`	A boolean. For use with fixest objects only. * 'FALSE' (the default): the mean of the dependent variable will not be shown in the resulting table. * 'TRUE': the mean of the dependent variable will be shown in the resulting table.
`colnames`	An optional vector of strings. The vector of strings should have the same length as the number columns of the table. * 'NULL' (the default): colnames are defaulted to a whitespace, followed by (1), (2), ...etc.
`caption`	A string. The table caption.
`format`	A character string. Possible values are latex, html, pipe (Pandoc's pipe tables), simple (Pandoc's simple tables), and rst. The value of this argument will be automatically determined if the function is called within a knitr document. The format value can also be set in the global option knitr.table.format. If format is a function, it must return a character string.
`pretty_num`	A logical. If TRUE, then numbers over 999 have a comma printing format.
`collapse_fe`	A boolean. For use with fixest objects only. Determines whether fixed effects should only be included in the bottom of the table. This is suited for when each panel has the same models with the same fixed effects. * 'FALSE' (the default): fixed effects are shown in each panel. * 'TRUE': fixed effects are shown only at the bottom of the final panel, separated by a horizontal line (see hline_before_fe)
`bold`	A boolean. Determines whether the panel names should be in bold font. * 'FALSE' (the default): the panel names are not in bold. * 'TRUE': the panel names are bolded

`italic` A boolean. Determines whether the panel names should be in italics. * 'FALSE' (the default): the panel names are not in italics. * 'TRUE': the panel names will be in italics.

`hline_after` A boolean. Adds a horizontal line after the panel labels. * 'FALSE' (the default): there is not horizonal line after the panel labels. * 'TRUE': a horizontal line will appear after the panel labels.

`hline_before_fe`
A boolean. To be used only when collapse_fe = TRUE, and hence with fixest objects only. Adds a horizontal line before the fixed effects portion of the table.

`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.

`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}`.

`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.

Value

A kableExtra object that is instantly customizable by kableExtra's suite of functions.

Examples

```
# Panelsummary with lm -------------------------

reg_1 <- lm(mpg ~ hp + cyl, data = mtcars)
reg_2 <- lm(disp ~ hp + cyl, data = mtcars)

panelsummary(reg_1, reg_2, panel_labels = c("Panel A: MPG", "Panel B: Displacement"))


# Panelsummary with fixest -------------------------
## Not run:
ols_1 <- mtcars |> fixest::feols(mpg ~  cyl | gear + carb, cluster = ~hp, nthreads = 2)

panelsummary(ols_1, ols_1, mean_dependent = TRUE,
            panel_labels = c("Panel A:MPG", "Panel B: DISP"),
            caption = "The effect of cyl on MPG and DISP",
            italic = TRUE, stars = TRUE)


## Collapsing fixed effects (fixest-only)----------------
```

```
panelsummary(ols_1, ols_1, mean_dependent = TRUE,
             collapse_fe = TRUE, panel_labels = c("Panel A: MPG", "Panel B: DISP"),
             caption = "The effect of cyl on MPG and DISP",
             italic = TRUE, stars = TRUE)

## Including multiple models------------------


panelsummary(list(ols_1, ols_1, ols_1), ols_1,
              panel_labels = c("Panel A: MPG", "Panel B: DISP"),
               caption = "Multiple models",
               stars = TRUE)


## End(Not run)
```

---

panelsummary_raw *Create a regression data.frame to manually edit further*

Description

'panelsummary_raw' Creates a data.frame for further editing. The data.frame can be directly passed into kableExtra::kbl(), or alternatively, passed into panelsummary::clean_raw() to get typical defaults from kableExtra::kbl().

Usage

```
panelsummary_raw(
  ...,
  mean_dependent = FALSE,
  colnames = NULL,
  caption = NULL,
  format = NULL,
  fmt = 3,
  estimate = "estimate",
  statistic = "std.error",
  vcov = NULL,
  conf_level = 0.95,
  stars = FALSE,
  coef_map = NULL,
  coef_omit = NULL,
  coef_rename = NULL,
  gof_map = NULL,
  gof_omit = NULL
)
```

Arguments

`...` 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.

`mean_dependent` A boolean. For use with fixest objects only. * 'FALSE' (the default): the mean of the dependent variable will not be shown in the resulting table. * 'TRUE': the mean of the dependent variable will be shown in the resulting table.

`colnames` An optional vector of strings. The vector of strings should have the same length as the number columns of the table. * 'NULL' (the default): colnames are defaulted to a whitespace, followed by (1), (2), ...etc.

`caption` A string. The table caption.

`format` A character string. Possible values are latex, html, pipe (Pandoc's pipe tables), simple (Pandoc's simple tables), and rst. The value of this argument will be automatically determined if the function is called within a knitr document. The format value can also be set in the global option knitr.table.format. If format is a function, it must return a character string.

`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.

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}.

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.

Value

A kableExtra object that is instantly customizable by kableExtra's suite of functions.

Examples

```
## Using panelsummary_raw

ols_1 <- lm(mpg ~ hp + cyl, data = mtcars)

panelsummary_raw(ols_1, ols_1)


## Including multiple models------------------

panelsummary_raw(list(ols_1, ols_1, ols_1), ols_1,
              caption = "Multiple models",
              stars = TRUE)
```

Index