

# Package ‘gtsummary’

January 20, 2022

**Title** Presentation-Ready Data Summary and Analytic Result  
Tables

**Version** 1.5.1

**Description** Creates presentation-ready tables summarizing data sets, regression models, and more. The code to create the tables is concise and highly customizable. Data frames can be summarized with any function, e.g. `mean()`, `median()`, even user-written functions. Regression models are summarized and include the reference rows for categorical variables. Common regression models, such as logistic regression and Cox proportional hazards regression, are automatically identified and the tables are pre-filled with appropriate column headers.

**License** MIT + file LICENSE

**URL** <https://github.com/ddsjoberg/gtsummary>,  
<https://www.danielsjoberg.com/gtsummary/>

**BugReports** <https://github.com/ddsjoberg/gtsummary/issues>

**Depends** R (>= 3.4)

**Imports** broom (>= 0.7.11),  
broom.helpers (>= 1.6.0),  
cli (>= 2.3.0),  
dplyr (>= 1.0.3),  
forcats (>= 0.5.0),  
glue (>= 1.4.1),  
gt (>= 0.3.1),  
knitr (>= 1.29),  
lifecycle (>= 0.2.0),  
purrr (>= 0.3.4),  
rlang (>= 0.4.10),  
stringr (>= 1.4.0),  
tibble (>= 3.0.3),  
tidyr (>= 1.1.1)

**Suggests** broom.mixed (>= 0.2.7),  
car (>= 3.0-11),  
covr,  
effectsize (>= 0.4.0),  
emmeans (>= 1.7.2),  
flextable (>= 0.5.10),

geepack,  
 GGally ( $\geq$  2.1.0),  
 Hmisc,  
 huxtable ( $\geq$  5.0.0),  
 insight ( $\geq$  0.14.4),  
 kableExtra ( $\geq$  1.3.4),  
 lme4,  
 mgcv,  
 mice ( $\geq$  3.10.0),  
 nnet,  
 officer,  
 parameters ( $\geq$  0.6.0),  
 parsnip ( $\geq$  0.1.7),  
 rmarkdown,  
 sandwich ( $\geq$  3.0.1),  
 scales,  
 smd ( $\geq$  0.6.6),  
 spelling ( $\geq$  2.2),  
 survey,  
 survival ( $\geq$  3.2-11),  
 testthat ( $\geq$  3.0.4),  
 workflows ( $\geq$  0.2.3)

**VignetteBuilder** knitr

**RdMacros** lifecycle

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.2

**Config/testthat/edition** 3

**Config/testthat/parallel** true

## R topics documented:

add_ci . . . . .	4
add_difference . . . . .	5
add_glance . . . . .	7
add_global_p . . . . .	9
add_n.tbl_summary . . . . .	11
add_n.tbl_survfit . . . . .	13
add_nevent.tbl_survfit . . . . .	13
add_nevent_regression . . . . .	14
add_n_regression . . . . .	15
add_overall . . . . .	16
add_p.tbl_continuous . . . . .	18
add_p.tbl_cross . . . . .	19
add_p.tbl_summary . . . . .	20
add_p.tbl_survfit . . . . .	22
add_p.tbl_svysummary . . . . .	24

add_q . . . . .	26
add_significance_stars . . . . .	27
add_stat . . . . .	29
add_stat_label . . . . .	31
add_vif . . . . .	33
as_flex_table . . . . .	34
as_gt . . . . .	35
as_hux_table . . . . .	36
as_kable . . . . .	38
as_kable_extra . . . . .	39
as_tibble.gtsummary . . . . .	41
bold_italicize_labels_levels . . . . .	42
bold_p . . . . .	44
combine_terms . . . . .	44
continuous_summary . . . . .	46
custom_tidiers . . . . .	47
inline_text.gtsummary . . . . .	49
inline_text.tbl_cross . . . . .	50
inline_text.tbl_regression . . . . .	51
inline_text.tbl_summary . . . . .	52
inline_text.tbl_survfit . . . . .	54
inline_text.tbl_uvregression . . . . .	55
modify . . . . .	57
modify_cols_merge . . . . .	60
modify_column_alignment . . . . .	61
modify_column_hide . . . . .	62
modify_fmt_fun . . . . .	62
modify_table_body . . . . .	64
modify_table_styling . . . . .	65
plot . . . . .	67
proportion_summary . . . . .	68
ratio_summary . . . . .	69
remove_row_type . . . . .	71
select_helpers . . . . .	72
separate_p_footnotes . . . . .	73
set_gtsummary_theme . . . . .	74
sort_filter_p . . . . .	75
style_number . . . . .	76
style_percent . . . . .	77
style_pvalue . . . . .	78
style_ratio . . . . .	79
style_sigfig . . . . .	80
tbl_butcher . . . . .	81
tbl_continuous . . . . .	82
tbl_cross . . . . .	83
tbl_custom_summary . . . . .	85
tbl_merge . . . . .	89
tbl_regression . . . . .	91
tbl_split . . . . .	93
tbl_stack . . . . .	94
tbl_strata . . . . .	96
tbl_summary . . . . .	99

tbl_survfit . . . . .	103
tbl_svsummary . . . . .	105
tbl_uvregression . . . . .	108
theme_gtsummary . . . . .	111
trial . . . . .	114

## Index 115

---

add_ci	<i>Add CI Column</i>
--------	----------------------

---

### Description

Add a new column with the confidence intervals for proportions, means, etc.

### Usage

```
add_ci(x, ...)

## S3 method for class 'tbl_summary'
add_ci(
  x,
  method = NULL,
  include = everything(),
  statistic = NULL,
  conf.level = 0.95,
  style_fun = NULL,
  pattern = NULL,
  ...
)
```

### Arguments

x	A <code>tbl_summary</code> object
...	Not used
method	Confidence interval method. Default is <code>list(all_categorical() ~ "wilson", all_continuous() ~ "t.test")</code> . Must be one of <code>c("wilson", "wilson.no.correct", "exact", "asymptotic")</code> for categorical variables, and <code>c("t.test", "wilcox.test")</code> for continuous variables. See details below.
include	variables to include in the summary table. Default is <code>everything()</code>
statistic	Formula indicating how the confidence interval will be displayed. Default is <code>list(all_categorical() ~ "{conf.low}%,{conf.high}%", all_continuous() ~ "{conf.low},{conf.high}")</code>
conf.level	Confidence level. Default is 0.95
style_fun	Function to style upper and lower bound of confidence interval. Default is <code>list(all_categorical() ~ purrr::partial(style_sigfig, scale = 100), all_continuous() ~ style_sigfig)</code> .
pattern	string indicating the pattern to use to merge the CI with the statistics cell. The default is <code>NULL</code> , where no columns are merged. The two columns that will be merged are the statistics column, represented by <code>"{stat}"</code> and the CI column represented by <code>"{ci}"</code> , e.g. <code>pattern = "{stat} ({ci})"</code> will merge the two columns with the CI in parentheses.

**Value**

gtsummary table

**method argument**

Methods `c("wilson", "wilson.no.correct")` are calculated with `prop.test(correct = c(TRUE, FALSE))`. The default method, "wilson", includes the Yates continuity correction. Methods `c("exact", "asymptotic")` are calculated with `Hmisc::binconf(method=)`. Confidence intervals for means are calculated using `t.test()` and `wilcox.test()` for pseudo-medians.

**Example Output****See Also**

Review [list, formula, and selector syntax](#) used throughout gtsummary

Other `tbl_summary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

**Examples**

```
# Example 1 -----
add_ci_ex1 <-
  trial %>%
  select(marker, response, trt) %>%
  tbl_summary(missing = "no",
              statistic = all_continuous() ~ "{mean} ({sd})") %>%
  add_ci()

# Example 2 -----
add_ci_ex2 <-
  trial %>%
  select(response, grade) %>%
  tbl_summary(statistic = all_categorical() ~ "{p}%",
              missing = "no") %>%
  add_ci(pattern = "{stat} ({ci}") %>%
  modify_footnote(everything() ~ NA)
```

---

add\_difference

*Add difference between groups*

---

**Description**

Add the difference between two groups (typically mean difference), along with the difference confidence interval and p-value.

**Usage**

```
add_difference(
  x,
  test = NULL,
  group = NULL,
  adj.vars = NULL,
  test.args = NULL,
  conf.level = 0.95,
  include = everything(),
  pvalue_fun = NULL,
  estimate_fun = NULL
)
```

**Arguments**

x	"tbl_summary" or "tbl_svysummary" object
test	List of formulas specifying statistical tests to perform for each variable, e.g. <code>list(all_continuous() ~ "t.test")</code> . Common tests include "t.test" or "ancova" for continuous data, and "prop.test" for dichotomous variables. See <a href="#">tests</a> for details and more tests.
group	Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is NULL. See <a href="#">tests</a> for methods that utilize the <code>group=</code> argument.
adj.vars	Variables to include in mean difference adjustment (e.g. in ANCOVA models)
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use <code>test.args = all_tests("t.test") ~ list(var.equal = TRUE)</code>
conf.level	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
pvalue_fun	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
estimate_fun	List of formulas specifying the formatting functions to round and format differences. Default is <code>list(all_continuous() ~ style_sigfig, all_categorical() ~ function(x) paste0(style_sigfig(x * 100), "%"))</code> Function to round and format difference. Default is <a href="#">style_sigfig()</a>

**Example Output****See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

**Examples**

```
# Example 1 -----
add_difference_ex1 <-
  trial %>%
  select(trt, age, marker, response, death) %>%
  tbl_summary(
    by = trt,
    statistic =
      list(
        all_continuous() ~ "{mean} ({sd})",
        all_dichotomous() ~ "{p}%"
      ),
    missing = "no"
  ) %>%
  add_n() %>%
  add_difference()

# Example 2 -----
# ANCOVA adjusted for grade and stage
add_difference_ex2 <-
  trial %>%
  select(trt, age, marker, grade, stage) %>%
  tbl_summary(
    by = trt,
    statistic = list(all_continuous() ~ "{mean} ({sd})"),
    missing = "no",
    include = c(age, marker, trt)
  ) %>%
  add_n() %>%
  add_difference(adj.vars = c(grade, stage))
```

---

 add\_glance

 Add Model Statistics
 

---

**Description**

Add model statistics returned from broom: `:glance()`. Statistics can either be appended to the table (`add_glance_table()`), or added as a table source note (`add_glance_source_note()`).

**Usage**

```
add_glance_table(
  x,
  include = everything(),
  label = NULL,
  fmt_fun = NULL,
  glance_fun = broom::glance
)

add_glance_source_note(
  x,
  include = everything(),
  label = NULL,
```

```

fmt_fun = NULL,
glance_fun = broom::glance,
text_interpret = c("md", "html"),
sep1 = " = ",
sep2 = "; "
)

```

### Arguments

x	'tbl_regression' object
include	list of statistics to include in output. Must be column names of the tibble returned by broom::glance(). The include argument can also be used to specify the order the statistics appear in the table.
label	List of formulas specifying statistic labels, e.g. list(r.squared ~ "R2", p.value ~ "P")
fmt_fun	List of formulas where the LHS is a statistic and the RHS is a function to format/round the statistics. The default is to round the number of observations and degrees of freedom to the nearest integer, p-values are styled with style_pvalue() and the remaining statistics are styled with style_sigfig(x, digits = 3)
glance_fun	function that returns model statistics. Default is broom::glance(). Custom functions must return a single row tibble.
text_interpret	String indicates whether source note text will be interpreted with gt::md() or gt::html(). Must be "md" (default) or "html".
sep1	Separator between statistic name and statistic. Default is " = ", e.g. "R2 = 0.456"
sep2	Separator between statistics. Default is "; "

### Value

gtsummary table

### Default Labels

The following statistics have set default labels when printed. When there is no default, the column name from broom::glance() is printed.

Statistic Name	Default Label
r.squared	R <sup>2</sup>
adj.r.squared	Adjusted R <sup>2</sup>
p.value	p-value
logLik	Log-likelihood
statistic	Statistic
df.residual	Residual df
null.deviance	Null deviance
df.null	Null df
nevent	N events
concordance	c-index
std.error.concordance	c-index SE
nobs	No. Obs.
deviance	Deviance



sigma

Sigma

**Tips**

When combining `add_glance_table()` with `tbl_merge()`, the ordering of the model terms and the glance statistics may become jumbled. To re-order the rows with glance statistics on bottom, use the script below:

```
tbl_merge(list(tbl1, tbl2)) %>%
  modify_table_body(~.x %>% arrange(row_type == "glance_statistic"))
```

**Example Output****See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

**Examples**

```
mod <- lm(age ~ marker + grade, trial) %>% tbl_regression()

# Example 1 -----
add_glance_ex1 <-
  mod %>%
  add_glance_table(
    label = list(sigma ~ "\u03C3"),
    include = c(r.squared, AIC, sigma)
  )

# Example 2 -----
add_glance_ex2 <-
  mod %>%
  add_glance_source_note(
    label = list(sigma ~ "\u03C3"),
    include = c(r.squared, AIC, sigma)
  )
```

---

 add\_global\_p

*Add the global p-values*


---

**Description**

This function uses `car::Anova(type = "III")` to calculate global p-values variables. Output from `tbl_regression` and `tbl_uvregression` objects supported.

**Usage**

```

add_global_p(x, ...)

## S3 method for class 'tbl_regression'
add_global_p(
  x,
  include = everything(),
  type = NULL,
  keep = FALSE,
  quiet = NULL,
  ...,
  terms = NULL
)

## S3 method for class 'tbl_uvregression'
add_global_p(
  x,
  type = NULL,
  include = everything(),
  keep = FALSE,
  quiet = NULL,
  ...
)

```

**Arguments**

x	Object with class <code>tbl_regression</code> from the <a href="#">tbl_regression</a> function
...	Additional arguments to be passed to <code>car::Anova</code>
include	Variables to calculate global p-value for. Input may be a vector of quoted or unquoted variable names. Default is <code>everything()</code>
type	Type argument passed to <code>car::Anova</code> . Default is "III"
keep	Logical argument indicating whether to also retain the individual p-values in the table output for each level of the categorical variable. Default is <code>FALSE</code>
quiet	Logical indicating whether to print messages in console. Default is <code>FALSE</code>
terms	DEPRECATED. Use <code>include=</code> argument instead.

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_uvregression` tools: [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_uvregression\(\)](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_uvregression\(\)](#)

Other `tbl_regression` tools: [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [combine\\_terms\(\)](#), [inline\\_text.tbl\\_regression](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_regression\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

**Examples**

```
# Example 1 -----
tbl_lm_global_ex1 <-
  lm(marker ~ age + grade, trial) %>%
  tbl_regression() %>%
  add_global_p()

# Example 2 -----
tbl_uv_global_ex2 <-
  trial[c("response", "trt", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    y = response,
    method.args = list(family = binomial),
    exponentiate = TRUE
  ) %>%
  add_global_p()
```

---

add_n.tbl_summary	<i>Add column with N</i>
-------------------	--------------------------

---

**Description**

For each variable in a `tbl_summary` table, the `add_n` function adds a column with the total number of non-missing (or missing) observations

**Usage**

```
## S3 method for class 'tbl_summary'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  missing = NULL,
  ...
)

## S3 method for class 'tbl_svsummary'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  missing = NULL,
  ...
)
```

**Arguments**

<code>x</code>	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function or with class <code>tbl_svysummary</code> from the <a href="#">tbl_svysummary</a> function
<code>statistic</code>	String indicating the statistic to report. Default is the number of non-missing observation for each variable, <code>statistic = "{n}"</code> . Other statistics available to report include: <ul style="list-style-type: none"> <li>• <code>"{N}"</code> total number of observations,</li> <li>• <code>"{n}"</code> number of non-missing observations,</li> <li>• <code>"{n_miss}"</code> number of missing observations,</li> <li>• <code>"{p}"</code> percent non-missing data,</li> <li>• <code>"{p_miss}"</code> percent missing data The argument uses <a href="#">glue::glue</a> syntax and multiple statistics may be reported, e.g. <code>statistic = "{n} / {N} ({p}%)"</code></li> </ul>
<code>col_label</code>	String indicating the column label. Default is <code>"**N**"</code>
<code>footnote</code>	Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is <code>FALSE</code>
<code>last</code>	Logical indicator to include N column last in table. Default is <code>FALSE</code> , which will display N column first.
<code>missing</code>	DEPRECATED. Logical argument indicating whether to print N ( <code>missing = FALSE</code> ), or N missing ( <code>missing = TRUE</code> ). Default is <code>FALSE</code>
<code>...</code>	Not used

**Value**

A `tbl_summary` or `tbl_svysummary` object

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_svysummary` tools: [add\\_overall\(\)](#), [add\\_p.tbl\\_svysummary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_svysummary\(\)](#)

**Examples**

```
# Example 1 -----
tbl_n_ex <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_summary(by = trt) %>%
  add_n()
```

---

add_n.tbl_survfit	<i>Add column with number of observations</i>
-------------------	---

---

### Description

**[Maturing]** For each `survfit()` object summarized with `tbl_survfit()` this function will add the total number of observations in a new column.

### Usage

```
## S3 method for class 'tbl_survfit'
add_n(x, ...)
```

### Arguments

x	object of class "tbl_survfit"
...	Not used

### Example Output

### See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_survfit` tools: [add\\_nevent.tbl\\_survfit\(\)](#), [add\\_p.tbl\\_survfit\(\)](#), [modify.tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_survfit\(\)](#)

### Examples

```
library(survival)
fit1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ trt, trial)

# Example 1 -----
add_n.tbl_survfit_ex1 <-
  list(fit1, fit2) %>%
  tbl_survfit(times = c(12, 24)) %>%
  add_n()
```

---

add_nevent.tbl_survfit	<i>Add column with number of observed events</i>
------------------------	--

---

### Description

**[Maturing]** For each `survfit()` object summarized with `tbl_survfit()` this function will add the total number of events observed in a new column.

**Usage**

```
## S3 method for class 'tbl_survfit'
add_nevent(x, ...)
```

**Arguments**

```
x          object of class 'tbl_survfit'
...        Not used
```

**Example Output****See Also**

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_survfit()`

**Examples**

```
library(survival)
fit1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ trt, trial)

# Example 1 -----
add_nevent.tbl_survfit_ex1 <-
  list(fit1, fit2) %>%
  tbl_survfit(times = c(12, 24)) %>%
  add_n() %>%
  add_nevent()
```

---

`add_nevent_regression` *Add event N to regression table*

---

**Description**

Add event N to regression table

**Usage**

```
## S3 method for class 'tbl_regression'
add_nevent(x, location = NULL, ...)

## S3 method for class 'tbl_uvregression'
add_nevent(x, location = NULL, ...)
```

**Arguments**

```
x          a tbl_regression or tbl_uvregression table
location   location to place Ns. When "label" total Ns are placed on each variable's label row. When "level" level counts are placed on the variable level for categorical variables, and total N on the variable's label row for continuous.
...        Not used
```

**Example Output****Examples**

```
# Example 1 -----
add_nevent.tbl_regression_ex1 <-
  trial %>%
  select(response, trt, grade) %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
  ) %>%
  add_nevent()
# Example 2 -----
add_nevent.tbl_regression_ex2 <-
  glm(response ~ age + grade, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE) %>%
  add_nevent(location = "level")
```

---

add_n_regression	<i>Add N to regression table</i>
------------------	----------------------------------

---

**Description**

Add N to regression table

**Usage**

```
## S3 method for class 'tbl_regression'
add_n(x, location = NULL, ...)

## S3 method for class 'tbl_uvregression'
add_n(x, location = NULL, ...)
```

**Arguments**

x	a <code>tbl_regression</code> or <code>tbl_uvregression</code> table
location	location to place Ns. When "label" total Ns are placed on each variable's label row. When "level" level counts are placed on the variable level for categorical variables, and total N on the variable's label row for continuous.
...	Not used

**Example Output**

**Examples**

```
# Example 1 -----
add_n.tbl_regression_ex1 <-
  trial %>%
  select(response, age, grade) %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
    hide_n = TRUE
  ) %>%
  add_n(location = "label")

# Example 2 -----
add_n.tbl_regression_ex2 <-
  glm(response ~ age + grade, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE) %>%
  add_n(location = "level")
```

---

add\_overall

*Add column with overall summary statistics*


---

**Description**

Adds a column with overall summary statistics to tables created by `tbl_summary`, `tbl_svsummary`, `tbl_continuous` or `tbl_custom_summary`.

**Usage**

```
add_overall(x, ...)
```

```
## S3 method for class 'tbl_summary'
add_overall(
  x,
  last = FALSE,
  col_label = NULL,
  statistic = NULL,
  digits = NULL,
  ...
)
```

```
## S3 method for class 'tbl_svsummary'
add_overall(
  x,
  last = FALSE,
  col_label = NULL,
  statistic = NULL,
  digits = NULL,
  ...
)
```



```
## S3 method for class 'tbl_continuous'
add_overall(
  x,
  last = FALSE,
  col_label = NULL,
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_custom_summary'
add_overall(
  x,
  last = FALSE,
  col_label = NULL,
  statistic = NULL,
  digits = NULL,
  ...
)
```

### Arguments

x	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function, object with class <code>tbl_svysummary</code> from the <a href="#">tbl_svysummary</a> function, object with class <code>tbl_continuous</code> from the <a href="#">tbl_continuous</a> function or object with class <code>tbl_custom_summary</code> from the <a href="#">tbl_custom_summary</a> function.
...	Not used
last	Logical indicator to display overall column last in table. Default is <code>FALSE</code> , which will display overall column first.
col_label	String indicating the column label. Default is <code>"**Overall**, N = {N}"</code>
statistic	Override the statistic argument in initial <code>tbl_*</code> function. call. Default is <code>NULL</code> .
digits	Override the digits argument in initial <code>tbl_*</code> function call. Default is <code>NULL</code> .

### Value

A `tbl_*` of same class as x

### Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_svsummary` tools: `add_n.tbl_summary()`, `add_p.tbl_svsummary()`, `add_q()`, `add_stat_label()`, `modify`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_svsummary()`

Other `tbl_custom_summary` tools: `continuous_summary()`, `proportion_summary()`, `ratio_summary()`, `tbl_custom_summary()`

## Examples

```
tbl_overall_ex1 <-
  trial %>%
  tbl_summary(include = c(age, grade), by = trt) %>%
  add_overall()
tbl_overall_ex2 <-
  trial %>%
  tbl_summary(
    include = grade,
    by = trt,
    percent = "row",
    statistic = ~ "{p}%",
    digits = ~ 1
  ) %>%
  add_overall(
    last = TRUE,
    statistic = ~ "{p}% (n={n})",
    digits = ~ c(1, 0)
  )
tbl_overall_ex3 <-
  trial %>%
  tbl_continuous(
    variable = age,
    by = trt,
    include = grade
  ) %>%
  add_overall(last = TRUE)
```

---

add\_p.tbl\_continuous *P-values for tbl\_continuous*

---

## Description

P-values for `tbl_continuous`

## Usage

```
## S3 method for class 'tbl_continuous'
add_p(
  x,
  test = NULL,
  pvalue_fun = NULL,
  include = everything(),
  test.args = NULL,
  group = NULL,
  ...
)
```

**Arguments**

x	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function
test	List of formulas specifying statistical tests to perform for each variable. Default is two-way ANOVA when <code>by=</code> is not <code>NULL</code> , and has the same defaults as <code>add_p.tbl_continuous()</code> when <code>by = NULL</code> . See <a href="#">tests</a> for details, more tests, and instruction for implementing a custom test.
pvalue_fun	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use <code>test.args = all_tests("t.test") ~ list(var.equal = TRUE)</code>
group	Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is <code>NULL</code> . See <a href="#">tests</a> for methods that utilize the <code>group=</code> argument.
...	Not used

**Examples**

```
add_p_continuous_ex1 <-
  tbl_continuous(
    data = trial,
    variable = age,
    by = trt,
    include = grade
  ) %>%
  add_p()
```

---

add_p.tbl_cross	<i>Adds p-value to crosstab table</i>
-----------------	---------------------------------------

---

**Description**

Calculate and add a p-value comparing the two variables in the cross table. Missing values are included in p-value calculations.

**Usage**

```
## S3 method for class 'tbl_cross'
add_p(
  x,
  test = NULL,
  pvalue_fun = NULL,
  source_note = NULL,
  test.args = NULL,
  ...
)
```

**Arguments**

x	Object with class <code>tbl_cross</code> from the <a href="#">tbl_cross</a> function
test	A string specifying statistical test to perform. Default is "chisq.test" when expected cell counts $\geq 5$ and "fisher.test" when expected cell counts $< 5$ .
pvalue_fun	Function to round and format p-value. Default is <a href="#">style_pvalue</a> , except when <code>source_note = TRUE</code> when the default is <code>style_pvalue(x, prepend_p = TRUE)</code>
source_note	Logical value indicating whether to show p-value in the {gt} table source notes rather than a column.
test.args	Named list containing additional arguments to pass to the test (if it accepts additional arguments). For example, add an argument for a chi-squared test with <code>test.args = list(correct = TRUE)</code>
...	Not used

**Example Output****Author(s)**

Karissa Whiting

**See Also**

Other `tbl_cross` tools: [inline\\_text.tbl\\_cross\(\)](#), [tbl\\_cross\(\)](#)

**Examples**

```
# Example 1 -----
add_p_cross_ex1 <-
  trial %>%
  tbl_cross(row = stage, col = trt) %>%
  add_p()

# Example 2 -----
add_p_cross_ex2 <-
  trial %>%
  tbl_cross(row = stage, col = trt) %>%
  add_p(source_note = TRUE)
```

---

`add_p.tbl_summary`      *Adds p-values to summary tables*

---

**Description**

Adds p-values to tables created by `tbl_summary` by comparing values across groups.

**Usage**

```
## S3 method for class 'tbl_summary'
add_p(
  x,
  test = NULL,
  pvalue_fun = NULL,
  group = NULL,
  include = everything(),
  test.args = NULL,
  exclude = NULL,
  ...
)
```

**Arguments**

x	Object with class <code>tbl_summary</code> from the <code>tbl_summary</code> function
test	List of formulas specifying statistical tests to perform for each variable, e.g. <code>list(all_continuous() ~ "t.test", all_categorical() ~ "fisher.test")</code> . Common tests include <code>"t.test"</code> , <code>"aov"</code> , <code>"wilcox.test"</code> , <code>"kruskal.test"</code> , <code>"chisq.test"</code> , <code>"fisher.test"</code> , and <code>"lme4"</code> (for clustered data). See <a href="#">tests</a> for details, more tests, and instruction for implementing a custom test. Tests default to <code>"kruskal.test"</code> for continuous variables ( <code>"wilcox.test"</code> when <code>"by"</code> variable has two levels), <code>"chisq.test.no.correct"</code> for categorical variables with all expected cell counts $\geq 5$ , and <code>"fisher.test"</code> for categorical variables with any expected cell count $< 5$ .
pvalue_fun	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
group	Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is <code>NULL</code> . See <a href="#">tests</a> for methods that utilize the <code>group=</code> argument.
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use <code>test.args = all_tests("t.test") ~ list(var.equal = TRUE)</code>
exclude	DEPRECATED.
...	Not used

**Value**

A `tbl_summary` object

**Example Output**

**Author(s)**

Daniel D. Sjoberg, Emily C. Zabor

**See Also**

See `tbl_summary` [vignette](#) for detailed examples

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

**Examples**

```
# Example 1 -----
add_p_ex1 <-
  trial[c("age", "grade", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p()

# Example 2 -----
add_p_ex2 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_p(
    # perform t-test for all variables
    test = everything() ~ "t.test",
    # assume equal variance in the t-test
    test.args = all_tests("t.test") ~ list(var.equal = TRUE)
  )
```

---

`add_p.tbl_survfit`      *Adds p-value to survfit table*

---

**Description**

**[Maturing]** Calculate and add a p-value

**Usage**

```
## S3 method for class 'tbl_survfit'
add_p(
  x,
  test = "logrank",
  test.args = NULL,
  pvalue_fun = style_pvalue,
  include = everything(),
  quiet = NULL,
  ...
)
```

**Arguments**

x	Object of class "tbl_survfit"
test	string indicating test to use. Must be one of "logrank", "survdiff", "petopeto_gehanwilcoxon", "coxph_lrt", "coxph_wald", "coxph_score". See details below
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests("t.test") ~ list(var.equal = TRUE)
pvalue_fun	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is <code>everything()</code> .
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Not used

**test argument**

The most common way to specify `test=` is by using a single string indicating the test name. However, if you need to specify different tests within the same table, the input is flexible using the list notation common throughout the `gtsummary` package. For example, the following code would call the log-rank test, and a second test of the *G-rho* family.

```
... %>%
  add_p(test = list(trt ~ "logrank", grade ~ "survdiff"),
        test.args = grade ~ list(rho = 0.5))
```

**Example Output****See Also**

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `modify.tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_survfit()`

**Examples**

```
library(survival)

gts_survfit <-
  list(
    survfit(Surv(ttdeath, death) ~ grade, trial),
    survfit(Surv(ttdeath, death) ~ trt, trial)
  ) %>%
  tbl_survfit(times = c(12, 24))

# Example 1 -----
add_p_tbl_survfit_ex1 <-
  gts_survfit %>%
  add_p()
```

```
# Example 2 -----
# Pass `rho=` argument to `survdiff()`
add_p_tbl_survfit_ex2 <-
  gts_survfit %>%
  add_p(test = "survdiff", test.args = list(rho = 0.5))
```

---

add\_p.tbl\_svysummary *Adds p-values to svysummary tables*

---

## Description

Adds p-values to tables created by `tbl_svysummary` by comparing values across groups.

## Usage

```
## S3 method for class 'tbl_svysummary'
add_p(
  x,
  test = NULL,
  pvalue_fun = NULL,
  include = everything(),
  test.args = NULL,
  ...
)
```

## Arguments

<code>x</code>	Object with class <code>tbl_svysummary</code> from the <a href="#">tbl_svysummary</a> function
<code>test</code>	List of formulas specifying statistical tests to perform, e.g. <code>list(all_continuous() ~ "svy.t.test", all_categorical() ~ "svy.wald.test")</code> . Options include <ul style="list-style-type: none"> <li>"svy.t.test" for a t-test adapted to complex survey samples (cf. <code>survey::svyttest</code>),</li> <li>"svy.wilcox.test" for a Wilcoxon rank-sum test for complex survey samples (cf. <code>survey::svyranktest</code>),</li> <li>"svy.kruskal.test" for a Kruskal-Wallis rank-sum test for complex survey samples (cf. <code>survey::svyranktest</code>),</li> <li>"svy.vanderwaerden.test" for a van der Waerden's normal-scores test for complex survey samples (cf. <code>survey::svyranktest</code>),</li> <li>"svy.median.test" for a Mood's test for the median for complex survey samples (cf. <code>survey::svyranktest</code>),</li> <li>"svy.chisq.test" for a Chi-squared test with Rao &amp; Scott's second-order correction (cf. <code>survey::svychisq</code>),</li> <li>"svy.adj.chisq.test" for a Chi-squared test adjusted by a design effect estimate (cf. <code>survey::svychisq</code>),</li> <li>"svy.wald.test" for a Wald test of independence for complex survey samples (cf. <code>survey::svychisq</code>),</li> <li>"svy.adj.wald.test" for an adjusted Wald test of independence for complex survey samples (cf. <code>survey::svychisq</code>),</li> <li>"svy.lincom.test" for a test of independence using the exact asymptotic distribution for complex survey samples (cf. <code>survey::svychisq</code>),</li> </ul>



- "svy.saddlepoint.test" for a test of independence using a saddlepoint approximation for complex survey samples (cf. `survey::svychisq`),

Tests default to "svy.wilcox.test" for continuous variables and "svy.chisq.test" for categorical variables.

pvalue_fun	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use <code>test.args = all_tests("t.test") ~ list(var.equal = TRUE)</code>
...	Not used

**Value**

A `tbl_svysummary` object

**Example Output****Author(s)**

Joseph Larmarange

**See Also**

Other `tbl_svysummary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_q()`, `add_stat_label()`, `modify`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_svysummary()`

**Examples**

```
# Example 1 -----
# A simple weighted dataset
add_p_svysummary_ex1 <-
  survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) %>%
  tbl_svysummary(by = Survived) %>%
  add_p()

# A dataset with a complex design
data(api, package = "survey")
d_clust <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

# Example 2 -----
add_p_svysummary_ex2 <-
  tbl_svysummary(d_clust, by = both, include = c(cname, api00, api99, both)) %>%
  add_p()

# Example 3 -----
```

```
# change tests to svy t-test and Wald test
add_p_svysummary_ex3 <-
  tbl_svysummary(d_clust, by = both, include = c(cname, api00, api99, both)) %>%
  add_p(
    test = list(
      all_continuous() ~ "svy.t.test",
      all_categorical() ~ "svy.wald.test"
    )
  )
```

---

add_q	<i>Add a column of q-values to account for multiple comparisons</i>
-------	---

---

### Description

Adjustments to p-values are performed with [stats::p.adjust](#).

### Usage

```
add_q(x, method = "fdr", pvalue_fun = NULL, quiet = NULL)
```

### Arguments

x	a gtsummary object
method	String indicating method to be used for p-value adjustment. Methods from <a href="#">stats::p.adjust</a> are accepted. Default is method = "fdr".
pvalue_fun	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
quiet	Logical indicating whether to print messages in console. Default is FALSE

### Example Output

#### Author(s)

Esther Drill, Daniel D. Sjoberg

#### See Also

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_svysummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svysummary\(\)](#), [add\\_stat\\_label\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_svysummary\(\)](#)

Other `tbl_regression` tools: `add_global_p()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`

Other `tbl_uvregression` tools: `add_global_p()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_uvregression()`

## Examples

```
# Example 1 -----
add_q_ex1 <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  add_q()

# Example 2 -----
add_q_ex2 <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
    exponentiate = TRUE
  ) %>%
  add_global_p() %>%
  add_q()
```

---

add\_significance\_stars

*Add significance stars*

---

## Description

**[Experimental]** Add significance stars to estimates with small p-values

## Usage

```
add_significance_stars(
  x,
  pattern = "{estimate}{stars}",
  thresholds = c(0.001, 0.01, 0.05),
  hide_ci = TRUE,
  hide_p = TRUE,
  hide_se = FALSE
)
```

**Arguments**

x	a 'tbl_regression' or 'tbl_uvregression' object
pattern	glue-syntax string indicating what to display in formatted column. Default is "{estimate}{stars}". Other common patterns are "{estimate}{stars} ({conf.low},{conf.high})" and "{estimate} ({conf.low} to {conf.high}){stars}"
thresholds	thresholds for significance stars. Default is c(0.001, 0.01, 0.05)
hide_ci	logical whether to hide confidence interval. Default is TRUE
hide_p	logical whether to hide p-value. Default is TRUE
hide_se	logical whether to hide standard error. Default is FALSE

**Future Updates**

There are planned updates to the implementation of this function with respect to the `pattern=` argument. Currently, this function replaces the numeric estimate column, with a formatted character column following `pattern=`. Once `gt::cols_merge()` gains the `rows=` argument the implementation will be updated to use it, which will keep numeric columns numeric. For the *vast majority* of users, *the planned change will be go unnoticed*.

**Example Output****Examples**

```
tbl <-
  lm(time ~ ph.ecog + sex, survival::lung) %>%
  tbl_regression(label = list(ph.ecog = "ECOG Score", sex = "Sex"))

# Example 1 -----
add_significance_stars_ex1 <-
  tbl %>%
  add_significance_stars(hide_ci = FALSE, hide_p = FALSE)

# Example 2 -----
add_significance_stars_ex2 <-
  tbl %>%
  add_significance_stars(
    pattern = "{estimate} ({conf.low}, {conf.high}){stars}",
    hide_ci = TRUE, hide_se = TRUE
  ) %>%
  modify_header(estimate ~ "**Beta (95% CI)**") %>%
  modify_footnote(estimate ~ "CI = Confidence Interval", abbreviation = TRUE)

# Example 3 -----
# Use <br> to put a line break between beta and SE in HTML output
add_significance_stars_ex3 <-
  tbl %>%
  add_significance_stars(
    hide_se = TRUE,
    pattern = "{estimate}{stars}<br>({std.error})"
  ) %>%
  modify_header(estimate ~ "**Beta (SE)**") %>%
  modify_footnote(estimate ~ "SE = Standard Error", abbreviation = TRUE) %>%
  as_gt() %>%
```

```
gt::tab_style(
  style = "vertical-align:top",
  locations = gt::cells_body(columns = vars(label))
)
```

---

add_stat	<i>Add a custom statistic column</i>
----------	--------------------------------------

---

## Description

**[Maturing]** The function allows a user to add a new column (or columns) of statistics to an existing `tbl_summary`, `tbl_svsummary`, or `tbl_continuous` object.

## Usage

```
add_stat(x, fns, location = NULL, ...)
```

## Arguments

<code>x</code>	<code>tbl_summary</code> , <code>tbl_svsummary</code> , or <code>tbl_continuous</code> object
<code>fns</code>	list of formulas indicating the functions that create the statistic. See details below.
<code>location</code>	list of formulas indicating the location the new statistics are placed. The RHS of the formula must be one of <code>c("label", "level", "missing")</code> . When "label", a single statistic is placed on the variable label row. When "level" the statistics are placed on the variable level rows. The length of the vector of statistics returned from the <code>fns</code> function must match the dimension of levels. Default is to place the new statistics on the label row.
<code>...</code>	DEPRECATED

## Details

The returns from custom functions passed in `fns=` are required to follow a specified format. Each of these function will execute on a single variable.

1. Each function must return a tibble or a vector. If a vector is returned, it will be converted to a tibble with one column and number of rows equal to the length of the vector.
2. When `location = "label"`, the returned statistic from the custom function must be a tibble with one row. When `location = "level"` the tibble must have the same number of rows as there are levels in the variable (excluding the row for unknown values).
3. Each function may take the following arguments: `foo(data, variable, by, tbl, ...)`
  - `data=` is the input data frame passed to `tbl_summary()`
  - `variable=` is a string indicating the variable to perform the calculation on. This is the variable in the label column of the table.
  - `by=` is a string indicating the by variable from `tbl_summary=`, if present
  - `tbl=` the original `tbl_summary()/tbl_svsummary()` object is also available to utilize

The user-defined does not need to utilize each of these inputs. It's encouraged the user-defined function accept `...` as each of the arguments *will* be passed to the function, even if not all inputs are utilized by the user's function, e.g. `foo(data, variable, by, ...)`

- Use `modify_header()` to update the column headers
- Use `modify_fmt_fun()` to update the functions that format the statistics
- Use `modify_footnote()` to add an explanatory footnote

If you return a tibble with column names `p.value` or `q.value`, default p-value formatting will be applied, and you may take advantage of subsequent p-value formatting functions, such as `bold_p()` or `add_q()`.

To access the continuous variable in a `tbl_continuous()` table, use `tbl$inputs$variable`.

## Example Output

## See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

## Examples

```
library(dplyr, warn.conflicts = FALSE)
library(stringr)
# Example 1 -----
# fn returns t-test pvalue
my_ttest <- function(data, variable, by, ...) {
  t.test(data[[variable]] ~ as.factor(data[[by]]))$p.value
}

add_stat_ex1 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(fns = everything() ~ my_ttest) %>%
  modify_header(
    list(
      add_stat_1 ~ "***p-value**",
      all_stat_cols() ~ "**{level}**"
    )
  )

# Example 2 -----
# fn returns t-test test statistic and pvalue
my_ttest2 <- function(data, variable, by, ...) {
  t.test(data[[variable]] ~ as.factor(data[[by]])) %>%
  broom::tidy() %>%
  mutate(
    stat = str_glue("t={style_sigfig(statistic)}, {style_pvalue(p.value, prepend_p = TRUE)}")
  ) %>%
  pull(stat)
}

add_stat_ex2 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(fns = everything() ~ my_ttest2) %>%
  modify_header(add_stat_1 ~ "**Treatment Comparison**")
```

```

# Example 3 -----
# return test statistic and p-value in separate columns
my_ttest3 <- function(data, variable, by, ...) {
  t.test(data[[variable]] ~ as.factor(data[[by]])) %>%
    broom::tidy() %>%
    select(statistic, p.value)
}

add_stat_ex3 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(fns = everything() ~ my_ttest3) %>%
  modify_header(
    list(
      statistic ~ "**t-statistic**",
      p.value ~ "**p-value**"
    )
  ) %>%
  modify_fmt_fun(
    list(
      statistic ~ style_sigfig,
      p.value ~ style_pvalue
    )
  )

```

---

add_stat_label	<i>Add statistic labels</i>
----------------	-----------------------------

---

## Description

Adds labels describing the summary statistics presented for each variable in the [tbl\\_summary](#) / [tbl\\_svysummary](#) table.

## Usage

```
add_stat_label(x, location = NULL, label = NULL)
```

## Arguments

x	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function or with class <code>tbl_svysummary</code> from the <a href="#">tbl_svysummary</a> function
location	location where statistic label will be included. "row" (the default) to add the statistic label to the variable label row, and "column" adds a column with the statistic label.
label	a list of formulas or a single formula updating the statistic label, e.g. <code>label = all_categorical() ~ "No. (%)"</code>

## Value

A `tbl_summary` or `tbl_svysummary` object

**Tips**

When using `add_stat_label(location='row')` with subsequent `tbl_merge()`, it's important to have somewhat of an understanding of the underlying structure of the `gtsummary` table. `add_stat_label(location='row')` works by adding a new column called "stat\_label" to `x$table_body`. The "label" and "stat\_label" columns are merged when the `gtsummary` table is printed. The `tbl_merge()` function merges on the "label" column (among others), which is typically the first column you see in a `gtsummary` table. Therefore, when you want to merge a table that has run `add_stat_label(location='row')` you need to match the "label" column values before the "stat\_column" is merged with it.

For example, the following two tables merge properly

```
tbl1 <- trial %>% select(age, grade) %>% tbl_summary() %>% add_stat_label()
tbl2 <- lm(marker ~ age + grade, trial) %>% tbl_regression()

tbl_merge(list(tbl1, tbl2))
```

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_svsummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svsummary\(\)](#), [add\\_q\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_svsummary\(\)](#)

**Examples**

```
tbl <- trial %>%
  dplyr::select(trt, age, grade, response) %>%
  tbl_summary(by = trt)

# Example 1 -----
# Add statistic presented to the variable label row
add_stat_label_ex1 <-
  tbl %>%
  add_stat_label(
    # update default statistic label for continuous variables
    label = all_continuous() ~ "med. (iqr)"
  )

# Example 2 -----
add_stat_label_ex2 <-
  tbl %>%
  add_stat_label(
```



```

    # add a new column with statistic labels
    location = "column"
  )

# Example 3 -----
add_stat_label_ex3 <-
  trial %>%
  select(age, grade, trt) %>%
  tbl_summary(
    by = trt,
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c("{mean} ({sd})", "{min} - {max}"),
  ) %>%
  add_stat_label(label = age ~ c("Mean (SD)", "Min - Max"))

```

---

add\_vif

*Add Variance Inflation Factor*


---

## Description

**[Maturing]** Add the variance inflation factor (VIF) or generalized VIF (GVIF) to the regression table. Function uses `car::vif()` to calculate the VIF.

## Usage

```
add_vif(x, statistic = NULL, estimate_fun = NULL)
```

## Arguments

<code>x</code>	'tbl_regression' object
<code>statistic</code>	"VIF" (variance inflation factors, for models with no categorical terms) or one of/combination of "GVIF" (generalized variance inflation factors), "aGVIF" 'adjusted GVIF, i.e. $GVIF^{1/(2*df)}$ and/or "df" (degrees of freedom). See <code>car::vif()</code> for details.
<code>estimate_fun</code>	Default is <code>style_sigfig()</code> .

## Example Output

## See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

## Examples

```

# Example 1 -----
add_vif_ex1 <-
  lm(age ~ grade + marker, trial) %>%
  tbl_regression() %>%
  add_vif()

# Example 2 -----

```

```
add_vif_ex2 <-
  lm(age ~ grade + marker, trial) %>%
  tbl_regression() %>%
  add_vif(c("aGVIF", "df"))
```

---

as\_flex\_table

*Convert gtsummary object to a flextable object*


---

## Description

Function converts a gtsummary object to a flextable object. A user can use this function if they wish to add customized formatting available via the flextable functions. The flextable output is particularly useful when combined with R markdown with Word output, since the gt package does not support Word.

## Usage

```
as_flex_table(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = TRUE
)
```

## Arguments

x	Object created by a function from the gtsummary package (e.g. <a href="#">tbl_summary</a> or <a href="#">tbl_regression</a> )
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed.

## Value

A flextable object

## Details

The as\_flex\_table() functions converts the gtsummary object to a flextable, and prints it with the following styling functions.

1. flextable::flextable()
2. flextable::set\_header\_labels() to set column labels
3. flextable::add\_header\_row(), if applicable, to set spanning column header
4. flextable::align() to set column alignment
5. flextable::padding() to indent variable levels

6. `flextable::fontsize()` to set font size
7. `flextable::autofit()` to estimate the column widths
8. `flextable::footnote()` to add table footnotes and source notes
9. `flextable::bold()` to bold cells in data frame
10. `flextable::italic()` to italicize cells in data frame
11. `flextable::border()` to set all border widths to 1
12. `flextable::padding()` to set consistent header padding
13. `flextable::valign()` to ensure label column is top-left justified

Any one of these commands may be omitted using the `include=` argument.

Pro tip: Use the `flextable::width()` function for exacting control over column width after calling `as_flex_table()`.

### Example Output

#### Author(s)

Daniel D. Sjoberg

#### See Also

Other `gtsummary` output types: [as\\_gt\(\)](#), [as\\_hux\\_table\(\)](#), [as\\_kable\\_extra\(\)](#), [as\\_kable\(\)](#), [as\\_tibble.gtsummary\(\)](#)

#### Examples

```
as_flex_table_ex1 <-
  trial %>%
  select(trt, age, grade) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  as_flex_table()
```

---

as\_gt

*Convert gtsummary object to a gt object*

---

#### Description

Function converts a `gtsummary` object to a "gt\_tbl" object, that is, a table created with `gt::gt()`. Function is used in the background when the results are printed or knit. A user can use this function if they wish to add customized formatting available via the [gt package](#).

Review the [tbl\\_summary vignette](#) or [tbl\\_regression vignette](#) for detailed examples in the 'Advanced Customization' section.

#### Usage

```
as_gt(x, include = everything(), return_calls = FALSE, ..., exclude = NULL)
```

**Arguments**

x	Object created by a function from the gtsummary package (e.g. <a href="#">tbl_summary</a> or <a href="#">tbl_regression</a> )
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
...	Arguments passed on to <a href="#">gt::gt</a>
exclude	DEPRECATED.

**Value**

A gt\_tbl object

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

Other gtsummary output types: [as\\_flex\\_table\(\)](#), [as\\_hux\\_table\(\)](#), [as\\_kable\\_extra\(\)](#), [as\\_kable\(\)](#), [as\\_tibble.gtsummary\(\)](#)

**Examples**

```
as_gt_ex <-
  trial[c("trt", "age", "response", "grade")] %>%
  tbl_summary(by = trt) %>%
  as_gt()
```

---

as\_hux\_table

---

*Convert gtsummary object to a huxtable object*


---

**Description**

Function converts a gtsummary object to a huxtable object. A user can use this function if they wish to add customized formatting available via the huxtable functions. The huxtable package supports output to PDF via LaTeX, as well as HTML and Word.

**Usage**

```
as_hux_table(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = FALSE
)
```

**Arguments**

x	Object created by a function from the gtsummary package (e.g. <a href="#">tbl_summary</a> or <a href="#">tbl_regression</a> )
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed.

**Value**

A huxtable object

**Details**

The `as_hux_table()` takes the data frame that will be printed, converts it to a huxtable and formats the table with the following huxtable functions:

1. `huxtable::huxtable()`
2. `huxtable::insert_row()` to insert header rows
3. `huxtable::set_left_padding()` to indent variable levels
4. `huxtable::add_footnote()` to add table footnotes and source notes
5. `huxtable::set_bold()` to bold cells
6. `huxtable::set_italic()` to italicize cells
7. `huxtable::set_top_border()` add horizontal line (when indicated)
8. `huxtable::set_na_string()` to use an em-dash for missing numbers
9. `huxtable::set_markdown()` use markdown for header rows
10. `huxtable::set_align()` to set column alignment

Any one of these commands may be omitted using the `include=` argument.

**Author(s)**

David Hugh-Jones

**See Also**

Other gtsummary output types: [as\\_flex\\_table\(\)](#), [as\\_gt\(\)](#), [as\\_kable\\_extra\(\)](#), [as\\_kable\(\)](#), [as\\_tibble.gtsummary\(\)](#)

**Examples**

```
trial %>%
  dplyr::select(trt, age, grade) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  as_hux_table()
```

as\_kable

*Convert gtsummary object to a kable object***Description**

Function converts a gtsummary object to a knitr\_kable object. This function may be used in the background when the tables are printed or knitted.

Output from `knitr::kable` is less full featured compared to summary tables produced with `gt`. For example, kable summary tables do not include indentation, footnotes, or spanning header rows.

**Usage**

```
as_kable(
  x,
  include = everything(),
  return_calls = FALSE,
  exclude = NULL,
  fmt_missing = TRUE,
  ...
)
```

**Arguments**

x	Object created by a function from the gtsummary package (e.g. <code>tbl_summary</code> or <code>tbl_regression</code> )
include	Commands to include in output. Input may be a vector of quoted or unquoted names. <code>tidyselect</code> and <code>gtsummary</code> select helper functions are also accepted. Default is <code>everything()</code> .
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED
fmt_missing	Logical argument adding the missing value formats.
...	Additional arguments passed to <code>knitr::kable</code>

**Details**

Tip: To better distinguish variable labels and level labels when indenting is not supported, try `bold_labels()` or `italicize_levels()`.

**Value**

A `knitr_kable` object

**Author(s)**

Daniel D. Sjoberg

**See Also**

Other gtsummary output types: `as_flex_table()`, `as_gt()`, `as_hux_table()`, `as_kable_extra()`, `as_tibble.gtsummary()`

**Examples**

```

trial %>%
  tbl_summary(by = trt) %>%
  bold_labels() %>%
  as_kable()

```

---

as_kable_extra	<i>Convert gtsummary object to a kableExtra object</i>
----------------	--

---

**Description**

Function converts a gtsummary object to a knitr\_kable + kableExtra object. This allows the customized formatting available via [knitr::kable](#) and kableExtra; as\_kable\_extra() supports arguments in knitr::kable(). as\_kable\_extra() output via gtsummary supports bold and italic cells for table bodies. Users creating pdf output should specify as\_kable\_extra(format = "latex").

**Usage**

```

as_kable_extra(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = TRUE,
  fmt_missing = TRUE,
  ...
)

```

**Arguments**

x	Object created by a function from the gtsummary package (e.g. <a href="#">tbl_summary</a> or <a href="#">tbl_regression</a> )
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed.
fmt_missing	Logical argument adding the missing value formats.
...	Additional arguments passed to <a href="#">knitr::kable</a>

**Value**

A kableExtra object

## PDF via LaTeX Tips

This section discusses options intended for use with

- `output: pdf_document` in yml of `.Rmd`.
- `as_kable_extra(format = "latex")`

### Custom column names:

In pdf output, column names do not currently inherit formatting applied in `gtsummary` tables. However, custom column names can be achieved with the `col.names` argument as shown in Example 2, including attributes such as bold formatting, italic formatting, and line breaks. Doing so requires the `escape = FALSE` argument; however, when using `escape = FALSE` special latex characters like `\` and `%` will need to be escaped prior to entering `as_kable_extra()`. Using `escape = FALSE` when the `gtsummary` table has special LaTeX characters will result in the error "LaTeX failed to compile..."

### Additional table styling:

Additional styling is available through `knitr::kable()` and `kableExtra::kable_styling()` as shown in Example 3, which implements row striping and repeated column headers in the presence of page breaks.

## Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

Other `gtsummary` output types: [as\\_flex\\_table\(\)](#), [as\\_gt\(\)](#), [as\\_hux\\_table\(\)](#), [as\\_kable\(\)](#), [as\\_tibble.gtsummary\(\)](#)

## Examples

```
# Example 1 (general) -----
as_kable_extra_ex1_gen <-
  trial %>%
  select(trt, age, stage) %>%
  tbl_summary(by = trt) %>%
  bold_labels() %>%
  as_kable_extra()

# Example 2 (PDF via LaTeX) -----
custom_names <- c(
  "\\textbf{Characteristic}",
  "\\textbf{Drug A}\\n\\textit{N = 98}",
  "\\textbf{Drug B}\\n\\textit{N = 102}"
)
as_kable_extra_ex2_pdf <-
  trial %>%
  select(trt, age, stage) %>%
  tbl_summary(
    by = trt,
    statistic = list(all_categorical() ~ "{n} ({p}\\%)")
```



```

) %>%
bold_labels() %>%
modify_footnote(
  update = all_stat_cols() ~ "Median (IQR); n (%)"
) %>%
as_kable_extra(
  format = "latex",
  col.names = kableExtra::linebreak(custom_names, align = "c"),
  escape = FALSE
)

# Example 3 (PDF via LaTeX) -----
as_kable_extra_ex3_pdf <-
trial %>%
  select(trt, age, stage) %>%
  tbl_summary(by = trt) %>%
  bold_labels() %>%
  as_kable_extra(
    format = "latex",
    booktabs = TRUE,
    longtable = TRUE,
    linesep = ""
  ) %>%
  kableExtra::kable_styling(
    position = "left",
    latex_options = c("striped", "repeat_header"),
    stripe_color = "gray!15"
  )

```

---

as\_tibble.gtsummary    *Convert gtsummary object to a tibble*

---

## Description

Function converts a gtsummary object to a tibble.

## Usage

```

## S3 method for class 'gtsummary'
as_tibble(
  x,
  include = everything(),
  col_labels = TRUE,
  return_calls = FALSE,
  exclude = NULL,
  fmt_missing = FALSE,
  ...
)

```

**Arguments**

<code>x</code>	Object created by a function from the <code>gtsummary</code> package (e.g. <code>tbl_summary</code> or <code>tbl_regression</code> )
<code>include</code>	Commands to include in output. Input may be a vector of quoted or unquoted names. <code>tidyselect</code> and <code>gtsummary</code> select helper functions are also accepted. Default is <code>everything()</code> .
<code>col_labels</code>	Logical argument adding column labels to output tibble. Default is <code>TRUE</code> .
<code>return_calls</code>	Logical. Default is <code>FALSE</code> . If <code>TRUE</code> , the calls are returned as a list of expressions.
<code>exclude</code>	DEPRECATED
<code>fmt_missing</code>	Logical argument adding the missing value formats.
<code>...</code>	Not used

**Value**

a `tibble`

**Author(s)**

Daniel D. Sjoberg

**See Also**

Other `gtsummary` output types: `as_flex_table()`, `as_gt()`, `as_hux_table()`, `as_kable_extra()`, `as_kable()`

**Examples**

```
tbl <-
  trial %>%
  select(trt, age, grade, response) %>%
  tbl_summary(by = trt)

as_tibble(tbl)

# without column labels
as_tibble(tbl, col_labels = FALSE)
```

---

`bold_italicize_labels_levels`

*Bold or Italicize labels or levels in gtsummary tables*

---

**Description**

Bold or Italicize labels or levels in `gtsummary` tables

**Usage**`bold_labels(x)``bold_levels(x)``italicize_labels(x)``italicize_levels(x)`**Arguments**

x                    Object created using gtsummary functions

**Value**

Functions return the same class of gtsummary object supplied

**Functions**

- `bold_labels`: Bold labels in gtsummary tables
- `bold_levels`: Bold levels in gtsummary tables
- `italicize_labels`: Italicize labels in gtsummary tables
- `italicize_levels`: Italicize levels in gtsummary tables

**Example Output****Author(s)**

Daniel D. Sjöberg

**See Also**

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`

Other `tbl_uvregression` tools: `add_global_p()`, `add_q()`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_uvregression()`

**Examples**

```
tbl_bold_ital_ex <-  
  trial[c("trt", "age", "grade")] %>%  
  tbl_summary() %>%  
  bold_labels() %>%  
  bold_levels() %>%  
  italicize_labels() %>%  
  italicize_levels()
```

---

bold_p	<i>Bold significant p-values or q-values</i>
--------	--

---

### Description

Bold values below a chosen threshold (e.g. <0.05) in a gtsummary tables.

### Usage

```
bold_p(x, t = 0.05, q = FALSE)
```

### Arguments

x	Object created using gtsummary functions
t	Threshold below which values will be bold. Default is 0.05.
q	Logical argument. When TRUE will bold the q-value column rather than the p-values. Default is FALSE.

### Example Output

### Author(s)

Daniel D. Sjoberg, Esther Drill

### Examples

```
# Example 1 -----
bold_p_ex1 <-
  trial[c("age", "grade", "response", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  bold_p(t = 0.65)

# Example 2 -----
bold_p_ex2 <-
  glm(response ~ trt + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE) %>%
  bold_p(t = 0.65)
```

---

combine_terms	<i>Combine terms in a regression model</i>
---------------	--

---

### Description

The function combines terms from a regression model, and replaces the terms with a single row in the output table. The p-value is calculated using `stats::anova()`.

**Usage**

```
combine_terms(x, formula_update, label = NULL, quiet = NULL, ...)
```

**Arguments**

x	a <code>tbl_regression</code> object
formula_update	formula update passed to the <code>stats::update</code> . This updated formula is used to construct a reduced model, and is subsequently passed to <code>stats::anova()</code> to calculate the p-value for the group of removed terms. See the <code>stats::update</code> help file for proper syntax. function's <code>formula.=</code> argument
label	Option string argument labeling the combined rows
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Additional arguments passed to <code>stats::anova</code>

**Value**

`tbl_regression` object

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_regression` tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_regression](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_regression\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

**Examples**

```
# Example 1 -----
# Logistic Regression Example, LRT p-value
combine_terms_ex1 <-
  glm(
    response ~ marker + I(marker^2) + grade,
    trial[c("response", "marker", "grade")] %>% na.omit(), # keep complete cases only!
    family = binomial
  ) %>%
  tbl_regression(label = grade ~ "Grade", exponentiate = TRUE) %>%
  # collapse non-linear terms to a single row in output using anova
  combine_terms(
    formula_update = . ~ . - marker - I(marker^2),
    label = "Marker (non-linear terms)",
    test = "LRT"
  )
```

---

continuous\_summary      *Summarize a continuous variable*

---

## Description

**[Experimental]** This helper, to be used with `tbl_custom_summary()`, creates a function summarizing a continuous variable.

## Usage

```
continuous_summary(variable)
```

## Arguments

`variable`      String indicating the name of the variable to be summarized. This variable should be continuous.

## Details

When using `continuous_summary`, you can specify in the `statistic=` argument of `tbl_custom_summary()` the same continuous statistics than in `tbl_summary()`. See the *statistic argument* section of the help file of `tbl_summary()`.

## Example Output

## Author(s)

Joseph Larmarange

## See Also

Other `tbl_custom_summary` tools: `add_overall()`, `proportion_summary()`, `ratio_summary()`, `tbl_custom_summary()`

## Examples

```
# Example 1 -----
continuous_summary_ex1 <-
  trial %>%
  tbl_custom_summary(
    include = c("stage", "grade"),
    by = "trt",
    stat_fns = ~ continuous_summary("age"),
    statistic = ~ "{median} [{p25}--{p75}]",
    overall_row = TRUE,
    overall_row_label = "All stages & grades"
  ) %>%
  modify_footnote(
    update = all_stat_cols() ~ "Median age (IQR)"
  )
```

---

`custom_tidiers`*Collection of custom tidiers*

---

### Description

**[Maturing]** Collection of tidiers that can be passed to `tbl_regression()` and `tbl_uvregression()` to obtain modified results. See examples below.

### Usage

```
tidy_standardize(  
  x,  
  exponentiate = FALSE,  
  conf.level = 0.95,  
  conf.int = TRUE,  
  ...,  
  quiet = FALSE  
)
```

```
tidy_bootstrap(  
  x,  
  exponentiate = FALSE,  
  conf.level = 0.95,  
  conf.int = TRUE,  
  ...,  
  quiet = FALSE  
)
```

```
tidy_robust(  
  x,  
  exponentiate = FALSE,  
  conf.level = 0.95,  
  conf.int = TRUE,  
  vcov_estimation = NULL,  
  vcov_type = NULL,  
  vcov_args = NULL,  
  ...,  
  quiet = FALSE  
)
```

```
pool_and_tidy_mice(x, pool.args = NULL, ..., quiet = FALSE)
```

```
tidy_gam(x, conf.int = FALSE, exponentiate = FALSE, conf.level = 0.95, ...)
```

### Arguments

<code>x</code>	a regression model object
<code>exponentiate</code>	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

<code>conf.level</code>	The confidence level to use for the confidence interval if <code>conf.int = TRUE</code> . Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
<code>conf.int</code>	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
<code>...</code>	arguments passed to method; <ul style="list-style-type: none"> <li>• <code>pool_and_tidy_mice()</code>: <code>mice::tidy(x, ...)</code></li> <li>• <code>tidy_standardize()</code>: <code>effectsize::standardize_parameters(x, ...)</code></li> <li>• <code>tidy_bootstrap()</code>: <code>parameters::bootstrap_parameters(x, ...)</code></li> <li>• <code>tidy_robust()</code>: <code>parameters::model_parameters(x, ...)</code></li> </ul>
<code>quiet</code>	Logical indicating whether to print messages in console. Default is FALSE
<code>vcov_estimation, vcov_type, vcov_args</code>	arguments passed to <code>parameters::model_parameters()</code>
<code>pool.args</code>	named list of arguments passed to <code>mice::pool()</code> in <code>pool_and_tidy_mice()</code> . Default is NULL

## Details

- `tidy_standardize()` tidier to report standardized coefficients. The `effectsize` package includes a wonderful function to estimate standardized coefficients. The tidier uses the output from `effectsize::standardize_parameters()`, and merely takes the result and puts it in `broom::tidy()` format.
- `tidy_bootstrap()` tidier to report bootstrapped coefficients. The `parameters` package includes a wonderful function to estimate bootstrapped coefficients. The tidier uses the output from `parameters::bootstrap_parameters(test = "p")`, and merely takes the result and puts it in `broom::tidy()` format.
- `tidy_robust()` tidier to report robust standard errors, confidence intervals, and p-values. The `parameters` package includes a wonderful function to calculate robust standard errors, confidence intervals, and p-values. The tidier uses the output from `parameters::model_parameters()`, and merely takes the result and puts it in `broom::tidy()` format. To use this function with `tbl_regression()`, pass a function with the arguments for `tidy_robust()` populated. This is easily done using `purrr::partial()` e.g. `tbl_regression(tidy_fun = partial(tidy_robust, vcov_estimation = "CL"))`
- `pool_and_tidy_mice()` tidier to report models resulting from multiply imputed data using the `mice` package. Pass the `mice` model object *before* the model results have been pooled. See example.

Ensure your model type is compatible with the methods/functions used to estimate the model parameters before attempting to use the tidier with `tbl_regression()`

## Example Output

## Examples

```
# Example 1 -----
mod <- lm(age ~ marker + grade, trial)

tbl_stnd <- tbl_regression(mod, tidy_fun = tidy_standardize)
```



```
tbl <- tbl_regression(mod)

tidy_standardize_ex1 <-
  tbl_merge(
    list(tbl_stnd, tbl),
    tab_spanner = c("**Standardized Model**", "**Original Model**")
  )

# Example 2 -----
# use "posthoc" method for coef calculation
tidy_standardize_ex2 <-
  tbl_regression(mod, tidy_fun = purrr::partial(tidy_standardize, method = "posthoc"))

# Example 3 -----
# Multiple Imputation using the mice package
set.seed(1123)
pool_and_tidy_mice_ex3 <-
  suppressWarnings(mice::mice(trial, m = 2)) %>%
  with(lm(age ~ marker + grade)) %>%
  tbl_regression()
```

---

inline\_text.gtsummary *Report statistics from summary tables inline*

---

## Description

Report statistics from summary tables inline

## Usage

```
## S3 method for class 'gtsummary'
inline_text(x, variable, level = NULL, column = NULL, pattern = NULL, ...)
```

## Arguments

x	gtsummary object
variable	Variable name of statistic to present
level	Level of the variable to display for categorical variables. Default is NULL
column	Column name to return from x\$table_body.
pattern	String indicating the statistics to return. Uses <code>glue::glue</code> formatting. Default is NULL
...	Not used

## column + pattern

Some gtsummary tables report multiple statistics in a single cell, e.g. "{mean} ({sd})" in `tbl_summary()` or `tbl_svsummary()`. We often need to report just the mean or the SD, and that can be accomplished by using both the `column=` and `pattern=` arguments. When both of these arguments are specified, the `column` argument selects the column to report statistics from, and the `pattern` argument specifies which statistics to report, e.g. `inline_text(x, column = "stat_1", pattern = "{mean}")` reports just the mean from a `tbl_summary()`.

---

inline\_text.tbl\_cross *Report statistics from cross table inline*

---

## Description

**[Maturing]** Extracts and returns statistics from a `tbl_cross` object for inline reporting in an R markdown document. Detailed examples in the [inline\\_text vignette](#)

## Usage

```
## S3 method for class 'tbl_cross'
inline_text(x, col_level = NULL, row_level = NULL, pvalue_fun = NULL, ...)
```

## Arguments

<code>x</code>	a <code>tbl_cross</code> object
<code>col_level</code>	Level of the column variable to display. Default is <code>NULL</code> . Can also specify "p.value" for the p-value and "stat_0" for Total column.
<code>row_level</code>	Level of the row variable to display. Can also specify the 'Unknown' row. Default is <code>NULL</code> .
<code>pvalue_fun</code>	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
<code>...</code>	Not used

## Value

A string reporting results from a `gtsummary` table

## See Also

Other `tbl_cross` tools: [add\\_p.tbl\\_cross\(\)](#), [tbl\\_cross\(\)](#)

## Examples

```
tbl_cross <-
  tbl_cross(trial, row = trt, col = response) %>%
  add_p()

inline_text(tbl_cross, row_level = "Drug A", col_level = "1")
inline_text(tbl_cross, row_level = "Total", col_level = "1")
inline_text(tbl_cross, col_level = "p.value")
```

---

```
inline_text.tbl_regression
```

*Report statistics from regression summary tables inline*

---

## Description

Takes an object with class `tbl_regression`, and the location of the statistic to report and returns statistics for reporting inline in an R markdown document. Detailed examples in the [inline\\_text vignette](#)

## Usage

```
## S3 method for class 'tbl_regression'
inline_text(
  x,
  variable,
  level = NULL,
  pattern = "{estimate} ({conf.level*100}% CI {conf.low}, {conf.high}; {p.value})",
  estimate_fun = NULL,
  pvalue_fun = NULL,
  ...
)
```

## Arguments

<code>x</code>	Object created from <a href="#">tbl_regression</a>
<code>variable</code>	Variable name of statistics to present
<code>level</code>	Level of the variable to display for categorical variables. Default is <code>NULL</code> , returning the top row in the table for the variable.
<code>pattern</code>	String indicating the statistics to return. Uses <a href="#">glue::glue</a> formatting. Default is <code>"{estimate} ({conf.level }% CI {conf.low}, {conf.high}; {p.value})"</code> . All columns from <code>x\$table_body</code> are available to print as well as the confidence level ( <code>conf.level</code> ). See below for details.
<code>estimate_fun</code>	function to style model coefficient estimates. Columns <code>'estimate'</code> , <code>'conf.low'</code> , and <code>'conf.high'</code> are formatted. Default is <code>x\$inputs\$estimate_fun</code>
<code>pvalue_fun</code>	function to style p-values and/or q-values. Default is <code>function(x) style_pvalue(x, prepend_p = TRUE)</code>
<code>...</code>	Not used

## Value

A string reporting results from a `gtsummary` table

## pattern argument

The following items (and more) are available to print. Use `print(x$table_body)` to print the table the estimates are extracted from.

- `{estimate}` coefficient estimate formatted with `'estimate_fun'`

- {conf.low} lower limit of confidence interval formatted with 'estimate\_fun'
- {conf.high} upper limit of confidence interval formatted with 'estimate\_fun'
- {p.value} p-value formatted with 'pvalue\_fun'
- {N} number of observations in model
- {label} variable/variable level label

### Author(s)

Daniel D. Sjoberg

### See Also

Other `tbl_regression` tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [combine\\_terms\(\)](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_regression\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

### Examples

```
inline_text_ex1 <-
  glm(response ~ age + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE)

inline_text(inline_text_ex1, variable = age)
inline_text(inline_text_ex1, variable = grade, level = "III")
```

---

```
inline_text.tbl_summary
```

*Report statistics from summary tables inline*

---

### Description

Extracts and returns statistics from a `tbl_summary` object for inline reporting in an R markdown document. Detailed examples in the [inline\\_text vignette](#)

### Usage

```
## S3 method for class 'tbl_summary'
inline_text(
  x,
  variable,
  column = NULL,
  level = NULL,
  pattern = NULL,
  pvalue_fun = NULL,
  ...
)

## S3 method for class 'tbl_svsummary'
inline_text(
  x,
  variable,
  column = NULL,
```

```

  level = NULL,
  pattern = NULL,
  pvalue_fun = NULL,
  ...
)

```

### Arguments

x	Object created from <a href="#">tbl_summary</a>
variable	Variable name of statistic to present
column	Column name to return from x\$table_body. Can also pass the level of a by variable.
level	Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is NULL
pattern	String indicating the statistics to return. Uses <a href="#">glue::glue</a> formatting. Default is pattern shown in <code>tbl_summary()</code> output
pvalue_fun	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
...	Not used

### Value

A string reporting results from a gtsummary table

### Author(s)

Daniel D. Sjoberg

### See Also

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

### Examples

```

t1 <- trial[c("trt", "grade")] %>%
  tbl_summary(by = trt) %>%
  add_p()

inline_text(t1, variable = grade, level = "I", column = "Drug A", pattern = "{n}/{N} ({p})%")
inline_text(t1, variable = grade, column = "p.value")

```

---

```
inline_text.tbl_survfit
```

*Report statistics from survfit tables inline*

---

## Description

**[Maturing]** Extracts and returns statistics from a `tbl_survfit` object for inline reporting in an R markdown document. Detailed examples in the [inline\\_text vignette](#)

## Usage

```
## S3 method for class 'tbl_survfit'
inline_text(
  x,
  variable = NULL,
  level = NULL,
  pattern = NULL,
  time = NULL,
  prob = NULL,
  column = NULL,
  estimate_fun = x$inputs$estimate_fun,
  pvalue_fun = NULL,
  ...
)
```

## Arguments

<code>x</code>	Object created from <a href="#">tbl_survfit</a>
<code>variable</code>	Variable name of statistic to present.
<code>level</code>	Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is NULL
<code>pattern</code>	String indicating the statistics to return.
<code>time</code>	time for which to return survival probabilities.
<code>prob</code>	probability with values in (0,1)
<code>column</code>	column to print from <code>x\$table_body</code> . Columns may be selected with <code>time=</code> or <code>prob=</code> as well.
<code>estimate_fun</code>	Function to round and format estimate and confidence limits. Default is the same function used in <code>tbl_survfit()</code>
<code>pvalue_fun</code>	Function to round and format p-values. Default is <a href="#">style_pvalue</a> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
<code>...</code>	Not used

## Value

A string reporting results from a gtsummary table

**Author(s)**

Daniel D. Sjöberg

**See Also**

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

**Examples**

```
library(survival)
# fit survfit
fit1 <- survfit(Surv(ttdeath, death) ~ trt, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ 1, trial)

# summarize survfit objects
tbl1 <-
  tbl_survfit(
    fit1,
    times = c(12, 24),
    label = "Treatment",
    label_header = "**{time} Month**"
  ) %>%
  add_p()

tbl2 <-
  tbl_survfit(
    fit2,
    probs = 0.5,
    label_header = "**Median Survival**"
  )

# report results inline
inline_text(tbl1, time = 24, level = "Drug B")
inline_text(tbl1, column = p.value)
inline_text(tbl2, prob = 0.5)
```

---

 inline\_text.tbl\_uvregression

*Report statistics from regression summary tables inline*

---

**Description**

Extracts and returns statistics from a table created by the `tbl_uvregression` function for inline reporting in an R markdown document. Detailed examples in the [inline\\_text vignette](#)

**Usage**

```
## S3 method for class 'tbl_uvregression'
inline_text(
  x,
```

```

variable,
level = NULL,
pattern = "{estimate} ({{conf.level*100}}% CI {conf.low}, {conf.high}; {p.value})",
estimate_fun = NULL,
pvalue_fun = NULL,
...
)

```

### Arguments

x	Object created from <a href="#">tbl_uvregression</a>
variable	Variable name of statistics to present
level	Level of the variable to display for categorical variables. Default is NULL, returning the top row in the table for the variable.
pattern	String indicating the statistics to return. Uses <a href="#">glue::glue</a> formatting. Default is "{estimate} ({{conf.level }}% CI {conf.low},{conf.high}; {p.value})". All columns from x\$table_body are available to print as well as the confidence level (conf.level). See below for details.
estimate_fun	function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is x\$inputs\$estimate_fun
pvalue_fun	function to style p-values and/or q-values. Default is function(x) style_pvalue(x,prepend_p = TRUE)
...	Not used

### Value

A string reporting results from a gtsummary table

### pattern argument

The following items (and more) are available to print. Use `print(x$table_body)` to print the table the estimates are extracted from.

- {estimate} coefficient estimate formatted with 'estimate\_fun'
- {conf.low} lower limit of confidence interval formatted with 'estimate\_fun'
- {conf.high} upper limit of confidence interval formatted with 'estimate\_fun'
- {p.value} p-value formatted with 'pvalue\_fun'
- {N} number of observations in model
- {label} variable/variable level label

### Author(s)

Daniel D. Sjoberg

### See Also

Other `tbl_uvregression` tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_uvregression\(\)](#)



**Examples**

```

inline_text_ex1 <-
  trial[c("response", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    method.args = list(family = binomial),
    y = response,
    exponentiate = TRUE
  )

inline_text(inline_text_ex1, variable = age)
inline_text(inline_text_ex1, variable = grade, level = "III")

```

---

modify	<i>Modify column headers, footnotes, spanning headers, and table captions</i>
--------	---

---

**Description**

These functions assist with updating or adding column headers (`modify_header()`), footnotes (`modify_footnote()`), spanning headers (`modify_spanning_header()`), and table captions (`modify_caption()`). Use `show_header_names()` to learn the column names.

**Usage**

```

modify_header(
  x,
  update = NULL,
  ...,
  text_interpret = c("md", "html"),
  quiet = NULL,
  stat_by = NULL
)

modify_footnote(
  x,
  update = NULL,
  ...,
  abbreviation = FALSE,
  text_interpret = c("md", "html"),
  quiet = NULL
)

modify_spanning_header(
  x,
  update = NULL,
  ...,
  text_interpret = c("md", "html"),
  quiet = NULL
)

```

```
modify_caption(x, caption, text_interpret = c("md", "html"))
```

```
show_header_names(x = NULL, include_example = TRUE, quiet = NULL)
```

### Arguments

<code>x</code>	a gtsummary object
<code>update, ...</code>	use these arguments to assign updates to headers, spanning headers, and footnotes. See examples below. <ul style="list-style-type: none"> <li>• <code>update</code> expects a list of assignments, with the variable name or selector on the LHS of the formula, and the updated string on the RHS. Also accepts a named list.</li> <li>• ... pass individual updates outside of a list, e.g. <code>modify_header(p.value = "***P**", all_stat_cols() ~ "**{level}**")</code></li> </ul> Use the <code>show_header_names()</code> to see the column names that can be modified.
<code>text_interpret</code>	String indicates whether text will be interpreted with <code>gt::md()</code> or <code>gt::html()</code> . Must be "md" (default) or "html".
<code>quiet</code>	Logical indicating whether to print messages in console. Default is FALSE
<code>stat_by</code>	DEPRECATED, use <code>update = all_stat_cols() ~ "&lt;label&gt;"</code> instead.
<code>abbreviation</code>	Logical indicating if an abbreviation is being updated.
<code>caption</code>	a string of the table caption/title
<code>include_example</code>	logical whether to include print of <code>modify_header()</code> example

### Value

Updated gtsummary object

### `tbl_summary()`, `tbl_svysummary()`, and `tbl_cross()`

When assigning column headers, footnotes, spanning headers, and captions for these gtsummary tables, you may use {N} to insert the number of observations. `tbl_svysummary` objects additionally have {N\_unweighted} available.

When there is a stratifying `by=` argument present, the following fields are additionally available to stratifying columns: {level}, {n}, and {p} ({n\_unweighted} and {p\_unweighted} for `tbl_svysummary` objects)

Syntax follows `glue::glue()`, e.g. `all_stat_cols() ~ "**{level}**", N = {n}"`.

### `tbl_regression()`

When assigning column headers for `tbl_regression` tables, you may use {N} to insert the number of observations, and {N\_event} for the number of events (when applicable).

### captions

Captions are assigned based on output type.

- `gt::gt(caption=)`
- `flextable::set_caption(caption=)`
- `huxtable::set_caption(value=)`
- `knitr::kable(caption=)`

## Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

Other `tbl_svysummary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svysummary()`, `add_q()`, `add_stat_label()`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_svysummary()`

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`

Other `tbl_uvregression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_uvregression()`

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`, `tbl_survfit()`

### Examples

```
# create summary table
tbl <- trial[c("age", "grade", "trt")] %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_p()

# print the column names that can be modified
show_header_names(tbl)

# Example 1 -----
# updating column headers, footnote, and table caption
modify_ex1 <- tbl %>%
  modify_header(label = "**Variable**", p.value = "**P**") %>%
  modify_footnote(all_stat_cols() ~ "median (IQR) for Age; n (%) for Grade") %>%
  modify_caption("**Patient Characteristics** (N = {N})")

# Example 2 -----
# updating headers, remove all footnotes, add spanning header
modify_ex2 <- tbl %>%
  modify_header(all_stat_cols() ~ "**{level}**", N = {n} ({style_percent(p)}%)") %>%
  # use `modify_footnote(everything() ~ NA, abbreviation = TRUE)` to delete abbrev. footnotes
  modify_footnote(update = everything() ~ NA) %>%
  modify_spanning_header(all_stat_cols() ~ "**Treatment Received**")

# Example 3 -----
# updating an abbreviation in table footnote
modify_ex3 <-
  glm(response ~ age + grade, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE) %>%
```

```
modify_footnote(ci = "CI = Credible Interval", abbreviation = TRUE)
```

---

modify_cols_merge	<i>Modify Column Merging</i>
-------------------	------------------------------

---

## Description

**[Experimental]** Merge two or more columns in a gtsummary table. Use `show_header_names()` to print underlying column names.

## Usage

```
modify_cols_merge(x, pattern, rows = NULL)
```

## Arguments

x	gtsummary object
pattern	glue syntax string indicating how to merge columns in <code>x\$table_body</code> . For example, to construct a confidence interval use <code>"{conf.low},{conf.high}"</code> .
rows	predicate expression to select rows in <code>x\$table_body</code> . Can be used to style footnote, formatting functions, missing symbols, and text formatting. Default is <code>NULL</code> . See details below.

## Value

gtsummary table

## Details

1. Calling this function merely records the instructions to merge columns. The actual merging occurs when the gtsummary table is printed or converted with a function like `as_gt()`.
2. Because the column merging is delayed, it is recommended to perform major modifications to the table, such as those with `tbl_merge()` and `tbl_stack()`, before assigning merging instructions. Otherwise, unexpected formatting may occur in the final table.

## Future Updates

There are planned updates to the implementation of this function with respect to the `pattern=` argument. Currently, this function replaces a numeric column with a formatted character column following `pattern=`. Once `gt::cols_merge()` gains the `rows=` argument the implementation will be updated to use it, which will keep numeric columns numeric. For the *vast majority* of users, *the planned change will be go unnoticed*.

## Example Output

## See Also

Other Advanced modifiers: [modify\\_column\\_alignment\(\)](#), [modify\\_column\\_hide\(\)](#), [modify\\_fmt\\_fun\(\)](#), [modify\\_table\\_body\(\)](#), [modify\\_table\\_styling\(\)](#)

**Examples**

```
# Example 1 -----
modify_cols_merge_ex1 <-
  trial %>%
  select(age, marker, trt) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_p(all_continuous() ~ "t.test",
        pvalue_fun = ~style_pvalue(., prepend_p = TRUE)) %>%
  modify_fmt_fun(statistic ~ style_sigfig) %>%
  modify_cols_merge(pattern = "t = {statistic}; {p.value}") %>%
  modify_header(statistic ~ "**t-test**")

# Example 2 -----
modify_cols_merge_ex2 <-
  lm(marker ~ age + grade, trial) %>%
  tbl_regression() %>%
  modify_cols_merge(
    pattern = "{estimate} ({ci})",
    rows = !is.na(estimate)
  )
```

---

 modify\_column\_alignment

*Modify Column Alignment*


---

**Description**

**[Maturing]** Update column alignment/justification in a gtsummary table.

**Usage**

```
modify_column_alignment(x, columns, align = c("left", "right", "center"))
```

**Arguments**

x	gtsummary object
columns	vector or selector of columns in x\$table_body
align	string indicating alignment of column, must be one of c("left", "right", "center")

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other Advanced modifiers: [modify\\_cols\\_merge\(\)](#), [modify\\_column\\_hide\(\)](#), [modify\\_fmt\\_fun\(\)](#), [modify\\_table\\_body\(\)](#), [modify\\_table\\_styling\(\)](#)

**Examples**

```
# Example 1 -----
tbl <-
  lm(age ~ marker + grade, trial) %>%
  tbl_regression() %>%
  modify_column_alignment(columns = everything(), align = "left")
```

---

modify_column_hide	<i>Modify Hidden Columns</i>
--------------------	------------------------------

---

### Description

**[Maturing]** Use these functions to hide or unhide columns in a gtsummary table.

### Usage

```
modify_column_hide(x, columns)

modify_column_unhide(x, columns)
```

### Arguments

x	gtsummary object
columns	vector or selector of columns in x\$table_body

### Example Output

### See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other Advanced modifiers: [modify\\_cols\\_merge\(\)](#), [modify\\_column\\_alignment\(\)](#), [modify\\_fmt\\_fun\(\)](#), [modify\\_table\\_body\(\)](#), [modify\\_table\\_styling\(\)](#)

### Examples

```
# Example 1 -----
# hide 95% CI, and replace with standard error
modify_column_hide_ex1 <-
  lm(age ~ marker + grade, trial) %>%
  tbl_regression() %>%
  modify_column_hide(columns = ci) %>%
  modify_column_unhide(columns = std.error)
```

---

modify_fmt_fun	<i>Modify Formatting Functions</i>
----------------	------------------------------------

---

### Description

**[Maturing]** Use this function to update the way numeric columns and rows of .\$table\_body are formatted

### Usage

```
modify_fmt_fun(x, update, rows = NULL)
```

**Arguments**

x	gtsummary object
update	list of formulas or a single formula specifying the updated formatting function. The LHS specifies the column(s) to be updated, and the RHS is the updated formatting function.
rows	predicate expression to select rows in x\$table_body. Default is NULL. See details below.

**Example Output****rows argument**

The rows argument accepts a predicate expression that is used to specify rows to apply formatting. The expression must evaluate to a logical when evaluated in x\$table\_body. For example, to apply formatting to the age rows pass rows = variable == "age". A vector of row numbers is NOT acceptable.

A couple of things to note when using the rows= argument.

1. You can use saved objects to create the predicate argument, e.g. rows = variable == letters[1].
2. The saved object cannot share a name with a column in x\$table\_body. The reason for this is that in tbl\_merge() the columns are renamed, and the renaming process cannot disambiguate the variable column from an external object named variable in the following expression  
rows = .data\$variable = .env\$variable.

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other Advanced modifiers: [modify\\_cols\\_merge\(\)](#), [modify\\_column\\_alignment\(\)](#), [modify\\_column\\_hide\(\)](#), [modify\\_table\\_body\(\)](#), [modify\\_table\\_styling\(\)](#)

**Examples**

```
# Example 1 -----
# show 'grade' p-values to 3 decimal places
modify_fmt_fun_ex1 <-
  lm(age ~ marker + grade, trial) %>%
  tbl_regression() %>%
  modify_fmt_fun(
    update = p.value ~ function(x) style_pvalue(x, digits = 3),
    rows = variable == "grade"
  )
```

---

modify_table_body	<i>Modify Table Body</i>
-------------------	--------------------------

---

## Description

**[Maturing]** Function is for advanced manipulation of gtsummary tables. It allow users to modify the `.$table_body` data frame included in each gtsummary object.

If a new column is added to the table, default printing instructions will then be added to `.$table_styling`. By default, columns are hidden. To show a column, add a column header with `modify_header()`.

## Usage

```
modify_table_body(x, fun, ...)
```

## Arguments

x	gtsummary object
fun	A function or formula. If a <i>function</i> , it is used as is. If a <i>formula</i> , e.g. <code>fun = ~ .x %&gt;% arrange(variable)</code> , it is converted to a function. The argument passed to <code>fun=</code> is <code>x\$table_body</code> .
...	Additional arguments passed on to the mapped function

## Example Output

## See Also

`modify_table_styling()`

See [gtsummary internals vignette](#)

Other Advanced modifiers: [modify\\_cols\\_merge\(\)](#), [modify\\_column\\_alignment\(\)](#), [modify\\_column\\_hide\(\)](#), [modify\\_fmt\\_fun\(\)](#), [modify\\_table\\_styling\(\)](#)

## Examples

```
# Example 1 -----
# Add number of cases and controls to regression table
modify_table_body_ex1 <-
  trial %>%
  select(response, age, marker) %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
    exponentiate = TRUE,
    hide_n = TRUE
  ) %>%
  # adding number of non-events to table
  modify_table_body(
    ~ .x %>%
    dplyr::mutate(N_nonevent = N_obs - N_event) %>%
```



```

      dplyr::relocate(c(N_event, N_nonevent), .before = estimate)
    ) %>%
    # assigning header labels
    modify_header(N_nonevent = "***Control N**", N_event = "***Case N**") %>%
    modify_fmt_fun(c(N_event, N_nonevent) ~ style_number)

```

---

modify\_table\_styling *Modify Table Styling*

---

## Description

This is a function meant for advanced users to gain more control over the characteristics of the resulting gtsummary table by directly modifying `.$table_styling`

## Usage

```

modify_table_styling(
  x,
  columns,
  rows = NULL,
  label = NULL,
  spanning_header = NULL,
  hide = NULL,
  footnote = NULL,
  footnote_abbrev = NULL,
  align = NULL,
  missing_symbol = NULL,
  fmt_fun = NULL,
  text_format = NULL,
  undo_text_format = FALSE,
  text_interpret = c("md", "html"),
  cols_merge_pattern = NULL
)

```

## Arguments

x	gtsummary object
columns	vector or selector of columns in <code>x\$table_body</code>
rows	predicate expression to select rows in <code>x\$table_body</code> . Can be used to style footnote, formatting functions, missing symbols, and text formatting. Default is NULL. See details below.
label	string of column label(s)
spanning_header	string with text for spanning header
hide	logical indicating whether to hide column from output
footnote	string with text for footnote
footnote_abbrev	string with abbreviation definition, e.g. "CI = Confidence Interval"
align	string indicating alignment of column, must be one of <code>c("left", "right", "center")</code>

missing_symbol	string indicating how missing values are formatted.
fmt_fun	function that formats the statistics in the columns/rows in columns= and rows=
text_format	string indicated which type of text formatting to apply to the rows and columns. Must be one of c("bold", "italic", "indent", "indent2"). Do not assign both "indent" and "indent2" to the same cell.
undo_text_format	rarely used. Logical that undoes the indent, bold, and italic styling when TRUE
text_interpret	string, must be one of "md" or "html"
cols_merge_pattern	<b>[Experimental]</b> glue-syntax string indicating how to merge columns in x\$table_body. For example, to construct a confidence interval use "{conf.low}, {conf.high}". The first column listed in the pattern string must match the single column name passed in columns=.

## Details

Review the [gtsummary definition](#) vignette for information on `.$table_styling` objects.

## rows argument

The rows argument accepts a predicate expression that is used to specify rows to apply formatting. The expression must evaluate to a logical when evaluated in `x$table_body`. For example, to apply formatting to the age rows pass `rows = variable == "age"`. A vector of row numbers is NOT acceptable.

A couple of things to note when using the rows= argument.

1. You can use saved objects to create the predicate argument, e.g. `rows = variable == letters[1]`.
2. The saved object cannot share a name with a column in `x$table_body`. The reason for this is that in `tbl_merge()` the columns are renamed, and the renaming process cannot disambiguate the variable column from an external object named variable in the following expression `rows = .data$variable = .env$variable`.

## cols\_merge\_pattern argument

There are planned updates to the implementation of column merging. Currently, this function replaces the numeric column with a formatted character column following `cols_merge_pattern=`. Once `gt::cols_merge()` gains the rows= argument the implementation will be updated to use it, which will keep numeric columns numeric. For the vast majority of users, *the planned change will be go unnoticed*.

If this functionality is used in conjunction with `tbl_stack()` (which includes `tbl_uvregression()`), there is potential issue with printing. When columns are stack AND when the column-merging is defined with a quosure, you may run into issues due to the loss of the environment when 2 or more quosures are combined. If the expression version of the quosure is the same as the quosure (i.e. no evaluated objects), there should be no issues. Regardless, this argument is used internally with care, and it is *not* recommended for users.

## See Also

`modify_table_body()`

See [gtsummary internals vignette](#)

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other Advanced modifiers: `modify_cols_merge()`, `modify_column_alignment()`, `modify_column_hide()`, `modify_fmt_fun()`, `modify_table_body()`

---

plot *Plot Regression Coefficients*

---

### Description

The `plot()` function extracts `x$table_body` and passes the it to `GGally::ggcoef_plot()` along with a formatting options.

### Usage

```
## S3 method for class 'tbl_regression'
plot(x, remove_header_rows = TRUE, remove_reference_rows = FALSE, ...)

## S3 method for class 'tbl_uvregression'
plot(x, remove_header_rows = TRUE, remove_reference_rows = FALSE, ...)
```

### Arguments

`x` 'tbl\_regression' or 'tbl\_uvregression' object

`remove_header_rows` logical indicating whether to remove header rows for categorical variables. Default is TRUE

`remove_reference_rows` logical indicating whether to remove reference rows for categorical variables. Default is FALSE.

`...` arguments passed to `GGally::ggcoef_plot(...)`

### Details

**[Experimental]**

### Value

a ggplot

### Examples

```
glm(response ~ marker + grade, trial, family = binomial) %>%
  tbl_regression(
    add_estimate_to_reference_rows = TRUE,
    exponentiate = TRUE
  ) %>%
  plot()
```

---

proportion\_summary     *Summarize a proportion*

---

### Description

**[Experimental]** This helper, to be used with `tbl_custom_summary()`, creates a function computing a proportion and its confidence interval.

### Usage

```
proportion_summary(
  variable,
  value,
  weights = NULL,
  na.rm = TRUE,
  conf.level = 0.95,
  method = c("wilson", "wilson.no.correct", "exact", "asymptotic")
)
```

### Arguments

variable	String indicating the name of the variable from which the proportion will be computed.
value	Value (or list of values) of <code>variable</code> to be taken into account in the numerator.
weights	Optional string indicating the name of a weighting variable. If <code>NULL</code> , all observations will be assumed to have a weight equal to 1.
na.rm	Should missing values be removed before computing the proportion? (default is <code>TRUE</code> )
conf.level	Confidence level for the returned confidence interval. Must be strictly greater than 0 and less than 1. Default to 0.95, which corresponds to a 95 percent confidence interval.
method	Confidence interval method. Must be one of <code>c("wilson", "wilson.no.correct", "exact", "asymptotic")</code> . See details below.

### Details

Computed statistics:

- `{n}` numerator, (weighted) number of observations equal to `values`
- `{N}` denominator, (weighted) number of observations
- `{prop}` proportion, i.e.  $n/N$
- `{conf.low}` lower confidence interval
- `{conf.high}` upper confidence interval

Methods `c("wilson", "wilson.no.correct")` are calculated with `stats::prop.test()` (with `correct = c(TRUE, FALSE)`). The default method, "wilson", includes the Yates continuity correction. Methods `c("exact", "asymptotic")` are calculated with `Hmisc::binconf()` and the corresponding method.

**Example Output****Author(s)**

Joseph Larmarange

**See Also**

Other `tbl_custom_summary` tools: [add\\_overall\(\)](#), [continuous\\_summary\(\)](#), [ratio\\_summary\(\)](#), [tbl\\_custom\\_summary\(\)](#)

**Examples**

```
# Example 1 -----
proportion_summary_ex1 <-
  Titanic %>%
  as.data.frame() %>%
  tbl_custom_summary(
    include = c("Age", "Class"),
    by = "Sex",
    stat_fns = ~ proportion_summary("Survived", "Yes", weights = "Freq"),
    statistic = ~ "{prop}% ({n}/{N}) [{conf.low}-{conf.high}]",
    digits = ~ list(
      function(x) {style_percent(x, digits = 1)},
      0, 0, style_percent, style_percent
    ),
    overall_row = TRUE,
    overall_row_last = TRUE
  ) %>%
  bold_labels() %>%
  modify_footnote(
    update = all_stat_cols() ~ "Proportion (%) of survivors (n/N) [95% CI]"
  )
```

ratio\_summary

*Summarize the ratio of two variables***Description**

**[Experimental]** This helper, to be used with `tbl_custom_summary()`, creates a function computing the ratio of two continuous variables and its confidence interval.

**Usage**

```
ratio_summary(numerator, denominator, na.rm = TRUE, conf.level = 0.95)
```

**Arguments**

numerator	String indicating the name of the variable to be summed for computing the numerator.
denominator	String indicating the name of the variable to be summed for computing the denominator.

<code>na.rm</code>	Should missing values be removed before summing the numerator and the denominator? (default is TRUE)
<code>conf.level</code>	Confidence level for the returned confidence interval. Must be strictly greater than 0 and less than 1. Default to 0.95, which corresponds to a 95 percent confidence interval.

## Details

Computed statistics:

- `{num}` sum of the variable defined by numerator
- `{denom}` sum of the variable defined by denominator
- `{ratio}` ratio of num by denom
- `{conf.low}` lower confidence interval
- `{conf.high}` upper confidence interval

Confidence interval is computed with `stats::poisson.test()`, if and only if num is an integer.

## Example Output

### Author(s)

Joseph Larmarange

### See Also

Other `tbl_custom_summary` tools: [add\\_overall\(\)](#), [continuous\\_summary\(\)](#), [proportion\\_summary\(\)](#), [tbl\\_custom\\_summary\(\)](#)

## Examples

```
# Example 1 -----
ratio_summary_ex1 <-
  trial %>%
  tbl_custom_summary(
    include = c("stage", "grade"),
    by = "trt",
    stat_fns = ~ ratio_summary("response", "ttdeath"),
    statistic = ~ "{ratio} [{conf.low}; {conf.high}] ({num}/{denom})",
    digits = ~ c(3, 2, 2, 0, 0),
    overall_row = TRUE,
    overall_row_label = "All stages & grades"
  ) %>%
  bold_labels() %>%
  modify_footnote(
    update = all_stat_cols() ~ "Ratio [95% CI] (n/N)"
  )
```

---

remove_row_type	<i>Remove rows by type</i>
-----------------	----------------------------

---

## Description

Removes either the header, reference, or missing rows from a gtsummary table.

## Usage

```
remove_row_type(  
  x,  
  variables = everything(),  
  type = c("header", "reference", "missing")  
)
```

## Arguments

x	gtsummary object
variables	variables to to remove rows from. Default is everything()
type	type of row to remove. Must be one of c("header", "reference", "missing")

## Example Output

## See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

## Examples

```
# Example 1 -----  
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)  
remove_row_type_ex1 <-  
  trial %>%  
  select(trt, age) %>%  
  mutate(  
    age60 = case_when(age < 60 ~ "<60", age >= 60 ~ "60+")  
  ) %>%  
  tbl_summary(by = trt, missing = "no") %>%  
  remove_row_type(age60, type = "header")
```

---

select\_helpers      *Select helper functions*

---

### Description

Set of functions to supplement the tidymodels set of functions for selecting columns of data frames (and other items as well).

- `all_continuous()` selects continuous variables
- `all_continuous2()` selects only type "continuous2"
- `all_categorical()` selects categorical (including "dichotomous") variables
- `all_dichotomous()` selects only type "dichotomous"
- `all_tests()` selects variables by the name of the test performed
- `all_stat_cols()` selects columns from `tbl_summary/tbl_svsummary` object with summary statistics (i.e. "stat\_0", "stat\_1", "stat\_2", etc.)
- `all_interaction()` selects interaction terms from a regression model
- `all_intercepts()` selects intercept terms from a regression model
- `all_contrasts()` selects variables in regression model based on their type of contrast

### Usage

```
all_continuous(continuous2 = TRUE)
```

```
all_continuous2()
```

```
all_categorical(dichotomous = TRUE)
```

```
all_dichotomous()
```

```
all_tests(tests = NULL)
```

```
all_stat_cols(stat_0 = TRUE)
```

```
all_interaction()
```

```
all_intercepts()
```

```
all_contrasts(contrasts_type = NULL)
```

### Arguments

`continuous2` Logical indicating whether to include continuous2 variables. Default is TRUE

`dichotomous` Logical indicating whether to include dichotomous variables. Default is TRUE

`tests` string indicating the test type of the variables to select, e.g. select all variables being compared with "t.test"

`stat_0` When FALSE, will not select the "stat\_0" column. Default is TRUE

`contrasts_type` type of contrast to select. When NULL, all variables with a contrast will be selected. Default is NULL. Select among contrast types `c("treatment", "sum", "poly", "helmert", "o`



**Value**

A character vector of column names selected

**Example Output****See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

**Examples**

```
select_ex1 <-
  trial %>%
  select(age, response, grade) %>%
  tbl_summary(
    statistic = all_continuous() ~ "{mean} ({sd})",
    type = all_dichotomous() ~ "categorical"
  )
```

---

separate\_p\_footnotes *Create footnotes for individual p-values*

---

**Description**

**[Experimental]** The usual presentation of footnotes for p-values on a gtsummary table is to have a single footnote that lists all statistical tests that were used to compute p-values on a given table. The `separate_p_footnotes()` function separates aggregated p-value footnotes to individual footnotes that denote the specific test used for each of the p-values.

**Usage**

```
separate_p_footnotes(x)
```

**Arguments**

x object with class "tbl\_summary" or "tbl\_svsummary"

**Example Output****See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_svsummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svsummary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_svsummary\(\)](#)

**Examples**

```

separate_p_footnotes_ex1 <-
  trial %>%
  select(trt, age, grade) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  separate_p_footnotes()

```

---

```

set_gtsummary_theme   Set a gtsummary theme

```

---

**Description**

**[Maturing]** Use this function to set preferences for the display of gtsummary tables. The default formatting and styling throughout the gtsummary package are taken from the published reporting guidelines of the top four urology journals: European Urology, The Journal of Urology, Urology and the British Journal of Urology International. Use this function to change the default reporting style to match another journal, or your own personal style.

**Usage**

```

set_gtsummary_theme(x)

reset_gtsummary_theme()

```

**Arguments**

x                    A gtsummary theme function, e.g. theme\_gtsummary\_journal(), or a named list defining a gtsummary theme. See details below.

**Example Output****See Also**

[Themes vignette](#)

Available [gtsummary themes](#)

**Examples**

```

# Setting JAMA theme for gtsummary
set_gtsummary_theme(theme_gtsummary_journal("jama"))
# Themes can be combined by including more than one
set_gtsummary_theme(theme_gtsummary_compact())

set_gtsummary_theme_ex1 <-
  trial %>%
  dplyr::select(age, grade, trt) %>%
  tbl_summary(by = trt) %>%
  add_stat_label() %>%

```

```

as_gt()

# reset gtsummary theme
reset_gtsummary_theme()

```

---

sort_filter_p	<i>Sort and filter variables in table by p-values</i>
---------------	---

---

## Description

Sort and filter variables in table by p-values

## Usage

```

sort_p(x, q = FALSE)

filter_p(x, q = FALSE, t = 0.05)

```

## Arguments

x	An object created using gtsummary functions
q	Logical argument. When TRUE will the q-value column is used
t	p-values/q-values less than or equal to this threshold will be retained. Default is 0.05

## Example Output

## Author(s)

Karissa Whiting, Daniel D. Sjoberg

## Examples

```

# Example 1 -----
sort_filter_p_ex1 <-
  trial %>%
  select(age, grade, response, trt) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  filter_p(t = 0.8) %>%
  sort_p()

# Example 2 -----
sort_p_ex2 <-
  glm(response ~ trt + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE) %>%
  sort_p()

```

---

style_number	<i>Style numbers</i>
--------------	----------------------

---

**Description**

Style numbers

**Usage**

```
style_number(
  x,
  digits = 0,
  big.mark = NULL,
  decimal.mark = NULL,
  scale = 1,
  ...
)
```

**Arguments**

x	Numeric vector
digits	Integer or vector of integers specifying the number of digits to round $x=$ . When vector is passed, each integer is mapped 1:1 to the numeric values in $x$
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", except when decimal.mark = " " when the default is a space.
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or <code>getOption("OutDec")</code>
scale	A scaling factor: $x$ will be multiplied by <code>scale</code> before formatting.
...	Other arguments passed on to <code>base::format()</code>

**Value**

formatted character vector

**See Also**

Other style tools: [style\\_percent\(\)](#), [style\\_pvalue\(\)](#), [style\\_ratio\(\)](#), [style\\_sigfig\(\)](#)

**Examples**

```
c(0.111, 12.3) %>% style_number(digits = 1)
c(0.111, 12.3) %>% style_number(digits = c(1, 0))
```

---

style_percent	<i>Style percentages</i>
---------------	--------------------------

---

**Description**

Style percentages

**Usage**

```
style_percent(
  x,
  symbol = FALSE,
  digits = 0,
  big.mark = NULL,
  decimal.mark = NULL,
  ...
)
```

**Arguments**

x	numeric vector of percentages
symbol	Logical indicator to include percent symbol in output. Default is FALSE.
digits	number of digits to round large percentages (i.e. greater than 10%). Smaller percentages are rounded to digits + 1 places. Default is 0
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", except when decimal.mark = ".", when the default is a space.
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
...	Other arguments passed on to base::format()

**Value**

A character vector of styled percentages

**Author(s)**

Daniel D. Sjoberg

**See Also**

See Table Gallery [vignette](#) for example

Other style tools: [style\\_number\(\)](#), [style\\_pvalue\(\)](#), [style\\_ratio\(\)](#), [style\\_sigfig\(\)](#)

**Examples**

```
percent_vals <- c(-1, 0, 0.0001, 0.005, 0.01, 0.10, 0.45356, 0.99, 1.45)
style_percent(percent_vals)
style_percent(percent_vals, symbol = TRUE, digits = 1)
```

---

style_pvalue	<i>Style p-values</i>
--------------	-----------------------

---

**Description**

Style p-values

**Usage**

```
style_pvalue(
  x,
  digits = 1,
  prepend_p = FALSE,
  big.mark = NULL,
  decimal.mark = NULL,
  ...
)
```

**Arguments**

x	Numeric vector of p-values.
digits	Number of digits large p-values are rounded. Must be 1, 2, or 3. Default is 1.
prepend_p	Logical. Should 'p=' be prepended to formatted p-value. Default is FALSE
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", except when decimal.mark = " " when the default is a space.
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or <code>getOption("OutDec")</code>
...	Other arguments passed on to <code>base::format()</code>

**Value**

A character vector of styled p-values

**Author(s)**

Daniel D. Sjoberg

**See Also**

See `tbl_summary` [vignette](#) for examples

Other style tools: [style\\_number\(\)](#), [style\\_percent\(\)](#), [style\\_ratio\(\)](#), [style\\_sigfig\(\)](#)

**Examples**

```
pvals <- c(
  1.5, 1, 0.999, 0.5, 0.25, 0.2, 0.197, 0.12, 0.10, 0.0999, 0.06,
  0.03, 0.002, 0.001, 0.00099, 0.0002, 0.00002, -1
)
style_pvalue(pvals)
style_pvalue(pvals, digits = 2, prepend_p = TRUE)
```

---

`style_ratio`*Style significant figure-like rounding for ratios*

---

### Description

When reporting ratios, such as relative risk or an odds ratio, we'll often want the rounding to be similar on each side of the number 1. For example, if we report an odds ratio of 0.95 with a confidence interval of 0.70 to 1.24, we would want to round to two decimal places for all values. In other words, 2 significant figures for numbers less than 1 and 3 significant figures 1 and larger. `style_ratio()` performs significant figure-like rounding in this manner.

### Usage

```
style_ratio(x, digits = 2, big.mark = NULL, decimal.mark = NULL, ...)
```

### Arguments

<code>x</code>	Numeric vector
<code>digits</code>	Integer specifying the number of significant digits to display for numbers below 1. Numbers larger than 1 will be <code>digits + 1</code> . Default is <code>digits = 2</code> .
<code>big.mark</code>	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is <code>" , "</code> , except when <code>decimal.mark = " , "</code> when the default is a space.
<code>decimal.mark</code>	The character to be used to indicate the numeric decimal point. Default is <code>" . "</code> or <code>getOption("OutDec")</code>
<code>...</code>	Other arguments passed on to <code>base::format()</code>

### Value

A character vector of styled ratios

### Author(s)

Daniel D. Sjoberg

### See Also

Other style tools: [style\\_number\(\)](#), [style\\_percent\(\)](#), [style\\_pvalue\(\)](#), [style\\_sigfig\(\)](#)

### Examples

```
c(
  0.123, 0.9, 1.1234, 12.345, 101.234, -0.123,
  -0.9, -1.1234, -12.345, -101.234
) %>%
  style_ratio()
```

---

 style\_sigfig

*Style significant figure-like rounding*


---

### Description

Converts a numeric argument into a string that has been rounded to a significant figure-like number. Scientific notation output is avoided, however, and additional significant figures may be displayed for large numbers. For example, if the number of significant digits requested is 2, 123 will be displayed (rather than 120 or  $1.2 \times 10^2$ ).

### Usage

```
style_sigfig(
  x,
  digits = 2,
  scale = 1,
  big.mark = NULL,
  decimal.mark = NULL,
  ...
)
```

### Arguments

x	Numeric vector
digits	Integer specifying the minimum number of significant digits to display
scale	A scaling factor: x will be multiplied by scale before formatting.
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", except when decimal.mark = ".", when the default is a space.
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or <code>getOption("OutDec")</code>
...	Other arguments passed on to <code>base::format()</code>

### Details

If 2 sig figs are input, the number is rounded to 2 decimal places when  $\text{abs}(x) < 1$ , 1 decimal place when  $\text{abs}(x) \geq 1$  &  $\text{abs}(x) < 10$ , and to the nearest integer when  $\text{abs}(x) \geq 10$ .

### Value

A character vector of styled numbers

### Author(s)

Daniel D. Sjoberg

### See Also

Other style tools: [style\\_number\(\)](#), [style\\_percent\(\)](#), [style\\_pvalue\(\)](#), [style\\_ratio\(\)](#)



**Examples**

```
c(0.123, 0.9, 1.1234, 12.345, -0.123, -0.9, -1.1234, -132.345, NA, -0.001) %>%
  style_sigfig()
```

---

tbl\_butcher

*Reduce size of gtsummary objects*

---

**Description**

Some gtsummary objects can become large and the size becomes cumbersome when working with the object. The function removes all elements from a gtsummary object, except those required to print the table. This may result in gtsummary functions that add information or modify the table, such as `add_global_p()`, will no longer execute after the excess elements have been removed (aka butchered). Of note, the majority of `inline_text()` calls will continue to execute properly.

**Usage**

```
tbl_butcher(x)
```

**Arguments**

x                    a gtsummary object

**Value**

a gtsummary object

**Examples**

```
tbl_large <-
  trial %>%
  tbl_uvregression(
    y = age,
    method = lm
  )

tbl_butchered <-
  tbl_large %>%
  tbl_butcher()

# size comparison
object.size(tbl_large) %>% format(units = "Mb")
object.size(tbl_butchered) %>% format(units = "Mb")
```

---

tbl\_continuous                      *Summarize a continuous variable*


---

### Description

**[Experimental]** Summarize a continuous variable by one or more categorical variables

### Usage

```
tbl_continuous(
  data,
  variable,
  include = everything(),
  digits = NULL,
  by = NULL,
  statistic = NULL,
  label = NULL
)
```

### Arguments

data	A data frame
variable	Variable name of the continuous column to be summarized
include	variables to include in the summary table. Default is everything()
digits	List of formulas specifying the number of decimal places to round continuous summary statistics. If not specified, an appropriate number of decimals to round statistics will be guessed based on the the variable's distribution.
by	A column name (quoted or unquoted) in data. Summary statistics will be calculated separately for each level of the by variable (e.g. by = trt). If NULL, summary statistics are calculated using all observations. To stratify a table by two or more variables, use tbl_strata()
statistic	List of formulas specifying types of summary statistics to display for each variable. The default is everything() ~ {median} ({p25}, {p75})
label	List of formulas specifying variables labels, e.g. list(age ~ "Age", stage ~ "Path T Stage"). If a variable's label is not specified here, the label attribute (attr(data\$age, "label")) is used. If attribute label is NULL, the variable name will be used.

### Value

a gtsummary table

### Example Output

### See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

**Examples**

```
# Example 1 -----
tbl_continuous_ex1 <-
  tbl_continuous(
    data = trial,
    variable = age,
    by = trt,
    include = grade
  )

# Example 2 -----
tbl_continuous_ex2 <-
  tbl_continuous(
    data = trial,
    variable = age,
    include = c(trt, grade)
  )
```

tbl\_cross

*Create a cross table of summary statistics***Description**

The function creates a cross table of two categorical variables.

**Usage**

```
tbl_cross(
  data,
  row = NULL,
  col = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  percent = c("none", "column", "row", "cell"),
  margin = c("column", "row"),
  missing = c("ifany", "always", "no"),
  missing_text = "Unknown",
  margin_text = "Total"
)
```

**Arguments**

data	A data frame
row	A column name in data= to be used for the rows of cross table.
col	A column name in data= to be used for the columns of cross table.
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute ( <code>attr(data\$age, "label")</code> ) is used. If attribute label is NULL, the variable name will be used.

statistic	A string with the statistic name in curly brackets to be replaced with the numeric statistic (see <code>glue::glue</code> ). The default is <code>{n}</code> . If percent argument is "column", "row", or "cell", default is <code>{n} ({p}%)</code> .
digits	Specifies the number of decimal places to round the summary statistics. By default integers are shown to the zero decimal places, and percentages are formatted with <code>style_percent()</code> . If you would like to modify either of these, pass a vector of integers indicating the number of decimal places to round the statistics. For example, if the statistic being calculated is <code>{n} ({p}%)</code> and you want the percent rounded to 2 decimal places use <code>digits = c(0, 2)</code> . User may also pass a styling function: <code>digits = style_sigfig</code>
percent	Indicates the type of percentage to return. Must be one of "none", "column", "row", or "cell". Default is "cell" when <code>{N}</code> or <code>{p}</code> is used in statistic.
margin	Indicates which margins to add to the table. Default is <code>c("row", "column")</code> . Use <code>margin = NULL</code> to suppress both row and column margins.
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
margin_text	Text to display for margin totals. Default is "Total"

**Value**

A `tbl_cross` object

**Example Output****Author(s)**

Karissa Whiting, Daniel D. Sjoberg

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_cross` tools: [add\\_p.tbl\\_cross\(\)](#), [inline\\_text.tbl\\_cross\(\)](#)

**Examples**

```
# Example 1 -----
tbl_cross_ex1 <-
  trial %>%
  tbl_cross(row = trt, col = response)

# Example 2 -----
tbl_cross_ex2 <-
  trial %>%
  tbl_cross(row = stage, col = trt, percent = "cell") %>%
  add_p()
```

---

tbl\_custom\_summary      *Create a table of summary statistics using a custom summary function*


---

## Description

**[Experimental]** The `tbl_custom_summary()` function calculates descriptive statistics for continuous, categorical, and dichotomous variables. This function is similar to `tbl_summary()` but allows you to provide a custom function in charge of computing the statistics (see Details).

## Usage

```
tbl_custom_summary(
  data,
  by = NULL,
  label = NULL,
  stat_fns,
  statistic,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  include = everything(),
  overall_row = FALSE,
  overall_row_last = FALSE,
  overall_row_label = NULL
)
```

## Arguments

<code>data</code>	A data frame
<code>by</code>	A column name (quoted or unquoted) in <code>data</code> . Summary statistics will be calculated separately for each level of the <code>by</code> variable (e.g. <code>by = trt</code> ). If <code>NULL</code> , summary statistics are calculated using all observations. To stratify a table by two or more variables, use <code>tbl_strata()</code>
<code>label</code>	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute ( <code>attr(data\$age, "label")</code> ) is used. If attribute label is <code>NULL</code> , the variable name will be used.
<code>stat_fns</code>	Formula or list of formulas specifying the function to be used to compute the statistics (see below for details and examples). You can also use dedicated helpers such as <code>continuous_summary()</code> , <code>ratio_summary()</code> or <code>proportion_summary()</code> .
<code>statistic</code>	List of formulas specifying the <code>glue::glue()</code> pattern to display the statistics for each variable. The statistics should be returned by the functions specified in <code>stat_fns</code> (see below for details and examples).
<code>digits</code>	List of formulas specifying the number of decimal places to round summary statistics. If not specified, <code>tbl_summary</code> guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is <code>"{mean} ({sd})"</code> and you want the mean rounded to 1 decimal

	place, and the SD to 2 use <code>digits = list(age ~ c(1,2))</code> . User may also pass a styling function: <code>digits = age ~ style_sigfig</code>
<code>type</code>	List of formulas specifying variable types. Accepted values are <code>c("continuous", "continuous2", "e.g. type = list(age ~ "continuous", female ~ "dichotomous")</code> . If type not specified for a variable, the function will default to an appropriate summary type. See below for details.
<code>value</code>	List of formulas specifying the value to display for dichotomous variables. See below for details.
<code>missing</code>	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
<code>missing_text</code>	String to display for count of missing observations. Default is "Unknown".
<code>include</code>	variables to include in the summary table. Default is <code>everything()</code>
<code>overall_row</code>	Logical indicator to display an overall row. Default is FALSE. Use <code>add_overall()</code> to add an overall column.
<code>overall_row_last</code>	Logical indicator to display overall row last in table. Default is FALSE, which will display overall row first.
<code>overall_row_label</code>	String indicating the overall row label. Default is "Overall".

### Value

A `tbl_custom_summary` and `tbl_summary` object

### Similarities with `tbl_summary()`

Please refer to the help file of `tbl_summary()` regarding the use of select helpers, and arguments `include`, `by`, `type`, `value`, `digits`, `missing` and `missing_text`.

### `stat_fns` argument

The `stat_fns` argument specify the custom function(s) to be used for computing the summary statistics. For example, `stat_fns = everything() ~ foo`.

Each function may take the following arguments: `foo(data, full_data, variable, by, type, ...)`

- `data=` is the input data frame passed to `tbl_custom_summary()`, subset according to the level of `by` or `variable` if any, excluding NA values of the current variable
- `full_data=` is the full input data frame passed to `tbl_custom_summary()`
- `variable=` is a string indicating the variable to perform the calculation on
- `by=` is a string indicating the by variable from `tbl_custom_summary=`, if present
- `type=` is a string indicating the type of variable (continuous, categorical, ...)
- `stat_display=` a string indicating the statistic to display (for the `statistic` argument, for that variable)

The user-defined does not need to utilize each of these inputs. It's encouraged the user-defined function accept `...` as each of the arguments *will* be passed to the function, even if not all inputs are utilized by the user's function, e.g. `foo(data, ...)` (see examples).

The user-defined function should return a one row `dplyr::tibble()` with one column per summary statistics (see examples).

**statistic argument**

The statistic argument specifies the statistics presented in the table. The input is a list of formulas that specify the statistics to report. For example, `statistic = list(age ~ "{mean} ({sd})")`. A statistic name that appears between curly brackets will be replaced with the numeric statistic (see [glue::glue\(\)](#)). All the statistics indicated in the statistic argument should be returned by the functions defined in the `stat_fns` argument.

When the summary type is "continuous2", pass a vector of statistics. Each element of the vector will result in a separate row in the summary table.

For both categorical and continuous variables, statistics on the number of missing and non-missing observations and their proportions are also available to display.

- `{N_obs}` total number of observations
- `{N_miss}` number of missing observations
- `{N_nonmiss}` number of non-missing observations
- `{p_miss}` percentage of observations missing
- `{p_nonmiss}` percentage of observations not missing

Note that for categorical variables, `{N_obs}`, `{N_miss}` and `{N_nonmiss}` refer to the total number, number missing and number non missing observations in the denominator, not at each level of the categorical variable.

It is recommended to use [modify\\_footnote\(\)](#) to properly describe the displayed statistics (see examples).

**Caution**

The returned table is compatible with all `gtsummary` features applicable to a `tbl_summary` object, like [add\\_overall\(\)](#), [modify\\_footnote\(\)](#) or [bold\\_labels\(\)](#).

However, some of them could be inappropriate in such case. In particular, [add\\_p\(\)](#) do not take into account the type of displayed statistics and always return the p-value of a comparison test of the current variable according to the by groups, which may be incorrect if the displayed statistics refer to a third variable.

**Example Output****Author(s)**

Joseph Larmarange

**See Also**

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_custom_summary` tools: [add\\_overall\(\)](#), [continuous\\_summary\(\)](#), [proportion\\_summary\(\)](#), [ratio\\_summary\(\)](#)

**Examples**

```

# Example 1 -----
my_stats <- function(data, ...) {
  marker_sum = sum(data$marker, na.rm = TRUE)
  mean_age = mean(data$age, na.rm = TRUE)
  dplyr::tibble(
    marker_sum = marker_sum,
    mean_age = mean_age
  )
}

my_stats(trial)

tbl_custom_summary_ex1 <-
  trial %>%
  tbl_custom_summary(
    include = c("stage", "grade"),
    by = "trt",
    stat_fns = everything() ~ my_stats,
    statistic = everything() ~ "A: {mean_age} - S: {marker_sum}",
    digits = everything() ~ c(1, 0),
    overall_row = TRUE,
    overall_row_label = "All stages & grades"
  ) %>%
  add_overall(last = TRUE) %>%
  modify_footnote(
    update = all_stat_cols() ~ "A: mean age - S: sum of marker"
  ) %>%
  bold_labels()

# Example 2 -----
# Use `data[[variable]]` to access the current variable
mean_ci <- function(data, variable, ...) {
  test <- t.test(data[[variable]])
  dplyr::tibble(
    mean = test$estimate,
    conf.low = test$conf.int[1],
    conf.high = test$conf.int[2]
  )
}

tbl_custom_summary_ex2 <-
  trial %>%
  tbl_custom_summary(
    include = c("marker", "ttdeath"),
    by = "trt",
    stat_fns = ~ mean_ci,
    statistic = ~ "{mean} [{conf.low}; {conf.high}]"
  ) %>%
  add_overall(last = TRUE) %>%
  modify_footnote(
    update = all_stat_cols() ~ "mean [95% CI]"
  )

# Example 3 -----
# Use `full_data` to access the full datasets

```



```

# Returned statistic can also be a character
diff_to_great_mean <- function(data, full_data, ...) {
  mean <- mean(data$marker, na.rm = TRUE)
  great_mean <- mean(full_data$marker, na.rm = TRUE)
  diff <- mean - great_mean
  dplyr::tibble(
    mean = mean,
    great_mean = great_mean,
    diff = diff,
    level = ifelse(diff > 0, "high", "low")
  )
}

tbl_custom_summary_ex3 <-
  trial %>%
  tbl_custom_summary(
    include = c("grade", "stage"),
    by = "trt",
    stat_fns = ~ diff_to_great_mean,
    statistic = ~ "{mean} ({level}, diff: {diff})",
    overall_row = TRUE
  ) %>%
  bold_labels()

```

tbl\_merge

*Merge two or more gtsummary objects***Description**

Merges two or more `tbl_regression`, `tbl_uvregression`, `tbl_stack`, `tbl_summary`, or `tbl_svysummary` objects and adds appropriate spanning headers.

**Usage**

```
tbl_merge(tbls, tab_spanner = NULL)
```

**Arguments**

<code>tbls</code>	List of <code>gtsummary</code> objects to merge
<code>tab_spanner</code>	Character vector specifying the spanning headers. Must be the same length as <code>tbls</code> . The strings are interpreted with <code>gt::md</code> . Must be same length as <code>tbls</code> argument. Default is <code>NULL</code> , and places a default spanning header. If <code>FALSE</code> , no header will be placed.

**Value**

A `tbl_merge` object

**Example Output**

**Author(s)**

Daniel D. Sjoberg

**See Also**[tbl\\_stack](#)

Other `tbl_regression` tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [combine\\_terms\(\)](#), [inline\\_text.tbl\\_regression\(\)](#), [modify](#), [tbl\\_regression\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

Other `tbl_uvregression` tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_uvregression\(\)](#), [modify](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_uvregression\(\)](#)

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_summary\(\)](#)

Other `tbl_survfit` tools: [add\\_n.tbl\\_survfit\(\)](#), [add\\_nevent.tbl\\_survfit\(\)](#), [add\\_p.tbl\\_survfit\(\)](#), [modify](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_survfit\(\)](#)

Other `tbl_svysummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svysummary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#), [tbl\\_svysummary\(\)](#)

**Examples**

```
# Example 1 -----
# Side-by-side Regression Models
library(survival)
t1 <-
  glm(response ~ trt + grade + age, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE)
t2 <-
  coxph(Surv(ttdeath, death) ~ trt + grade + age, trial) %>%
  tbl_regression(exponentiate = TRUE)
tbl_merge_ex1 <-
  tbl_merge(
    tbls = list(t1, t2),
    tab_spanner = c("**Tumor Response**", "**Time to Death**")
  )

# Example 2 -----
# Descriptive statistics alongside univariate regression, with no spanning header
t3 <-
  trial[c("age", "grade", "response")] %>%
  tbl_summary(missing = "no") %>%
  add_n() %>%
  modify_header(stat_0 ~ "**Summary Statistics**")
t4 <-
  tbl_uvregression(
    trial[c("ttdeath", "death", "age", "grade", "response")],
    method = coxph,
    y = Surv(ttdeath, death),
    exponentiate = TRUE,
    hide_n = TRUE
  )
```

```
tbl_merge_ex2 <-
  tbl_merge(tbls = list(t3, t4)) %>%
  modify_spanning_header(everything() ~ NA_character_)
```

---

tbl_regression	<i>Display regression model results in table</i>
----------------	--

---

## Description

This function takes a regression model object and returns a formatted table that is publication-ready. The function is highly customizable allowing the user to obtain a bespoke summary table of the regression model results. Review the [tbl\\_regression vignette](#) for detailed examples.

## Usage

```
tbl_regression(x, ...)

## Default S3 method:
tbl_regression(
  x,
  label = NULL,
  exponentiate = FALSE,
  include = everything(),
  show_single_row = NULL,
  conf.level = NULL,
  intercept = FALSE,
  estimate_fun = NULL,
  pvalue_fun = NULL,
  tidy_fun = NULL,
  add_estimate_to_reference_rows = FALSE,
  conf.int = NULL,
  ...
)
```

## Arguments

x	Regression model object
...	Not used
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code>
exponentiate	Logical indicating whether to exponentiate the coefficient estimates. Default is FALSE.
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
show_single_row	By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here—quoted and unquoted variable name accepted.

<code>conf.level</code>	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
<code>intercept</code>	Logical argument indicating whether to include the intercept in the output. Default is FALSE
<code>estimate_fun</code>	Function to round and format coefficient estimates. Default is <code>style_sigfig</code> when the coefficients are not transformed, and <code>style_ratio</code> when the coefficients have been exponentiated.
<code>pvalue_fun</code>	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
<code>tidy_fun</code>	Option to specify a particular tidier function for the model. Default is to use <code>broom::tidy</code> , but if an error occurs then tidying of the model is attempted with <code>parameters::model_parameters()</code> , if installed.
<code>add_estimate_to_reference_rows</code>	add a reference value. Default is FALSE
<code>conf.int</code>	Logical indicating whether or not to include a confidence interval in the output. Defaults to TRUE.

## Value

A `tbl_regression` object

## Methods

The default method for `tbl_regression()` model summary uses `broom::tidy(x)` to perform the initial tidying of the model object. There are, however, a few models that use [modifications](#).

- "parsnip/workflows": If the model was prepared using `parsnip/workflows`, the original model fit is extracted and the original `x=` argument is replaced with the model fit. This will typically go unnoticed; however, if you've provided a custom tidier in `tidy_fun=` the tidier will be applied to the model fit object and not the `parsnip/workflows` object.
- "survreg": The scale parameter is removed, `broom::tidy(x) %>% dplyr::filter(term != "Log(scale)")`
- "multinom": This multinomial outcome is complex, with one line per covariate per outcome (less the reference group)
- "gam": Uses the internal tidier `tidy_gam()` to print both parametric and smooth terms.
- "lmerMod", "glmerMod", "glmmTMB", "glmmadmb", "stanreg", "brmsfit": These mixed effects models use `broom.mixed::tidy(x, effects = "fixed")`. Specify `tidy_fun = broom.mixed::tidy` to print the random components.

This list is not exhaustive, and care should be taken for each number reported.

## Example Output

## Author(s)

Daniel D. Sjoberg

**See Also**

See `tbl_regression` [vignette](#) for detailed examples

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`

**Examples**

```
# Example 1 -----
library(survival)
tbl_regression_ex1 <-
  coxph(Surv(ttdeath, death) ~ age + marker, trial) %>%
  tbl_regression(exponentiate = TRUE)

# Example 2 -----
tbl_regression_ex2 <-
  glm(response ~ age + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE)

# Example 3 -----
suppressMessages(library(lme4))
tbl_regression_ex3 <-
  glmer(am ~ hp + (1 | gear), mtcars, family = binomial) %>%
  tbl_regression(exponentiate = TRUE)
```

tbl\_split

*Split gtsummary table***Description**

**[Experimental]** The `tbl_split` function splits a single `gtsummary` table into multiple tables. Updates to the print method are expected.

**Usage**

```
tbl_split(x, ...)

## S3 method for class 'gtsummary'
tbl_split(x, variables, ...)

## S3 method for class 'tbl_split'
print(x, ...)
```

**Arguments**

<code>x</code>	<code>gtsummary</code> table
<code>...</code>	not used
<code>variables</code>	variables at which to split the <code>gtsummary</code> table rows (tables will be separated after each of these variables)

**Value**

tbl\_split object

**See Also**

Other tbl\_regression tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_stack()`, `tbl_strata()`

Other tbl\_uvregression tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_strata()`, `tbl_uvregression()`

Other tbl\_summary tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_stack()`, `tbl_strata()`, `tbl_summary()`

Other tbl\_survfit tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_strata()`, `tbl_survfit()`

Other tbl\_svysummary tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svysummary()`, `add_q()`, `add_stat_label()`, `modify`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_stack()`, `tbl_strata()`, `tbl_svysummary()`

**Examples**

```
tbl <-
  tbl_summary(trial) %>%
  tbl_split(variables = c(marker, grade))
```

---

tbl\_stack

*Stacks two or more gtsummary objects*


---

**Description**

Assists in patching together more complex tables. `tbl_stack()` appends two or more `tbl_regression`, `tbl_summary`, `tbl_svysummary`, or `tbl_merge` objects. Column attributes, including number formatting and column footnotes, are retained from the first passed `gtsummary` object.

**Usage**

```
tbl_stack(tbls, group_header = NULL, quiet = NULL)
```

**Arguments**

tbls	List of <code>gtsummary</code> objects
group_header	Character vector with table headers where length matches the length of <code>tbls</code>
quiet	Logical indicating whether to print messages in console. Default is FALSE

**Value**

A `tbl_stack` object

## Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

#### [tbl\\_merge](#)

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_strata()`, `tbl_summary()`

Other `tbl_svsummary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svsummary()`, `add_q()`, `add_stat_label()`, `modify`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_split()`, `tbl_strata()`, `tbl_svsummary()`

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_strata()`

Other `tbl_uvregression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_strata()`, `tbl_uvregression()`

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_strata()`, `tbl_survfit()`

### Examples

```
# Example 1 -----
# stacking two tbl_regression objects
t1 <-
  glm(response ~ trt, trial, family = binomial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)")
  )

t2 <-
  glm(response ~ trt + grade + stage + marker, trial, family = binomial) %>%
  tbl_regression(
    include = "trt",
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (adjusted)")
  )

tbl_stack_ex1 <- tbl_stack(list(t1, t2))

# Example 2 -----
# stacking two tbl_merge objects
library(survival)
t3 <-
  coxph(Surv(ttdeath, death) ~ trt, trial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)")
  )
```

```

)

t4 <-
  coxph(Surv(ttdeath, death) ~ trt + grade + stage + marker, trial) %>%
  tbl_regression(
    include = "trt",
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (adjusted)")
  )

# first merging, then stacking
row1 <- tbl_merge(list(t1, t3), tab_spanner = c("Tumor Response", "Death"))
row2 <- tbl_merge(list(t2, t4))
tbl_stack_ex2 <-
  tbl_stack(list(row1, row2), group_header = c("Unadjusted Analysis", "Adjusted Analysis"))

```

tbl\_strata

*Stratified gtsummary tables*

## Description

**[Maturing]** Build a stratified gtsummary table. Any gtsummary table that accepts a data frame as its first argument can be stratified.

- In `tbl_strata()`, the stratified or subset data frame is passed to the function in `.tbl_fun=`, e.g. `purrr::map(data, .tbl_fun)`.
- In `tbl_strata2()`, both the stratified data frame and the strata level are passed to `.tbl_fun=`, e.g. `purrr::map2(data, strata, .tbl_fun)`

## Usage

```

tbl_strata(
  data,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",
  .combine_with = c("tbl_merge", "tbl_stack"),
  .combine_args = NULL,
  .header = ifelse(.combine_with == "tbl_merge", "**{strata}**", "{strata}"),
  .stack_group_header = NULL,
  .quiet = NULL
)

tbl_strata2(
  data,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",

```



```

.combine_with = c("tbl_merge", "tbl_stack"),
.combine_args = NULL,
.header = ifelse(.combine_with == "tbl_merge", "**{strata}**", "{strata}"),
.stack_group_header = NULL,
.quiet = NULL
)

```

## Arguments

<code>data</code>	a data frame or survey object
<code>strata</code>	character vector or tidy-selector of columns in data to stratify results by
<code>.tbl_fun</code>	A function or formula. If a <i>function</i> , it is used as is. If a formula, e.g. <code>~ .x %&gt;% tbl_summary() %&gt;% add_p()</code> , it is converted to a function. The stratified data frame is passed to this function.
<code>...</code>	Additional arguments passed on to the <code>.tbl_fun</code> function.
<code>.sep</code>	when more than one stratifying variable is passed, this string is used to separate the levels in the spanning header. Default is <code>" , "</code>
<code>.combine_with</code>	One of <code>c("tbl_merge", "tbl_stack")</code> . Names the function used to combine the stratified tables.
<code>.combine_args</code>	named list of arguments that are passed to function specified in <code>.combine_with=</code>
<code>.header</code>	String indicating the headers that will be placed. Default is <code>"**{strata}**"</code> when <code>.combine_with = "tbl_merge"</code> and <code>"{strata}"</code> when <code>.combine_with = "tbl_stack"</code> . Items placed in curly brackets will be evaluated according to <code>glue::glue()</code> syntax. <ul style="list-style-type: none"> <li>• <code>strata</code> stratum levels</li> <li>• <code>n</code> N within stratum</li> <li>• <code>N</code> Overall N The evaluated value of <code>.header=</code> is also available within <code>tbl_strata2(.tbl_fun=)</code></li> </ul>
<code>.stack_group_header</code>	DEPRECATED.
<code>.quiet</code>	Logical indicating whether to print messages in console. Default is <code>FALSE</code>

## Tips

- `tbl_summary()`
  - The number of digits continuous variables are rounded to is determined separately within each stratum of the data frame. Set the `digits=` argument to ensure continuous variables are rounded to the same number of decimal places.
  - If some levels of a categorical variable are unobserved within a stratum, convert the variable to a factor to ensure all levels appear in each stratum's summary table.

## Example Output

## Author(s)

Daniel D. Sjoberg

**See Also**

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_stack()`

Other `tbl_uvregression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_uvregression()`

Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `separate_p_footnotes()`, `tbl_custom_summary()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_summary()`

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_survfit()`

Other `tbl_svysummary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svysummary()`, `add_q()`, `add_stat_label()`, `modify`, `separate_p_footnotes()`, `tbl_merge()`, `tbl_split()`, `tbl_stack()`, `tbl_svysummary()`

**Examples**

```
# Example 1 -----
tbl_strata_ex1 <-
  trial %>%
  select(age, grade, stage, trt) %>%
  mutate(grade = paste("Grade", grade)) %>%
  tbl_strata(
    strata = grade,
    .tbl_fun =
      ~ .x %>%
        tbl_summary(by = trt, missing = "no") %>%
        add_n(),
    .header = "**{strata}**", N = {n}"
  )

# Example 2 -----
tbl_strata_ex2 <-
  trial %>%
  select(grade, response) %>%
  mutate(grade = paste("Grade", grade)) %>%
  tbl_strata2(
    strata = grade,
    .tbl_fun =
      ~.x %>%
        tbl_summary(
          label = list(response = .y),
          missing = "no",
          statistic = response ~ "{p}%"
        ) %>%
        add_ci(pattern = "{stat} ({ci})") %>%
        modify_header(stat_0 = "**Rate (95% CI)**") %>%
        modify_footnote(stat_0 = NA),
    .combine_with = "tbl_stack",
    .combine_args = list(group_header = NULL),
    .quiet = TRUE
  ) %>%
  modify_caption("**Response Rate by Grade**")
```

---

tbl\_summary *Create a table of summary statistics*


---

### Description

The `tbl_summary` function calculates descriptive statistics for continuous, categorical, and dichotomous variables. Review the [tbl\\_summary vignette](#) for detailed examples.

### Usage

```
tbl_summary(
  data,
  by = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  sort = NULL,
  percent = NULL,
  include = everything()
)
```

### Arguments

<code>data</code>	A data frame
<code>by</code>	A column name (quoted or unquoted) in <code>data</code> . Summary statistics will be calculated separately for each level of the <code>by</code> variable (e.g. <code>by = trt</code> ). If <code>NULL</code> , summary statistics are calculated using all observations. To stratify a table by two or more variables, use <code>tbl_strata()</code>
<code>label</code>	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute ( <code>attr(data\$age, "label")</code> ) is used. If attribute label is <code>NULL</code> , the variable name will be used.
<code>statistic</code>	List of formulas specifying types of summary statistics to display for each variable. The default is <code>list(all_continuous() ~ "{median} ({p25},{p75})", all_categorical() ~ "{n} ({p}%))"</code> . See below for details.
<code>digits</code>	List of formulas specifying the number of decimal places to round summary statistics. If not specified, <code>tbl_summary</code> guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is <code>"{mean} ({sd})"</code> and you want the mean rounded to 1 decimal place, and the SD to 2 use <code>digits = list(age ~ c(1,2))</code> . User may also pass a styling function: <code>digits = age ~ style_sigfig</code>
<code>type</code>	List of formulas specifying variable types. Accepted values are <code>c("continuous", "continuous2", "dichotomous")</code> , e.g. <code>type = list(age ~ "continuous", female ~ "dichotomous")</code> . If type not specified for a variable, the function will default to an appropriate summary type. See below for details.

value	List of formulas specifying the value to display for dichotomous variables. See below for details.
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
sort	List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. <code>sort = list(everything() ~ "frequency")</code>
percent	Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".
include	variables to include in the summary table. Default is <code>everything()</code>

### Value

A `tbl_summary` object

### select helpers

**Select helpers** from the `\tidyselect\` package and `\gtsummary\` package are available to modify default behavior for groups of variables. For example, by default continuous variables are reported with the median and IQR. To change all continuous variables to mean and standard deviation use `statistic = list(all_continuous() ~ "{mean} ({sd})")`.

All columns with class logical are displayed as dichotomous variables showing the proportion of events that are TRUE on a single row. To show both rows (i.e. a row for TRUE and a row for FALSE) use `type = list(where(is.logical) ~ "categorical")`.

The select helpers are available for use in any argument that accepts a list of formulas (e.g. `statistic`, `type`, `digits`, `value`, `sort`, etc.)

Read more on the [syntax](#) used through the package.

### type argument

The `tbl_summary()` function has four summary types:

- "continuous" summaries are shown on a *single row*. Most numeric variables default to summary type continuous.
- "continuous2" summaries are shown on *2 or more rows*
- "categorical" *multi-line* summaries of nominal data. Character variables, factor variables, and numeric variables with fewer than 10 unique levels default to type categorical. To change a numeric variable to continuous that defaulted to categorical, use `type = list(varname ~ "continuous")`
- "dichotomous" categorical variables that are displayed on a *single row*, rather than one row per level of the variable. Variables coded as TRUE/FALSE, 0/1, or yes/no are assumed to be dichotomous, and the TRUE, 1, and yes rows are displayed. Otherwise, the value to display must be specified in the value argument, e.g. `value = list(varname ~ "level to show")`

**statistic argument**

The statistic argument specifies the statistics presented in the table. The input is a list of formulas that specify the statistics to report. For example, `statistic = list(age ~ "{mean} ({sd})")` would report the mean and standard deviation for age; `statistic = list(all_continuous() ~ "{mean} ({sd})")` would report the mean and standard deviation for all continuous variables. A statistic name that appears between curly brackets will be replaced with the numeric statistic (see [glue::glue](#)).

For categorical variables the following statistics are available to display.

- {n} frequency
- {N} denominator, or cohort size
- {p} formatted percentage

For continuous variables the following statistics are available to display.

- {median} median
- {mean} mean
- {sd} standard deviation
- {var} variance
- {min} minimum
- {max} maximum
- {sum} sum
- {p##} any integer percentile, where ## is an integer from 0 to 100
- {foo} any function of the form `foo(x)` is accepted where `x` is a numeric vector

When the summary type is "continuous2", pass a vector of statistics. Each element of the vector will result in a separate row in the summary table.

For both categorical and continuous variables, statistics on the number of missing and non-missing observations and their proportions are available to display.

- {N\_obs} total number of observations
- {N\_miss} number of missing observations
- {N\_nonmiss} number of non-missing observations
- {p\_miss} percentage of observations missing
- {p\_nonmiss} percentage of observations not missing

Note that for categorical variables, {N\_obs}, {N\_miss} and {N\_nonmiss} refer to the total number, number missing and number non missing observations in the denominator, not at each level of the categorical variable.

**Example Output****Author(s)**

Daniel D. Sjoberg

**See Also**

See [tbl\\_summary vignette](#) for detailed tutorial

See [table gallery](#) for additional examples

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other `tbl_summary` tools: [add\\_ci\(\)](#), [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_summary\(\)](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_custom\\_summary\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

**Examples**

```
# Example 1 -----
tbl_summary_ex1 <-
  trial %>%
  select(age, grade, response) %>%
  tbl_summary()

# Example 2 -----
tbl_summary_ex2 <-
  trial %>%
  select(age, grade, response, trt) %>%
  tbl_summary(
    by = trt,
    label = list(age ~ "Patient Age"),
    statistic = list(all_continuous() ~ "{mean} ({sd})"),
    digits = list(age ~ c(0, 1))
  )

# Example 3 -----
# for convenience, you can also pass named lists to any arguments
# that accept formulas (e.g label, digits, etc.)
tbl_summary_ex3 <-
  trial %>%
  select(age, trt) %>%
  tbl_summary(
    by = trt,
    label = list(age = "Patient Age")
  )

# Example 4 -----
# multi-line summaries of continuous data with type 'continuous2'
tbl_summary_ex4 <-
  trial %>%
  select(age, marker) %>%
  tbl_summary(
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c("{median} ({p25}, {p75})", "{min}, {max}"),
    missing = "no"
  )
```

---

tbl_survfit	<i>Creates table of survival probabilities</i>
-------------	--

---

### Description

**[Maturing]** Function takes a survfit object as an argument, and provides a formatted summary table of the results

### Usage

```
tbl_survfit(x, ...)

## S3 method for class 'list'
tbl_survfit(
  x,
  times = NULL,
  probs = NULL,
  statistic = NULL,
  label = NULL,
  label_header = NULL,
  estimate_fun = NULL,
  missing = NULL,
  conf.level = 0.95,
  reverse = FALSE,
  quiet = NULL,
  ...
)

## S3 method for class 'survfit'
tbl_survfit(x, ...)

## S3 method for class 'data.frame'
tbl_survfit(x, y, include = everything(), ...)
```

### Arguments

x	a survfit object, list of survfit objects, or a data frame. If a data frame is passed, a list of survfit objects is constructed using each variable as a stratifying variable.
...	For <code>tbl_survfit.data.frame()</code> and <code>tbl_survfit.survfit()</code> the arguments are passed to <code>tbl_survfit.list()</code> . They are not used when <code>tbl_survfit.list()</code> is called directly.
times	numeric vector of times for which to return survival probabilities.
probs	numeric vector of probabilities with values in (0,1) specifying the survival quantiles to return
statistic	string defining the statistics to present in the table. Default is "{estimate} ({conf.low},{conf.high})"
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age, yrs", stage ~ "Path T Stage")</code> , or a string for a single variable table.

label_header	string specifying column labels above statistics. Default is "{prob} Percentile" for survival percentiles, and "Time {time}" for n-year survival estimates
estimate_fun	function to format the Kaplan-Meier estimates. Default is <code>style_percent()</code> for survival probabilities and <code>style_sigfig</code> for survival times
missing	text to fill when estimate is not estimable. Default is "--"
conf.level	Confidence level for confidence intervals. Default is 0.95
reverse	Flip the probability reported, i.e. 1 - estimate. Default is FALSE. Does not apply to survival quantile requests
quiet	Logical indicating whether to print messages in console. Default is FALSE
y	outcome call, e.g. <code>y = Surv(ttdeath, death)</code>
include	Variable to include as stratifying variables.

### Example Output

#### Author(s)

Daniel D. Sjoberg

#### See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_survfit` tools: [add\\_n.tbl\\_survfit\(\)](#), [add\\_nevent.tbl\\_survfit\(\)](#), [add\\_p.tbl\\_survfit\(\)](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

#### Examples

```
library(survival)

# Example 1 -----
# Pass single survfit() object
tbl_survfit_ex1 <- tbl_survfit(
  survfit(Surv(ttdeath, death) ~ trt, trial),
  times = c(12, 24),
  label_header = "**{time} Month**"
)

# Example 2 -----
# Pass a data frame
tbl_survfit_ex2 <- tbl_survfit(
  trial,
  y = Surv(ttdeath, death),
  include = c(trt, grade),
  probs = 0.5,
  label_header = "**Median Survival**"
)

# Example 3 -----
# Pass a list of survfit() objects
tbl_survfit_ex3 <-
  list(
    survfit(Surv(ttdeath, death) ~ 1, trial),
```



```

    survfit(Surv(ttdeath, death) ~ trt, trial)
  ) %>%
  tbl_survfit(times = c(12, 24))

# Example 4 Competing Events Example -----
# adding a competing event for death (cancer vs other causes)
set.seed(1123)
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)
trial2 <- trial %>%
  mutate(
    death_cr = case_when(
      death == 0 ~ "censor",
      runif(n()) < 0.5 ~ "death from cancer",
      TRUE ~ "death other causes"
    ) %>% factor()
  )

survfit_cr_ex4 <-
  survfit(Surv(ttdeath, death_cr) ~ grade, data = trial2) %>%
  tbl_survfit(times = c(12, 24), label = "Tumor Grade")

```

tbl\_svysummary

*Create a table of summary statistics from a survey object*

## Description

The `tbl_svysummary` function calculates descriptive statistics for continuous, categorical, and dichotomous variables taking into account survey weights and design. It is similar to `tbl_summary()`.

## Usage

```
tbl_svysummary(
  data,
  by = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  sort = NULL,
  percent = NULL,
  include = everything()
)
```

## Arguments

<code>data</code>	A survey object created with <code>survey::svydesign()</code>
<code>by</code>	A column name (quoted or unquoted) in <code>data</code> . Summary statistics will be calculated separately for each level of the <code>by</code> variable (e.g. <code>by = trt</code> ). If <code>NULL</code> , summary statistics are calculated using all observations. To stratify a table by two or more variables, use <code>tbl_strata()</code>

label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute ( <code>attr(data\$age, "label")</code> ) is used. If attribute label is NULL, the variable name will be used.
statistic	List of formulas specifying types of summary statistics to display for each variable. The default is <code>list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)"</code> . See below for details.
digits	List of formulas specifying the number of decimal places to round summary statistics. If not specified, <code>tbl_summary</code> guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is <code>"{mean} ({sd})"</code> and you want the mean rounded to 1 decimal place, and the SD to 2 use <code>digits = list(age ~ c(1, 2))</code> . User may also pass a styling function: <code>digits = age ~ style_sigfig</code>
type	List of formulas specifying variable types. Accepted values are <code>c("continuous", "continuous2", "dichotomous")</code> , e.g. <code>type = list(age ~ "continuous", female ~ "dichotomous")</code> . If type not specified for a variable, the function will default to an appropriate summary type. See below for details.
value	List of formulas specifying the value to display for dichotomous variables. See below for details.
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
sort	List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. <code>sort = list(everything() ~ "frequency")</code>
percent	Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".
include	variables to include in the summary table. Default is <code>everything()</code>

## Value

A `tbl_svsummary` object

## statistic argument

The `statistic` argument specifies the statistics presented in the table. The input is a list of formulas that specify the statistics to report. For example, `statistic = list(age ~ "{mean} ({sd})"`) would report the mean and standard deviation for age; `statistic = list(all_continuous() ~ "{mean} ({sd})"`) would report the mean and standard deviation for all continuous variables. A statistic name that appears between curly brackets will be replaced with the numeric statistic (see [glue::glue](#)).

For categorical variables the following statistics are available to display.

- `{n}` frequency
- `{N}` denominator, or cohort size
- `{p}` formatted percentage
- `{n_unweighted}` unweighted frequency

- {N\_unweighted} unweighted denominator
- {p\_unweighted} unweighted formatted percentage

For continuous variables the following statistics are available to display.

- {median} median
- {mean} mean
- {sd} standard deviation
- {var} variance
- {min} minimum
- {max} maximum
- {p##} any integer percentile, where ## is an integer from 0 to 100
- {sum} sum

Unlike `tbl_summary()`, it is not possible to pass a custom function.

For both categorical and continuous variables, statistics on the number of missing and non-missing observations and their proportions are available to display.

- {N\_obs} total number of observations
- {N\_miss} number of missing observations
- {N\_nonmiss} number of non-missing observations
- {p\_miss} percentage of observations missing
- {p\_nonmiss} percentage of observations not missing
- {N\_obs\_unweighted} unweighted total number of observations
- {N\_miss\_unweighted} unweighted number of missing observations
- {N\_nonmiss\_unweighted} unweighted number of non-missing observations
- {p\_miss\_unweighted} unweighted percentage of observations missing
- {p\_nonmiss\_unweighted} unweighted percentage of observations not missing

Note that for categorical variables, {N\_obs}, {N\_miss} and {N\_nonmiss} refer to the total number, number missing and number non missing observations in the denominator, not at each level of the categorical variable.

## Example Output

### type argument

The `tbl_summary()` function has four summary types:

- "continuous" summaries are shown on a *single row*. Most numeric variables default to summary type continuous.
- "continuous2" summaries are shown on *2 or more rows*
- "categorical" *multi-line* summaries of nominal data. Character variables, factor variables, and numeric variables with fewer than 10 unique levels default to type categorical. To change a numeric variable to continuous that defaulted to categorical, use `type = list(varname ~ "continuous")`
- "dichotomous" categorical variables that are displayed on a *single row*, rather than one row per level of the variable. Variables coded as TRUE/FALSE, 0/1, or yes/no are assumed to be dichotomous, and the TRUE, 1, and yes rows are displayed. Otherwise, the value to display must be specified in the value argument, e.g. `value = list(varname ~ "level to show")`

## select helpers

**Select helpers** from the `\tidyselect\` package and `\gtsummary\` package are available to modify default behavior for groups of variables. For example, by default continuous variables are reported with the median and IQR. To change all continuous variables to mean and standard deviation use `statistic = list(all_continuous() ~ "{mean} ({sd})")`.

All columns with class logical are displayed as dichotomous variables showing the proportion of events that are TRUE on a single row. To show both rows (i.e. a row for TRUE and a row for FALSE) use `type = list(where(is.logical) ~ "categorical")`.

The select helpers are available for use in any argument that accepts a list of formulas (e.g. `statistic`, `type`, `digits`, `value`, `sort`, etc.)

Read more on the [syntax](#) used through the package.

## Author(s)

Joseph Larmarange

## See Also

Review [list](#), [formula](#), and [selector syntax](#) used throughout `gtsummary`

Other `tbl_svysummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svysummary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [modify](#), [separate\\_p\\_footnotes\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

## Examples

```
# A simple weighted dataset
tbl_svysummary_ex1 <-
  survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) %>%
  tbl_svysummary(by = Survived, percent = "row")

# Example 2 -----
# A dataset with a complex design
data(api, package = "survey")
tbl_svysummary_ex2 <-
  survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc) %>%
  tbl_svysummary(by = "both", include = c(cname, api00, api99, both))
```

---

tbl\_uvregression

*Display univariate regression model results in table*

---

## Description

This function estimates univariate regression models and returns them in a publication-ready table. It can create univariate regression models holding either a covariate or outcome constant.

For models holding outcome constant, the function takes as arguments a data frame, the type of regression model, and the outcome variable `y=`. Each column in the data frame is regressed on the specified outcome. The `tbl_uvregression` function arguments are similar to the [tbl\\_regression](#) arguments. Review the [tbl\\_uvregression vignette](#) for detailed examples.

You may alternatively hold a single covariate constant. For this, pass a data frame, the type of regression model, and a single covariate in the `x=` argument. Each column of the data frame will serve as the outcome in a univariate regression model. Take care using the `x` argument that each of the columns in the data frame are appropriate for the same type of model, e.g. they are all continuous variables appropriate for `lm`, or dichotomous variables appropriate for logistic regression with `glm`.

## Usage

```
tbl_uvregression(
  data,
  method,
  y = NULL,
  x = NULL,
  method.args = NULL,
  exponentiate = FALSE,
  label = NULL,
  include = everything(),
  tidy_fun = NULL,
  hide_n = FALSE,
  show_single_row = NULL,
  conf.level = NULL,
  estimate_fun = NULL,
  pvalue_fun = NULL,
  formula = "{y} ~ {x}",
  add_estimate_to_reference_rows = NULL,
  conf.int = NULL,
  ...
)
```

## Arguments

<code>data</code>	Data frame to be used in univariate regression modeling. Data frame includes the outcome variable(s) and the independent variables. Survey design objects are also accepted.
<code>method</code>	Regression method (e.g. <code>lm</code> , <code>glm</code> , <code>survival::coxph</code> , <code>survey::svyglm</code> , and more).
<code>y</code>	Model outcome (e.g. <code>y = recurrence</code> or <code>y = Surv(time, recur)</code> ). All other column in data will be regressed on <code>y</code> . Specify one and only one of <code>y</code> or <code>x</code>
<code>x</code>	Model covariate (e.g. <code>x = trt</code> ). All other columns in data will serve as the outcome in a regression model with <code>x</code> as a covariate. Output table is best when <code>x</code> is a continuous or dichotomous variable displayed on a single row. Specify one and only one of <code>y</code> or <code>x</code>
<code>method.args</code>	List of additional arguments passed on to the regression function defined by <code>method</code> .
<code>exponentiate</code>	Logical indicating whether to exponentiate the coefficient estimates. Default is <code>FALSE</code> .
<code>label</code>	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code>
<code>include</code>	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .

tidy_fun	Option to specify a particular tidier function for the model. Default is to use <code>broom::tidy</code> , but if an error occurs then tidying of the model is attempted with <code>parameters::model_parameters()</code> , if installed.
hide_n	Hide N column. Default is FALSE
show_single_row	By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here—quoted and unquoted variable name accepted.
conf.level	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
estimate_fun	Function to round and format coefficient estimates. Default is <code>style_sigfig</code> when the coefficients are not transformed, and <code>style_ratio</code> when the coefficients have been exponentiated.
pvalue_fun	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code> ).
formula	String of the model formula. Uses <code>glue::glue</code> syntax. Default is " <code>{y} ~ {x}</code> ", where <code>{y}</code> is the dependent variable, and <code>{x}</code> represents a single covariate. For a random intercept model, the formula may be <code>formula = "{y} ~ {x} + (1   gear)"</code> .
add_estimate_to_reference_rows	add a reference value. Default is FALSE
conf.int	Logical indicating whether or not to include a confidence interval in the output. Defaults to TRUE.
...	Not used

### Value

A `tbl_uvregression` object

### Example Output

### Methods

The default method for `tbl_regression()` model summary uses `broom::tidy(x)` to perform the initial tidying of the model object. There are, however, a few models that use [modifications](#).

- "parsnip/workflows": If the model was prepared using `parsnip/workflows`, the original model fit is extracted and the original `x=` argument is replaced with the model fit. This will typically go unnoticed; however, if you've provided a custom tidier in `tidy_fun=` the tidier will be applied to the model fit object and not the `parsnip/workflows` object.
- "survreg": The scale parameter is removed, `broom::tidy(x) %>% dplyr::filter(term != "Log(scale)")`
- "multinom": This multinomial outcome is complex, with one line per covariate per outcome (less the reference group)

- "gam": Uses the internal tidier tidy\_gam() to print both parametric and smooth terms.
- "lmerMod", "glmerMod", "glmmTMB", "glmmadmb", "stanreg", "brmsfit": These mixed effects models use broom.mixed::tidy(x, effects = "fixed"). Specify tidy\_fun = broom.mixed::tidy to print the random components.

This list is not exhaustive, and care should be taken for each number reported.

### Author(s)

Daniel D. Sjoberg

### See Also

See tbl\_regression [vignette](#) for detailed examples

Review [list](#), [formula](#), and [selector syntax](#) used throughout gtsummary

Other tbl\_uvregression tools: [add\\_global\\_p\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_uvregression](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_split\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_strata\(\)](#)

### Examples

```
# Example 1 -----
tbl_uv_ex1 <-
  tbl_uvregression(
    trial[c("response", "age", "grade")],
    method = glm,
    y = response,
    method.args = list(family = binomial),
    exponentiate = TRUE
  )

# Example 2 -----
# rounding pvalues to 2 decimal places
library(survival)
tbl_uv_ex2 <-
  tbl_uvregression(
    trial[c("ttdeath", "death", "age", "grade", "response")],
    method = coxph,
    y = Surv(ttdeath, death),
    exponentiate = TRUE,
    pvalue_fun = function(x) style_pvalue(x, digits = 2)
  )
```

---

theme\_gtsummary

*Available gtsummary themes*

---

### Description

**[Maturing]** The following themes are available to use within the gtsummary package. Print theme elements with theme\_gtsummary\_journal(set\_theme = FALSE) %>% print(). Review the [themes vignette](#) for details.

**Usage**

```

theme_gtsummary_journal(
  journal = c("jama", "lancet", "nejm", "qjecon"),
  set_theme = TRUE
)

theme_gtsummary_compact(set_theme = TRUE, font_size = NULL)

theme_gtsummary_printer(
  print_engine = c("gt", "kable", "kable_extra", "flextable", "huxtable", "tibble"),
  set_theme = TRUE
)

theme_gtsummary_language(
  language = c("de", "en", "es", "fr", "gu", "hi", "is", "ja", "kr", "mr", "pt", "se",
    "zh-cn", "zh-tw"),
  decimal.mark = NULL,
  big.mark = NULL,
  iqr.sep = NULL,
  ci.sep = NULL,
  set_theme = TRUE
)

theme_gtsummary_continuous2(
  statistic = "{median} ({p25}, {p75})",
  set_theme = TRUE
)

theme_gtsummary_mean_sd(set_theme = TRUE)

theme_gtsummary_eda(set_theme = TRUE)

```

**Arguments**

journal	String indicating the journal theme to follow. One of c("jama", "lancet", "nejm", "qjecon"). Details below.
set_theme	Logical indicating whether to set the theme. Default is TRUE. When FALSE the named list of theme elements is returned invisibly
font_size	Numeric font size for compact theme. Default is 13 for gt tables, and 8 for all other output types
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"
language	String indicating language. Must be one of "de" (German), "en" (English), "es" (Spanish), "fr" (French), "gu" (Gujarati), "hi" (Hindi), "is" (Icelandic), "ja" (Japanese), "kr" (Korean), "mr" (Marathi), "pt" (Portuguese), "se" (Swedish), "zh-c,n" (Chinese Simplified), "zh-tw" (Chinese Traditional)  If a language is missing a translation for a word or phrase, please feel free to reach out on <a href="#">GitHub</a> with the translated text!
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")



big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", except when decimal.mark = " " when the default is a space.
iqr.sep	string indicating separator for the default IQR in tbl_summary(). If decimal.mark= is NULL, iqr.sep= is ", ". The comma separator, however, can look odd when decimal.mark = " ". In this case the argument will default to an en dash
ci.sep	string indicating separator for confidence intervals. If decimal.mark= is NULL, ci.sep= is ", ". The comma separator, however, can look odd when decimal.mark = " ". In this case the argument will default to an en dash
statistic	Default statistic continuous variables

## Themes

- theme\_gtsummary\_journal(journal=)
  - "jama" *The Journal of the American Medical Association*
    - \* Round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
    - \* tbl\_summary() Doesn't show percent symbol; use em-dash to separate IQR; run add\_stat\_label()
    - \* tbl\_regression()/tbl\_uvregression() show coefficient and CI in same column
  - "lancet" *The Lancet*
    - \* Use mid-point as decimal separator; round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
    - \* tbl\_summary() Doesn't show percent symbol; use em-dash to separate IQR
  - "nejm" *The New England Journal of Medicine*
    - \* Round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
    - \* tbl\_summary() Doesn't show percent symbol; use em-dash to separate IQR
  - "qjecon" *The Quarterly Journal of Economics* **Under Development**
    - \* tbl\_summary() all percentages rounded to one decimal place
    - \* tbl\_regression()/tbl\_uvregression() add significance stars with add\_significance\_stars(); hides CI and p-value from output
- theme\_gtsummary\_compact()
  - tables printed with gt, flextable, kableExtra, or huxtable will be compact with smaller font size and reduced cell padding
- theme\_gtsummary\_printer(print\_engine=)
  - Use this theme to permanently change the default printer.
- theme\_gtsummary\_continuous2()
  - Set all continuous variables to summary type "continuous2" by default
- theme\_gtsummary\_mean\_sd()
  - Set default summary statistics to mean and standard deviation in tbl\_summary()
  - Set default continuous tests in add\_p() to t-test and ANOVA
- theme\_gtsummary\_eda()
  - Set all continuous variables to summary type "continuous2" by default
  - In tbl\_summary() show the median, mean, IQR, SD, and Range by default

Use reset\_gtsummary\_theme() to restore the default settings

Review the [themes vignette](#) to create your own themes.

**Example Output****See Also**[Themes vignette](#)[set\\_gtsummary\\_theme\(\)](#), [reset\\_gtsummary\\_theme\(\)](#)**Examples**

```
# Setting JAMA theme for gtsummary
theme_gtsummary_journal("jama")
# Themes can be combined by including more than one
theme_gtsummary_compact()

set_gtsummary_theme_ex1 <-
  trial %>%
  select(age, grade, trt) %>%
  tbl_summary(by = trt) %>%
  as_gt()

# reset gtsummary themes
reset_gtsummary_theme()
```

---

 trial

---

*Results from a simulated study of two chemotherapy agents*


---

**Description**

A dataset containing the baseline characteristics of 200 patients who received Drug A or Drug B. Dataset also contains the outcome of tumor response to the treatment.

**Usage**

```
trial
```

**Format**

A data frame with 200 rows—one row per patient

**trt** Chemotherapy Treatment

**age** Age

**marker** Marker Level (ng/mL)

**stage** T Stage

**grade** Grade

**response** Tumor Response

**death** Patient Died

**ttdeath** Months to Death/Censor

# Index

- \* **Advanced modifiers**
  - modify\_cols\_merge, 60
  - modify\_column\_alignment, 61
  - modify\_column\_hide, 62
  - modify\_fmt\_fun, 62
  - modify\_table\_body, 64
  - modify\_table\_styling, 65
- \* **datasets**
  - trial, 114
- \* **gtsummary output types**
  - as\_flex\_table, 34
  - as\_gt, 35
  - as\_hux\_table, 36
  - as\_kable, 38
  - as\_kable\_extra, 39
  - as\_tibble.gtsummary, 41
- \* **style tools**
  - style\_number, 76
  - style\_percent, 77
  - style\_pvalue, 78
  - style\_ratio, 79
  - style\_sigfig, 80
- \* **tbl\_cross tools**
  - add\_p.tbl\_cross, 19
  - inline\_text.tbl\_cross, 50
  - tbl\_cross, 83
- \* **tbl\_custom\_summary tools**
  - add\_overall, 16
  - continuous\_summary, 46
  - proportion\_summary, 68
  - ratio\_summary, 69
  - tbl\_custom\_summary, 85
- \* **tbl\_regression tools**
  - add\_global\_p, 9
  - add\_q, 26
  - bold\_italicize\_labels\_levels, 42
  - combine\_terms, 44
  - inline\_text.tbl\_regression, 51
  - modify, 57
  - tbl\_merge, 89
  - tbl\_regression, 91
  - tbl\_split, 93
  - tbl\_stack, 94
  - tbl\_strata, 96
- \* **tbl\_summary tools**
  - add\_ci, 4
  - add\_n.tbl\_summary, 11
  - add\_overall, 16
  - add\_p.tbl\_summary, 20
  - add\_q, 26
  - add\_stat\_label, 31
  - bold\_italicize\_labels\_levels, 42
  - inline\_text.tbl\_summary, 52
  - inline\_text.tbl\_survfit, 54
  - modify, 57
  - separate\_p\_footnotes, 73
  - tbl\_custom\_summary, 85
  - tbl\_merge, 89
  - tbl\_split, 93
  - tbl\_stack, 94
  - tbl\_strata, 96
  - tbl\_summary, 99
- \* **tbl\_survfit tools**
  - add\_n.tbl\_survfit, 13
  - add\_nevent.tbl\_survfit, 13
  - add\_p.tbl\_survfit, 22
  - modify, 57
  - tbl\_merge, 89
  - tbl\_split, 93
  - tbl\_stack, 94
  - tbl\_strata, 96
  - tbl\_survfit, 103
- \* **tbl\_svsummary tools**
  - add\_n.tbl\_summary, 11
  - add\_overall, 16
  - add\_p.tbl\_svsummary, 24
  - add\_q, 26
  - add\_stat\_label, 31
  - modify, 57
  - separate\_p\_footnotes, 73
  - tbl\_merge, 89
  - tbl\_split, 93
  - tbl\_stack, 94
  - tbl\_strata, 96
  - tbl\_svsummary, 105
- \* **tbl\_uvregression tools**

- add\_global\_p, 9
- add\_q, 26
- bold\_italicize\_labels\_levels, 42
- inline\_text.tbl\_uvregression, 55
- modify, 57
- tbl\_merge, 89
- tbl\_split, 93
- tbl\_stack, 94
- tbl\_strata, 96
- tbl\_uvregression, 108
- add\_ci, 4, 12, 17, 22, 26, 32, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 102
- add\_difference, 5
- add\_glance, 7
- add\_glance\_source\_note (add\_glance), 7
- add\_glance\_table (add\_glance), 7
- add\_global\_p, 9, 27, 43, 45, 52, 56, 59, 90, 93–95, 98, 111
- add\_n.tbl\_regression (add\_n\_regression), 15
- add\_n.tbl\_summary, 5, 11, 17, 18, 22, 25, 26, 32, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 102, 108
- add\_n.tbl\_survfit, 13, 14, 23, 59, 90, 94, 95, 98, 104
- add\_n.tbl\_svsummary (add\_n.tbl\_summary), 11
- add\_n.tbl\_uvregression (add\_n\_regression), 15
- add\_n\_regression, 15
- add\_nevent.tbl\_regression (add\_nevent\_regression), 14
- add\_nevent.tbl\_survfit, 13, 13, 23, 59, 90, 94, 95, 98, 104
- add\_nevent.tbl\_uvregression (add\_nevent\_regression), 14
- add\_nevent\_regression, 14
- add\_overall, 5, 12, 16, 22, 25, 26, 32, 43, 46, 53, 55, 59, 69, 70, 73, 87, 90, 94, 95, 98, 102, 108
- add\_overall(), 86, 87
- add\_p(), 87
- add\_p.tbl\_continuous, 18
- add\_p.tbl\_cross, 19, 50, 84
- add\_p.tbl\_summary, 5, 12, 17, 20, 26, 32, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 102
- add\_p.tbl\_survfit, 13, 14, 22, 59, 90, 94, 95, 98, 104
- add\_p.tbl\_svsummary, 12, 18, 24, 26, 32, 59, 73, 90, 94, 95, 98, 108
- add\_q, 5, 10, 12, 17, 18, 22, 25, 26, 32, 43, 45, 52, 53, 55, 56, 59, 73, 87, 90, 93–95, 98, 102, 108, 111
- add\_significance\_stars, 27
- add\_stat, 29
- add\_stat\_label, 5, 12, 17, 18, 22, 25, 26, 31, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 102, 108
- add\_vif, 33
- all\_categorical (select\_helpers), 72
- all\_continuous (select\_helpers), 72
- all\_continuous2 (select\_helpers), 72
- all\_contrasts (select\_helpers), 72
- all\_dichotomous (select\_helpers), 72
- all\_interaction (select\_helpers), 72
- all\_intercepts (select\_helpers), 72
- all\_stat\_cols (select\_helpers), 72
- all\_tests (select\_helpers), 72
- as\_flex\_table, 34, 36–38, 40, 42
- as\_gt, 35, 35, 37, 38, 40, 42
- as\_hux\_table, 35, 36, 36, 38, 40, 42
- as\_kable, 35–37, 38, 40, 42
- as\_kable\_extra, 35–38, 39, 42
- as\_tibble.gtsummary, 35–38, 40, 41
- bold\_italicize\_labels\_levels, 5, 10, 12, 17, 22, 26, 27, 32, 42, 45, 52, 53, 55, 56, 59, 73, 87, 90, 93–95, 98, 102, 111
- bold\_labels (bold\_italicize\_labels\_levels), 42
- bold\_labels(), 38, 87
- bold\_levels (bold\_italicize\_labels\_levels), 42
- bold\_p, 44
- combine\_terms, 10, 27, 43, 44, 52, 59, 90, 93–95, 98
- continuous\_summary, 18, 46, 69, 70, 87
- continuous\_summary(), 85
- custom\_tidiers, 47
- dplyr::tibble(), 86
- filter\_p (sort\_filter\_p), 75
- glm, 109
- glue::glue, 12, 49, 51, 53, 56, 101, 106, 110
- glue::glue(), 58, 85, 87
- gt::gt, 36
- gt::html(), 8, 58
- gt::md(), 8, 58
- gtsummary themes, 74

- Hmisc::binconf(), 68
- inline\_text.gtsummary, 49
- inline\_text.tbl\_cross, 20, 50, 84
- inline\_text.tbl\_regression, 10, 27, 43, 45, 51, 59, 90, 93–95, 98
- inline\_text.tbl\_summary, 5, 12, 17, 22, 26, 32, 43, 52, 55, 59, 73, 87, 90, 94, 95, 98, 102
- inline\_text.tbl\_survfit, 5, 12, 17, 22, 26, 32, 43, 53, 54, 59, 73, 87, 90, 94, 95, 98, 102
- inline\_text.tbl\_svysummary  
(inline\_text.tbl\_summary), 52
- inline\_text.tbl\_uvregression, 10, 27, 43, 55, 59, 90, 94, 95, 98, 111
- italicize\_labels  
(bold\_italicize\_labels\_levels), 42
- italicize\_levels  
(bold\_italicize\_labels\_levels), 42
- italicize\_levels(), 38
- knitr::kable, 38, 39
- list, formula, and selector syntax, 5, 6, 9, 10, 12, 13, 22, 30, 32, 33, 45, 61–63, 66, 71, 73, 82, 84, 87, 93, 102, 104, 108, 111
- lm, 109
- modifications, 92, 110
- modify, 5, 10, 12–14, 17, 18, 22, 23, 25–27, 32, 43, 45, 52, 53, 55, 56, 57, 73, 87, 90, 93–95, 98, 102, 104, 108, 111
- modify\_caption(modify), 57
- modify\_cols\_merge, 60, 61–64, 67
- modify\_column\_alignment, 60, 61, 62–64, 67
- modify\_column\_hide, 60, 61, 62, 63, 64, 67
- modify\_column\_unhide  
(modify\_column\_hide), 62
- modify\_fmt\_fun, 60–62, 62, 64, 67
- modify\_footnote(modify), 57
- modify\_footnote(), 87
- modify\_header(modify), 57
- modify\_spanning\_header(modify), 57
- modify\_table\_body, 60–63, 64, 67
- modify\_table\_styling, 60–64, 65
- plot, 67
- pool\_and\_tidy\_mice(custom\_tidiers), 47
- print.tbl\_split(tbl\_split), 93
- proportion\_summary, 18, 46, 68, 70, 87
- proportion\_summary(), 85
- ratio\_summary, 18, 46, 69, 69, 87
- ratio\_summary(), 85
- remove\_row\_type, 71
- reset\_gtsummary\_theme  
(set\_gtsummary\_theme), 74
- reset\_gtsummary\_theme(), 114
- select\_helpers, 72
- separate\_p\_footnotes, 5, 12, 17, 18, 22, 25, 26, 32, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 102, 108
- set\_gtsummary\_theme, 74
- set\_gtsummary\_theme(), 114
- show\_header\_names(modify), 57
- sort\_filter\_p, 75
- sort\_p(sort\_filter\_p), 75
- stats::anova, 45
- stats::anova(), 44, 45
- stats::p.adjust, 26
- stats::poisson.test(), 70
- stats::prop.test(), 68
- stats::update, 45
- style\_number, 76, 77–80
- style\_percent, 76, 77, 78–80
- style\_percent(), 104
- style\_pvalue, 6, 19–21, 23, 25, 26, 50, 53, 54, 76, 77, 78, 79, 80, 92, 110
- style\_ratio, 76–78, 79, 80, 92, 110
- style\_sigfig, 76–79, 80, 92, 104, 110
- style\_sigfig(), 6, 33
- survival::coxph, 109
- syntax, 100, 108
- tbl\_butcher, 81
- tbl\_continuous, 17, 82
- tbl\_cross, 20, 50, 83
- tbl\_custom\_summary, 5, 12, 17, 18, 22, 26, 32, 43, 46, 53, 55, 59, 69, 70, 73, 85, 90, 94, 95, 98, 102
- tbl\_custom\_summary(), 46, 68, 69
- tbl\_merge, 5, 10, 12–14, 17, 18, 22, 23, 25–27, 32, 43, 45, 52, 53, 55, 56, 59, 73, 87, 89, 93–95, 98, 102, 104, 108, 111
- tbl\_regression, 10, 27, 34, 36–39, 42, 43, 45, 51, 52, 59, 90, 91, 94, 95, 98, 108
- tbl\_split, 5, 10, 12–14, 17, 18, 22, 23, 25–27, 32, 43, 45, 52, 53, 55, 56, 59,

*73, 87, 90, 93, 93, 95, 98, 102, 104, 108, 111*

`tbl_stack`, *5, 10, 12–14, 17, 18, 22, 23, 25–27, 32, 43, 45, 52, 53, 55, 56, 59, 73, 87, 90, 93, 94, 94, 98, 102, 104, 108, 111*

`tbl_strata`, *5, 10, 12–14, 17, 18, 22, 23, 25–27, 32, 43, 45, 52, 53, 55, 56, 59, 73, 87, 90, 93–95, 96, 102, 104, 108, 111*

`tbl_strata2` (`tbl_strata`), *96*

`tbl_summary`, *5, 12, 17, 19, 21, 22, 26, 31, 32, 34, 36–39, 42, 43, 53, 55, 59, 73, 87, 90, 94, 95, 98, 99*

`tbl_summary()`, *46, 85, 86, 105, 107*

`tbl_survfit`, *13, 14, 23, 54, 59, 90, 94, 95, 98, 103*

`tbl_survfit.data.frame()`, *103*

`tbl_survfit.list()`, *103*

`tbl_survfit.survfit()`, *103*

`tbl_svsummary`, *12, 17, 18, 24–26, 31, 32, 59, 73, 90, 94, 95, 98, 105*

`tbl_uvregression`, *10, 27, 43, 56, 59, 90, 94, 95, 98, 108*

`tests`, *6, 19, 21*

`theme_gtsummary`, *111*

`theme_gtsummary_compact`  
(`theme_gtsummary`), *111*

`theme_gtsummary_continuous2`  
(`theme_gtsummary`), *111*

`theme_gtsummary_eda` (`theme_gtsummary`), *111*

`theme_gtsummary_journal`  
(`theme_gtsummary`), *111*

`theme_gtsummary_language`  
(`theme_gtsummary`), *111*

`theme_gtsummary_mean_sd`  
(`theme_gtsummary`), *111*

`theme_gtsummary_printer`  
(`theme_gtsummary`), *111*

`tibble`, *42*

`tidy_bootstrap` (`custom_tidiers`), *47*

`tidy_gam` (`custom_tidiers`), *47*

`tidy_robust` (`custom_tidiers`), *47*

`tidy_standardize` (`custom_tidiers`), *47*

`trial`, *114*