

# Package ‘gtsummary’

February 26, 2021

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

**Version** 1.3.7

**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>,  
<http://www.danielsjoberg.com/gtsummary/>

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

**Depends** R (>= 3.4)

**Imports** broom (>= 0.7.3),  
broom.helpers (>= 1.1.0),  
dplyr (>= 1.0.1),  
forcats (>= 0.5.0),  
glue (>= 1.4.1),  
gt (>= 0.2.2),  
knitr (>= 1.29),  
lifecycle (>= 0.2.0),  
purrr (>= 0.3.4),  
rlang (>= 0.4.10),  
stringr (>= 1.4.0),  
survival,  
tibble (>= 3.0.3),  
tidyr (>= 1.1.1),  
usethis (>= 1.6.1)

**Suggests** broom.mixed (>= 0.2.6),  
car,  
covr,  
effectsize,  
flextable (>= 0.5.10),

geepack,  
Hmisc,  
huxtable (>= 5.0.0),  
kableExtra,  
lme4,  
mice,  
nnet,  
officer,  
parameters,  
pkgdown,  
rmarkdown,  
scales,  
spelling (>= 2.2),  
survey,  
testthat

**VignetteBuilder** knitr

**RdMacros** lifecycle

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1

## R topics documented:

add_difference . . . . .	3
add_glance_source_note . . . . .	5
add_global_p . . . . .	6
add_n . . . . .	8
add_n.tbl_summary . . . . .	8
add_n.tbl_survfit . . . . .	10
add_nevent . . . . .	11
add_nevent.tbl_regression . . . . .	11
add_nevent.tbl_survfit . . . . .	12
add_nevent.tbl_uvregression . . . . .	13
add_overall . . . . .	14
add_p . . . . .	15
add_p.tbl_cross . . . . .	16
add_p.tbl_summary . . . . .	17
add_p.tbl_survfit . . . . .	18
add_p.tbl_svsummary . . . . .	20
add_q . . . . .	22
add_stat . . . . .	23
add_stat_label . . . . .	25
as_flex_table . . . . .	27
as_gt . . . . .	29
as_hux_table . . . . .	30
as_kable . . . . .	31
as_kable_extra . . . . .	32
as_tibble.gtsummary . . . . .	33

bold_italicize_labels_levels . . . . .	34
bold_p . . . . .	35
combine_terms . . . . .	36
custom_tidiers . . . . .	37
inline_text . . . . .	39
inline_text.tbl_cross . . . . .	40
inline_text.tbl_regression . . . . .	41
inline_text.tbl_summary . . . . .	42
inline_text.tbl_survfit . . . . .	44
inline_text.tbl_uvregression . . . . .	45
modify . . . . .	47
modify_column_hide . . . . .	49
modify_table_body . . . . .	50
modify_table_header . . . . .	51
print_gtsummary . . . . .	53
remove_row_type . . . . .	53
select_helpers . . . . .	54
set_gtsummary_theme . . . . .	55
sort_filter_p . . . . .	56
style_number . . . . .	57
style_percent . . . . .	58
style_pvalue . . . . .	59
style_ratio . . . . .	60
style_sigfig . . . . .	61
tbl_cross . . . . .	62
tbl_merge . . . . .	63
tbl_regression . . . . .	65
tbl_regression_methods . . . . .	67
tbl_stack . . . . .	69
tbl_summary . . . . .	70
tbl_survfit . . . . .	74
tbl_svysummary . . . . .	77
tbl_uvregression . . . . .	80
tests . . . . .	83
theme_gtsummary . . . . .	85
trial . . . . .	88

**Index****89**

add\_difference

*Add difference between groups***Description**

**[Experimental]** 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 = style_sigfig
)
```

**Arguments**

x	"tbl_summary" 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 tidyselect 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	Function to round and format difference. Default is <a href="#">style_sigfig()</a>

**Example Output****Examples**

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

```

add_n() %>%
add_difference()

# Example 2 -----
add_difference_ex2 <-
  trial %>%
  select(trt, response, death) %>%
  tbl_summary(by = trt,
              statistic = all_dichotomous() ~ "{p}%",
              missing = "no") %>%
  modify_footnote(all_stat_cols() ~ NA) %>%
  add_n() %>%
  add_difference(estimate_fun = ~paste0(style_sigfig(. * 100), "%"))

```

---

add\_glance\_source\_note

*Add glance statistics*


---

## Description

**[Experimental]** Add the statistics returned in `broom::glance()` as a table source note.

## Usage

```

add_glance_source_note(
  x,
  include = everything(),
  label = NULL,
  fmt_fun = NULL,
  glance_fun = broom::glance,
  sep1 = " = ",
  sep2 = "; ",
  ...
)

```

## Arguments

<code>x</code>	'tbl_regression' object
<code>include</code>	tidyselect list of statistics to include. Default is <code>everything()</code>
<code>label</code>	use to update statistic labels
<code>fmt_fun</code>	use to update default formatting function. Default is <code>everything() ~ purrr::partial(style_sigfig, 3)</code>
<code>glance_fun</code>	function to calculate and return glance statistics. Default is <code>broom::glance()</code>
<code>sep1</code>	Separator between statistic name and statistic. Default is <code>" = "</code> , e.g. <code>"R2 = 0.456"</code>
<code>sep2</code>	Separator between statistics. Default is <code>" ; "</code>
<code>...</code>	additional arguments passed to <code>broom::glance()</code>

## Default Labels

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

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

Example Output

Examples

```
# Example 1 -----
add_glance_source_note_ex1 <-
  lm(age ~ marker + grade, trial) %>%
  tbl_regression() %>%
  add_glance_source_note(
    label = list(df ~ "Degrees of Freedom", sigma ~ "\u03C3"),
    fmt_fun = df ~ style_number,
    include = c(r.squared, AIC, sigma, df)
  )
```

---

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 <a href="#">car::Anova</a>
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 <a href="#">car::Anova</a> . 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 FALSE
quiet	Logical indicating whether to print messages in console. Default is FALSE
terms	DEPRECATED. Use <code>include=</code> argument instead.

### Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

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

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

### 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	Adds column with N to gtsummary table
-------	---------------------------------------

---

**Description**

Adds column with N to gtsummary table

**Usage**

```
add_n(x, ...)
```

**Arguments**

- x                    Object created from a gtsummary function
- ...                  Additional arguments passed to other methods.

**Author(s)**

Daniel D. Sjoberg

**See Also**

```
add\_n.tbl\_summary\(\), add\_n.tbl\_svsummary\(\), add\_n.tbl\_survfit\(\)
```

---

add_n.tbl_summary	Add column with N
-------------------	-------------------

---

**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_svysummary'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  missing = 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
statistic	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>
col_label	String indicating the column label. Default is <code>"**N**"</code>
footnote	Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is <code>FALSE</code>
last	Logical indicator to include N column last in table. Default is <code>FALSE</code> , which will display N column first.
missing	DEPRECATED. Logical argument indicating whether to print N ( <code>missing = FALSE</code> ), or N missing ( <code>missing = TRUE</code> ). Default is <code>FALSE</code>
...	Not used

### Value

A `tbl_summary` or `tbl_svysummary` object

### Example Output

### Author(s)

Daniel D. Sjoberg

See Also

Other `tbl_summary` tools: `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_merge()`, `tbl_stack()`, `tbl_summary()`

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

Examples

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

---

add_n.tbl_survfit	Add column with number of observations
-------------------	--

---

Description

**[Experimental]** 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

Other `tbl_survfit` tools: `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `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	<i>Add number of events to a regression table</i>
------------	---

---

### Description

Adds a column of the number of events to tables created with [tbl\\_regression](#) or [tbl\\_uvregression](#). Supported model types include GLMs with binomial distribution family (e.g. [stats::glm](#), [lme4::glmer](#), and [geepack::geeglm](#)) and Cox Proportion Hazards regression models ([survival::coxph](#)).

### Usage

```
add_nevent(x, ...)
```

### Arguments

x	tbl_regression or tbl_uvregression object
...	Additional arguments passed to or from other methods.

### Author(s)

Daniel D. Sjoberg

### See Also

[add\\_nevent.tbl\\_regression](#), [add\\_nevent.tbl\\_uvregression](#), [tbl\\_regression](#), [tbl\\_uvregression](#)

---

add_nevent.tbl_regression	<i>Add number of events to a regression table</i>
---------------------------	---

---

### Description

This function adds a column of the number of events to tables created with [tbl\\_regression](#). Supported model types include GLMs with binomial distribution family (e.g. [stats::glm](#), [lme4::glmer](#), and [geepack::geeglm](#)) and Cox Proportion Hazards regression models ([survival::coxph](#)).

The number of events is added to the internal `.$table_body` tibble, and not printed in the default output table (similar to `N`). The number of events is accessible via the [inline\\_text](#) function for printing in a report.

### Usage

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

### Arguments

x	tbl_regression object
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Not used

**Value**

A tbl\_regression object

**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\\_stack\(\)](#)

**Examples**

```
add_nevent_ex <-
  glm(response ~ trt, trial, family = binomial) %>%
  tbl_regression() %>%
  add_nevent()
```

---

add\_nevent.tbl\_survfit

*Add column with number of observed events*

---

**Description**

**[Experimental]** 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\\_stack\(\)](#), [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.tbl_uvregression
```

*Add number of events to a regression table*

---

**Description**

Adds a column of the number of events to tables created with [tbl\\_uvregression](#). Supported model types include GLMs with binomial distribution family (e.g. [stats::glm](#), [lme4::glmer](#), and [geepack::geeglm](#)) and Cox Proportion Hazards regression models ([survival::coxph](#)).

**Usage**

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

**Arguments**

x	tbl_uvregression object
...	Not used

**Value**

A `tbl_uvregression` object

**Reporting Event N**

The number of events is added to the internal `.$table_body` tibble, and printed to the right of the N column. The number of events is also accessible via the [inline\\_text](#) function for printing in a report.

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

Daniel D. Sjoberg

**See Also**

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

## Examples

```
tbl_uv_nevent_ex <-
  trial[c("response", "trt", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    y = response,
    method.args = list(family = binomial)
  ) %>%
  add_nevent()
```

---

add_overall	<i>Add column with overall summary statistics</i>
-------------	---

---

## Description

Adds a column with overall summary statistics to tables created by `tbl_summary` or `tbl_svysummary`.

## Usage

```
add_overall(x, last, col_label)

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

## S3 method for class 'tbl_svysummary'
add_overall(x, last = FALSE, col_label = NULL)
```

## Arguments

<code>x</code>	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function or object with class <code>tbl_svysummary</code> from the <a href="#">tbl_svysummary</a> function.
<code>last</code>	Logical indicator to display overall column last in table. Default is <code>FALSE</code> , which will display overall column first.
<code>col_label</code>	String indicating the column label. Default is <code>"**Overall**, N = {N}"</code>

## Value

A `tbl_summary` object or a `tbl_svysummary` object

## Example Output

## Author(s)

Daniel D. Sjoberg

See Also

Other `tbl_summary` tools: [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](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_summary\(\)](#)

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

Examples

```
tbl_overall_ex <-  
  trial[c("age", "grade", "trt")] %>%  
  tbl_summary(by = trt) %>%  
  add_overall()
```

---

add_p	Adds p-values to gtsummary table
-------	----------------------------------

---

Description

Adds p-values to gtsummary table

Usage

```
add_p(x, ...)
```

Arguments

- x                    Object created from a gtsummary function
- ...                  Additional arguments passed to other methods.

Author(s)

Daniel D. Sjoberg

See Also

[add\\_p.tbl\\_summary](#), [add\\_p.tbl\\_cross](#), [add\\_p.tbl\\_svysummary](#), [add\\_p.tbl\\_survfit](#)

---

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, ...)
```

## 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.
...	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

<code>x</code>	Object with class <code>tbl_summary</code> from the <a href="#">tbl_summary</a> function
<code>test</code>	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 $< 5$ .
<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>group</code>	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.
<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> .
<code>test.args</code>	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>
<code>exclude</code>	DEPRECATED
<code>...</code>	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

Other `tbl_summary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `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)
  )
```

---

<code>add_p.tbl_survfit</code>	<i>Adds p-value to survfit table</i>
--------------------------------	--------------------------------------

---

**Description**

**[Experimental]** 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

<code>x</code>	Object of class "tbl_survfit"
<code>test</code>	string indicating test to use. Must be one of "logrank", "survdiff", "petopeto_gehanwilcoxon", "coxph_lrt", "coxph_wald", "coxph_score". See details below
<code>test.args</code>	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>
<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>include</code>	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> .
<code>quiet</code>	Logical indicating whether to print messages in console. Default is FALSE
<code>...</code>	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\\_stack\(\)](#), [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 `survdifff()`
add_p_tbl_survfit_ex2 <-
  gts_survfit %>%
  add_p(test = "survdifff", 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. <a href="#">survey::svyttest</a>),</li> <li>"svy.wilcox.test" for a Wilcoxon rank-sum test for complex survey samples (cf. <a href="#">survey::svyranktest</a>),</li> <li>"svy.kruskal.test" for a Kruskal-Wallis rank-sum test for complex survey samples (cf. <a href="#">survey::svyranktest</a>),</li> <li>"svy.vanderwaerden.test" for a van der Waerden's normal-scores test for complex survey samples (cf. <a href="#">survey::svyranktest</a>),</li> <li>"svy.median.test" for a Mood's test for the median for complex survey samples (cf. <a href="#">survey::svyranktest</a>),</li> <li>"svy.chisq.test" for a Chi-squared test with Rao &amp; Scott's second-order correction (cf. <a href="#">survey::svychisq</a>),</li> <li>"svy.adj.chisq.test" for a Chi-squared test adjusted by a design effect estimate (cf. <a href="#">survey::svychisq</a>),</li> </ul>

- "svy.wald.test" for a Wald test of independence for complex survey samples (cf. [survey::svychisq](#)),
- "svy.adj.wald.test" for an adjusted Wald test of independence for complex survey samples (cf. [survey::svychisq](#)),
- "svy.lincom.test" for a test of independence using the exact asymptotic distribution for complex survey samples (cf. [survey::svychisq](#)),
- "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 <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 tidyselect 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.tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [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\\_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](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_summary\(\)](#)

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

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

Other `tbl_uvregression` tools: `add_global_p()`, `add_nevent.tbl_uvregression()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `modify.tbl_merge()`, `tbl_stack()`, `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_stat	Add a custom statistic column
----------	-------------------------------

---

Description

**[Experimental]** The function allows a user to add a new column with a custom, user-defined statistic.

Usage

```
add_stat(
  x,
  fns,
  fmt_fun = NULL,
  header = "**Statistic**",
  footnote = NULL,
  new_col_name = NULL,
  location = c("label", "level")
)
```

Arguments

- x                      `tbl_summary` or `tbl_svsummary` object
- fns                    list of formulas indicating the functions that create the statistic

fmt_fun	for numeric statistics, fmt_fun= is the styling/formatting function. Default is NULL
header	Column header of new column. Default is "**Statistic**"
footnote	Footnote associated with new column. Default is no footnote (i.e. NULL)
new_col_name	name of new column to be created in .stable_body. Default is "add_stat_1", unless that column exists then it is "add_stat_2", etc.
location	Must be one of c("label", "level") and indicates which row(s) the new statistics are placed on. 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 fns function must match the dimension of levels. Continuous and dichotomous statistics are placed on the variable label row.

## Details

The custom functions passed in fns= are required to follow a specified format. Each of these function will execute on a single variable from tbl\_summary()/tbl\_svsummary().

1. Each function must return a single scalar or character value of length one when location = "label". When location = "level", the returned statistic must be a vector of the length of the number of levels (excluding the row for unknown values).
2. 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
  - by= is a string indicating the by variable from tbl\_summary=, if present
  - tbl= the original tbl\_summary() 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, ...)

## Example Output

## Examples

```
# Example 1 -----
# this example replicates `add_p()`

# 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_p(test = everything() ~ t.test) %>%
  # replicating result of `add_p()` with `add_stat()`
  add_stat(
```



```

    fns = everything() ~ my_ttest, # all variables compared with with t-test
    fmt_fun = style_pvalue,        # format result with style_pvalue()
    header = "**My p-value**"      # new column header
  )

# Example 2 -----
# fn returns t-test test statistic and pvalue
my_ttest2 <- function(data, variable, by, ...) {
  tt <- t.test(data[[variable]] ~ as.factor(data[[by]]))

  # returning test statistic and pvalue
  stringr::str_glue(
    "t={style_sigfig(tt$statistic)}, {style_pvalue(tt$p.value, prepend_p = TRUE)}"
  )
}

add_stat_ex2 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(
    fns = everything() ~ my_ttest2, # all variables will be compared by t-test
    fmt_fun = NULL, # fn returns and chr, so no formatting function needed
    header = "**Treatment Comparison**", # column header
    footnote = "T-test statistic and p-value" # footnote
  )

# Example 3 -----
# Add CI for categorical variables
categorical_ci <- function(variable, tbl, ...) {
  dplyr::filter(tbl$meta_data, variable == .env$variable) %>%
  purrr::pluck("df_stats", 1) %>%
  dplyr::mutate(
    # calculate and format 95% CI
    prop_ci = purrr::map2(n, N, ~prop.test(.x, .y)$conf.int %>% style_percent(symbol = TRUE)),
    ci = purrr::map_chr(prop_ci, ~glue::glue("{.x[1]}, {.x[2]}"))
  ) %>%
  dplyr::pull(ci)
}

add_stat_ex3 <-
  trial %>%
  select(grade) %>%
  tbl_summary(statistic = everything() ~ "{p}%") %>%
  add_stat(
    fns = everything() ~ "categorical_ci",
    location = "level",
    header = "**95% CI**"
  ) %>%
  modify_footnote(everything() ~ NA)

```

**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

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

Daniel D. Sjoberg

**See Also**

Other `tbl_summary` tools: [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](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [tbl\\_summary\(\)](#)

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

**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"))

```

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. Default is TRUE

**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. 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,
  omit = 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.
omit	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 = 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. Default is TRUE

## 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::align\(\)](#) to set column alignment
4. [huxtable::set\\_left\\_padding\(\)](#) to indent variable levels
5. [huxtable::add\\_footnote\(\)](#) to add table footnotes and source notes

6. `huxtable::set_bold()` to bold cells
7. `huxtable::set_italic()` to italicize cells
8. `huxtable::set_na_string()` to use an em-dash for missing numbers

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	<i>Convert gtsummary object to a kable object</i>
----------	---

---

### Description

Function converts a gtsummary object to a knitr\_kable object. This 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 `knitr::kable`.

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, ...)
```

### 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. tidyselect and gtsummary 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
...	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

---

*Convert gtsummary object to a kableExtra object*


---

**Description**

Function converts a gtsummary object to a `knitr_kable` + `kableExtra` object. A user can use this function if they wish to add customized formatting available via `knitr::kable` and `kableExtra`. Note that gtsummary uses the standard markdown `**` to bold headers, and they may need to be changed manually with `kableExtra` output.

**Usage**

```
as_kable_extra(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = 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.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed. Default is TRUE
...	Additional arguments passed to <a href="#">knitr::kable</a>

**Value**

A kableExtra object

**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**

```
tbl <-
  trial %>%
  tbl_summary(by = trt) %>%
  as_kable_extra()
```

---

as_tibble.gtsummary	<i>Convert gtsummary object to a tibble</i>
---------------------	---

---

**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,
  ...
)
```

**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().
col_labels	Logical argument adding column labels to output tibble. Default is TRUE.

<code>return_calls</code>	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
<code>exclude</code>	DEPRECATED
<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. Sjoberg

**See Also**

Other `tbl_summary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

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

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

**Examples**

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

---

<code>bold_p</code>	<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

*Combine terms in a regression model*

---

**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**

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

**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"
  )
```

---

 custom\_tidiers

---

*Collection of custom tidiers*


---

**Description**

**[Experimental]** 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
)

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

```

### 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(): mice::tidy(x, ...)</code></li> <li>• <code>tidy_standardize(): effectsize::standardize_parameters(x, ...)</code></li> <li>• <code>tidy_bootstrap(): parameters::bootstrap_parameters(x, ...)</code></li> </ul>
<code>quiet</code>	Logical indicating whether to print messages in console. Default is FALSE
<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.
- `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
pool_and_tidy_mice_ex3 <-
  suppressWarnings(mice::mice(trial, m = 2)) %>%
  with(lm(age ~ marker + grade)) %>%
  tbl_regression() # mice method called that uses `pool_and_tidy_mice()` as tidier
```

---

inline\_text

---

*Report statistics from gtsummary tables inline*


---

**Description**

Report statistics from gtsummary tables inline

**Usage**

```
inline_text(x, ...)
```

**Arguments**

x	Object created from a gtsummary function
...	Additional arguments passed to other methods.

**Value**

A string reporting results from a gtsummary table

**Author(s)**

Daniel D. Sjoberg

**See Also**

[inline\\_text.tbl\\_summary](#), [inline\\_text.tbl\\_regression](#), [inline\\_text.tbl\\_uvregression](#), [inline\\_text.tbl\\_survfit](#)

---

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

---

## Description

**[Experimental]** 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 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'
- {ci} confidence interval formatted with x\$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\\_nevent.tbl\\_regression\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_labels.tbl\\_regression\(\)](#), [combine\\_terms\(\)](#), [modify.tbl\\_regression\(\)](#), [tbl\\_regression\(\)](#), [tbl\\_stack\(\)](#)

### 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_svysummary'
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\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_summary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [bold\\_italicize\\_labels\\_levels](#), [inline\\_text.tbl\\_survfit\(\)](#), [modify](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [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

**[Experimental]** 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 coefficient estimates. Default is <a href="#">style_sigfig</a> when the coefficients are not transformed, and <a href="#">style_ratio</a> when the coefficients have been exponentiated.
<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>	<code>tbl_survfit</code> used

## Value

A string reporting results from a gtsummary table

**Author(s)**

Daniel D. Sjöberg

**See Also**

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\(\)](#), [modify.tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [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

<code>x</code>	Object created from <a href="#">tbl_uvregression</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 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'`
- `{ci}` confidence interval formatted with `x$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\\_nevent.tbl\\_uvregression\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_label\(\)](#), [modify\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#), [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, quiet = NULL)

modify_spanning_header(x, update = NULL, quiet = NULL)

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

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

## Arguments

x	a gtsummary object
update	list of formulas or a single formula specifying the updated column header, footnote, or spanning header. The LHS specifies the column(s) to be updated, and the RHS is the updated text. Use the <code>show_header_names()</code> to see the column names that can be modified.
text_interpret	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>...</code>	Specify a column and updated column label, e.g. <code>modify_header(p.value = "Model P-values")</code> . This is provided as an alternative to the <code>update=</code> argument. They accomplish the same goal of updating column headers.
<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

## Value

Updated gtsummary object

## tbl\_summary(), tbl\_svsummary(), 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_svsummary` 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_svsummary` 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.

## captions

Captions are assigned based on output type.

- `gt::gt(caption=)`, available in gt version >0.2.2
- `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_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()`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

Other `tbl_svsummary` tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svsummary()`, `add_q()`, `add_stat_label()`, `tbl_merge()`, `tbl_stack()`, `tbl_svsummary()`



Other `tbl_regression` tools: `add_global_p()`, `add_nevent.tbl_regression()`, `add_q()`, `bold_italicize_labels_1`, `combine_terms()`, `inline_text.tbl_regression()`, `tbl_merge()`, `tbl_regression()`, `tbl_stack()`

Other `tbl_uvregression` tools: `add_global_p()`, `add_nevent.tbl_uvregression()`, `add_q()`, `bold_italicize_labels_1`, `inline_text.tbl_uvregression()`, `tbl_merge()`, `tbl_stack()`, `tbl_uvregression()`

Other `tbl_survfit` tools: `add_n.tbl_survfit()`, `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `tbl_merge()`, `tbl_stack()`, `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(
    update = list(label ~ "***Variable**",
                  p.value ~ "***P**")
  ) %>%
  modify_footnote(
    update = 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(update = 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_column_hide	<i>Modify Hidden Columns</i>
--------------------	------------------------------

---

## Description

**[Experimental]** Use these functions to hide or unhide columns in a `gtsummary` tables.

**Usage**

```
modify_column_hide(x, column)

modify_column_unhide(x, column)
```

**Arguments**

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

**Example Output****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(column = ci) %>%
  modify_column_unhide(column = std.error)
```

---

modify_table_body	<i>Modify table_body</i>
-------------------	--------------------------

---

**Description**

**[Experimental]** 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\_header. 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	unquoted (bare) function name
...	arguments passed to fun() function. First argument of fun() must be x\$table_body

**Example Output**

**See Also**

modify\_table\_header()

See [gtsummary internals vignette](#)

**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) %>%
  add_nevent() %>%
  # adding number of non-events to table
  modify_table_body(dplyr::mutate, n_nonevent = N - nevent) %>%
  # move new column to before the nevent column
  modify_table_body(dplyr::relocate, n_nonevent, .before = nevent) %>%
  modify_header(n_nonevent = "***Control N**", nevent = "***Case N**")
```

---

modify_table_header	<i>Modify table_header</i>
---------------------	----------------------------

---

**Description**

This is a function meant for advanced users to gain more control over the characteristics of the resulting gtsummary table.

**Usage**

```
modify_table_header(
  x,
  column,
  label = NULL,
  hide = NULL,
  align = NULL,
  missing_emdash = NULL,
  indent = NULL,
  text_interpret = NULL,
  bold = NULL,
  italic = NULL,
  fmt_fun = NULL,
  footnote_abbrev = NULL,
  footnote = NULL,
  spanning_header = NULL
)
```

**Arguments**

x	gtsummary object
column	vector or selector of columns in x\$table_body
label	string of column label
hide	logical indicating whether to hide column from output
align	string indicating alignment of column, must be one of c("left", "right", "center")
missing_emdash	string that evaluates to logical identifying rows to include em-dash for missing values, e.g. "reference_row == TRUE"
indent	string that evaluates to logical identifying rows to indent
text_interpret	string, must be one of "gt::md" or "gt::html"
bold	string that evaluates to logical identifying rows to bold
italic	string that evaluates to logical identifying rows to italicize
fmt_fun	function that formats the statistics in the column
footnote_abbrev	string with abbreviation definition, e.g. "CI = Confidence Interval"
footnote	string with text for column footnote
spanning_header	string with text for spanning header

**Details**

Review the [gtsummary definition](#) vignette for information on . \$table\_header objects.

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

`modify_table_body()`

See [gtsummary internals vignette](#)

**Examples**

```
# Example 1 -----
modify_table_header_ex1 <-
  lm(mpg ~ factor(cyl), mtcars) %>%
  tbl_regression() %>%
  modify_table_header(column = estimate,
                      label = "**Coefficient**",
                      fmt_fun = function(x) style_sigfig(x, digits = 5),
                      footnote = "Regression Coefficient") %>%
  modify_table_header(column = "p.value",
                      hide = TRUE)
```

---

print_gtsummary	<i>print and knit_print methods for gtsummary objects</i>
-----------------	---

---

**Description**

print and knit\_print methods for gtsummary objects

**Usage**

```
## S3 method for class 'gtsummary'
print(x, print_engine = NULL, ...)

## S3 method for class 'gtsummary'
knit_print(x, ...)
```

**Arguments**

x	An object created using gtsummary functions
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"
...	Not used

**Author(s)**

Daniel D. Sjoberg

**See Also**

[tbl\\_summary](#) [tbl\\_regression](#) [tbl\\_uvregression](#) [tbl\\_merge](#) [tbl\\_stack](#)

---

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

### 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

### Examples

```

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

```

---

set_gtsummary_theme	<i>Set a gtsummary theme</i>
---------------------	------------------------------

---

### Description

**[Experimental]** 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

*Sort and filter variables in table by p-values*

---

**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	Style numbers
--------------	---------------

---

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 "." orgetOption("OutDec")
scale	A scaling factor: x will be multiplied by scale before formatting.
...	Other arguments passed on to base::format()

**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 <code>getOption("OutDec")</code>
...	Other arguments passed on to <code>base::format()</code>

**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**

<code>x</code>	Numeric vector of p-values.
<code>digits</code>	Number of digits large p-values are rounded. Must be 1, 2, or 3. Default is 1.
<code>prepend_p</code>	Logical. Should 'p=' be prepended to formatted p-value. Default is FALSE
<code>big.mark</code>	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ", ", 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 "." or <code>getOption("OutDec")</code>
<code>...</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, big.mark = NULL, decimal.mark = NULL, ...)
```

## Arguments

x	Numeric vector
digits	Integer specifying the minimum number of significant digits to display
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\_cross

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

**[Experimental]** The function creates a cross table of two categorical variables.

**Usage**

```
tbl_cross(
  data,
  row = NULL,
  col = NULL,
  label = NULL,
  statistic = 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 columns of cross table.
col	A column name in data to be used for rows 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> .
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

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_merge	<i>Merge two or more gtsummary objects</i>
-----------	--

---

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

- tbls                      List of gtsummary objects to merge
- tab\_spanner              Character vector specifying the spanning headers. Must be the same length as tbls. The strings are interpreted with gt::md. Must be same length as tbls argument

Value

A tbl\_merge object

Example Output

**Author(s)**

Daniel D. Sjoberg

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

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

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

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.tbl\\_stack\(\)](#), [tbl\\_summary\(\)](#)

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

Other `tbl_svysummary` tools: [add\\_n.tbl\\_summary\(\)](#), [add\\_overall\(\)](#), [add\\_p.tbl\\_svysummary\(\)](#), [add\\_q\(\)](#), [add\\_stat\\_label\(\)](#), [modify.tbl\\_stack\(\)](#), [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 = broom::tidy,
  add_estimate_to_reference_rows = FALSE,
  show_yesno = NULL,
  exclude = 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 tidyselect 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.
conf.level	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

intercept	Logical argument indicating whether to include the intercept in the output. Default is FALSE
estimate_fun	Function to round and format coefficient estimates. Default is <a href="#">style_sigfig</a> when the coefficients are not transformed, and <a href="#">style_ratio</a> when the coefficients have been exponentiated.
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> ).
tidy_fun	Option to specify a particular tidier function if the model is not a <a href="#">vetted model</a> or you need to implement a custom method. Default is NULL
add_estimate_to_reference_rows	add a reference value. Default is FALSE
show_yesno	DEPRECATED
exclude	DEPRECATED

### 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](#).

- "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)
- "lmerMod", "glmerMod", "glmmTMB", "glmmadmb", "stanreg": These mixed effects models use `broom.mixed::tidy(x, effects = "fixed")`. Specify `tidy_fun = broom.mixed::tidy` to print the random components.

### Note

The N reported in the output is the number of observations in the data frame `model.frame(x)`. Depending on the model input, this N may represent different quantities. In most cases, it is the number of people or units in your model. Here are some common exceptions.

1. Survival regression models including time dependent covariates.
2. Random- or mixed-effects regression models with clustered data.
3. GEE regression models with clustered data.

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 examplesOther `tbl_regression` tools: `add_global_p()`, `add_nevent.tbl_regression()`, `add_q()`, `bold_italicize_labels.tbl_regression()`, `combine_terms()`, `inline_text.tbl_regression()`, `modify.tbl_regression()`, `tbl_merge()`, `tbl_stack()`**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\_regression\_methods

*Methods for tbl\_regression***Description**

Most regression models are handled by `tbl_regression.default()`, which uses `broom::tidy()` to perform initial tidying of results. There are, however, some model types that have modified default printing behavior. Those methods are listed below.

**Usage**

```
## S3 method for class 'survreg'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom::tidy(x, ...) %>% dplyr::filter(.data$term !=
    "Log(scale)"),
  ...
)

## S3 method for class 'mira'
tbl_regression(x, tidy_fun = pool_and_tidy_mice, ...)
```

```

## S3 method for class 'mipo'
tbl_regression(x, ...)

## S3 method for class 'lmerMod'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

## S3 method for class 'glmerMod'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

## S3 method for class 'glmmTMB'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

## S3 method for class 'glmmadmb'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

## S3 method for class 'stanreg'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

## S3 method for class 'multinom'
tbl_regression(x, ...)

```

### Arguments

<code>x</code>	Regression model object
<code>tidy_fun</code>	Option to specify a particular tidier function if the model is not a <a href="#">vetted model</a> or you need to implement a custom method. Default is NULL
<code>...</code>	arguments passed to <code>tbl_regression.default()</code>

### 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](#).

- "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)
- "lmerMod", "glmerMod", "glmmTMB", "glmmadmb", "stanreg": These mixed effects models use `broom.mixed::tidy(x, effects = "fixed")`. Specify `tidy_fun = broom.mixed::tidy` to print the random components.

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

<code>tbls</code>	List of <code>gtsummary</code> objects
<code>group_header</code>	Character vector with table headers where length matches the length of <code>tbls</code>
<code>quiet</code>	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\\_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\\_merge\(\)](#), [tbl\\_summary\(\)](#)

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

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

Other `tbl_uvregression` tools: `add_global_p()`, `add_nevent.tbl_uvregression()`, `add_q()`, `bold_italicize_label`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_uvregression()`

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

## 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(),
  group = 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.
<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 <code>label</code> attribute ( <code>attr(data\$age, "label")</code> ) is used. If attribute <code>label</code> 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 continuous 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.
<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 <code>"no"</code> (never display NA values), <code>"ifany"</code> (only display if any NA values), and <code>"always"</code> (includes NA count row for all variables). Default is <code>"ifany"</code> .

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>
group	DEPRECATED. Migrated to <a href="#">add_p</a>

## 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(all_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.)

## 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
- {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

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

## 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

*Creates table of survival probabilities*

## Description

**[Experimental]** 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

<code>x</code>	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.
<code>...</code>	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.
<code>times</code>	numeric vector of times for which to return survival probabilities.
<code>probs</code>	numeric vector of probabilities with values in (0,1) specifying the survival quantiles to return
<code>statistic</code>	string defining the statistics to present in the table. Default is "{estimate} ({conf.low},{conf.high})"
<code>label</code>	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.
<code>label_header</code>	string specifying column labels above statistics. Default is "{prob} Percentile" for survival percentiles, and "Time {time}" for n-year survival estimates
<code>estimate_fun</code>	function to format the Kaplan-Meier estimates. Default is <code>style_percent()</code> for survival probabilities and <code>style_sigfig</code> for survival times
<code>missing</code>	text to fill when estimate is not estimable. Default is "--"
<code>conf.level</code>	Confidence level for confidence intervals. Default is 0.95
<code>reverse</code>	Flip the probability reported, i.e. 1 -estimate. Default is FALSE. Does not apply to survival quantile requests
<code>quiet</code>	Logical indicating whether to print messages in console. Default is FALSE
<code>y</code>	outcome call, e.g. <code>y = Surv(ttdeath, death)</code>
<code>include</code>	Variable to include as stratifying variables.

### Example Output

**Author(s)**

Daniel D. Sjoberg

**See Also**

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

**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)
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**

**[Experimental]** 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 = NULL
)
```

**Arguments**

<code>data</code>	A survey object created with created with <code>survey::svydesign()</code>
<code>by</code>	A column name (quoted or unquoted) in data. 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.
<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 continuous 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_svysummary` 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(all_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.)

### Author(s)

Joseph Larmarange

**See Also**

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

**Examples**

```
# Example 1 -----
# 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,
show_ynsno = NULL,
exclude = NULL
)

```

## Arguments

data	Data frame to be used in univariate regression modeling. Data frame includes the outcome variable(s) and the independent variables.
method	Regression method (e.g. <a href="#">lm</a> , <a href="#">glm</a> , <a href="#">survival::coxph</a> , and more).
y	Model outcome (e.g. <code>y = recurrence</code> or <code>y = Surv(time, recur)</code> ). All other column in data will be regressed on y. Specify one and only one of y or x
x	Model covariate (e.g. <code>x = trt</code> ). All other columns in data will serve as the outcome in a regression model with x as a covariate. Output table is best when x is a continuous or dichotomous variable displayed on a single row. Specify one and only one of y or x
method.args	List of additional arguments passed on to the regression function defined by method.
exponentiate	Logical indicating whether to exponentiate the coefficient estimates. Default is FALSE.
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</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> .
tidy_fun	Option to specify a particular tidier function if the model is not a <a href="#">vetted model</a> or you need to implement a custom method. Default is NULL
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 <a href="#">style_sigfig</a> when the coefficients are not transformed, and <a href="#">style_ratio</a> when the coefficients have been exponentiated.
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> ).
formula	String of the model formula. Uses <a href="#">glue::glue</a> syntax. Default is <code>"{y} ~ {x}"</code> , where {y} is the dependent variable, and {x} 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
show_yesno	DEPRECATED
exclude	DEPRECATED

### 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](#).

- "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)
- "lmerMod", "glmerMod", "glmmTMB", "glmmadmb", "stanreg": These mixed effects models use `broom.mixed::tidy(x, effects = "fixed")`. Specify `tidy_fun = broom.mixed::tidy` to print the random components.

### Note

The N reported in the output is the number of observations in the data frame `model.frame(x)`. Depending on the model input, this N may represent different quantities. In most cases, it is the number of people or units in your model. Here are some common exceptions.

1. Survival regression models including time dependent covariates.
2. Random- or mixed-effects regression models with clustered data.
3. GEE regression models with clustered data.

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

Other `tbl_uvregression` tools: [add\\_global\\_p\(\)](#), [add\\_nevent.tbl\\_uvregression\(\)](#), [add\\_q\(\)](#), [bold\\_italicize\\_label.tbl\\_uvregression\(\)](#), [inline\\_text.tbl\\_uvregression\(\)](#), [modify.tbl\\_uvregression\(\)](#), [tbl\\_merge\(\)](#), [tbl\\_stack\(\)](#)

## 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)
  )
```

---

tests	<i>Tests/methods available in add_p() and add_difference()</i>
-------	--

---

## Description

Below is a listing of tests available internally within gtsummary.

Tests listed with . . . may have additional arguments passed to them using `add_p(test.args=)`. For example, to calculate a p-value from `t.test()` assuming equal variance, use `tbl_summary(trial, by = trt) %>% add_p(age ~ "t.test", test.args = age ~ list(var.equal = TRUE))`

### tbl\_summary() %>% add\_p()

alias	description	pseudo-code
"t.test"	t-test	<code>t.test(variable ~ as.factor(by), data = data)</code>
"aov"	One-way ANOVA	<code>aov(variable ~ as.factor(by), data = data) %&gt;%</code>
"kruskal.test"	Kruskal-Wallis test	<code>kruskal.test(data[[variable]], as.factor(by))</code>
"wilcox.test"	Wilcoxon rank-sum test	<code>wilcox.test(variable ~ as.factor(by), data = data)</code>
"chisq.test"	chi-square test of independence	<code>chisq.test(x = data[[variable]], y = as.factor(by))</code>
"chisq.test.no.correct"	chi-square test of independence	<code>chisq.test(x = data[[variable]], y = as.factor(by), correct = FALSE)</code>
"fisher.test"	Fisher's exact test	<code>fisher.test(data[[variable]], as.factor(by))</code>
"mcnemar.test"	McNemar's test	<code>mcnemar.test(data[[variable]], data[[by]])</code>
"lme4"	random intercept logistic regression	<code>lme4::glmer(by ~ (1   UFF5C group), data, family = binomial)</code>
"paired.t.test"	Paired t-test	<code>tidyr::pivot_wider(id_cols = group, ...); t.test(by_1, by_2)</code>
"paired.wilcox.test"	Paired Wilcoxon rank-sum test	<code>tidyr::pivot_wider(id_cols = group, ...); wilcox.test(by_1, by_2)</code>
"prop.test"	Test for equality of proportions	<code>prop.test(x, n, conf.level = 0.95, ...)</code>
"ancova"	ANCOVA	<code>lm(variable ~ by + adj.vars)</code>

**tbl\_svysummary() %>% add\_p()**

alias	description
"svy.t.test"	t-test adapted to complex survey samples
"svy.wilcox.test"	Wilcoxon rank-sum test for complex survey samples
"svy.kruskal.test"	Kruskal-Wallis rank-sum test for complex survey samples
"svy.vanderwaarden.test"	van der Waerden's normal-scores test for complex survey samples
"svy.median.test"	Mood's test for the median for complex survey samples
"svy.chisq.test"	chi-squared test with Rao & Scott's second-order correction
"svy.adj.chisq.test"	chi-squared test adjusted by a design effect estimate
"svy.wald.test"	Wald test of independence for complex survey samples
"svy.adj.wald.test"	adjusted Wald test of independence for complex survey samples
"svy.lincom.test"	test of independence using the exact asymptotic distribution for complex survey samples
"svy.saddlepoint.test"	test of independence using a saddlepoint approximation for complex survey samples

**tbl\_survfit() %>% add\_p()**

alias	description	pseudo-code
"logrank"	Log-rank test	survival::survdiff(Surv(.) ~ v
"petopeto_gehanwilcoxon"	Peto & Peto modification of Gehan-Wilcoxon test	survival::survdiff(Surv(.) ~ v
"survdiff"	G-rho family test	survival::survdiff(Surv(.) ~ v
"coxph_lrt"	Cox regression (LRT)	survival::coxph(Surv(.) ~ vari
"coxph_wald"	Cox regression (Wald)	survival::coxph(Surv(.) ~ vari
"coxph_score"	Cox regression (Score)	survival::coxph(Surv(.) ~ vari

**tbl\_summary() %>% add\_difference()**

alias	description	difference statistic	pseudo-code
"t.test"	t-test	mean difference	t.test(variable ~ as.f
"paired.t.test"	Paired t-test	mean difference	tidyr::pivot_wider(id_cols
"paired.wilcox.test"	Paired Wilcoxon rank-sum test	rate difference	tidyr::pivot_wider(id_cols
"prop.test"	Test for equality of proportions	rate difference	prop.test(x, n, conf.l
"ancova"	ANCOVA	mean difference	lm(variable ~ by + adj.
"ancova_lme4"	ANCOVA with random intercept	mean difference	lme4::lmer(variable ~ by +
"cohens_d"	Cohen's D	standardized mean difference	effectsize::cohens_d

**Custom Functions**

To report a p-value (or difference) for a test not available in gtsummary, you can create a custom function. The output is a data frame that is one line long. The structure is similar to the output of

`broom::tidy()` of a typical statistical test. The `add_p()` and `add_comparison()` functions will look for columns called "p.value", "estimate", "conf.low", "conf.high", and "method" for the p-value, difference, confidence interval, and the test name used in the footnote.

Example calculating a p-value from a t-test assuming a common variance between groups.

```
ttest_common_variance <- function(data, variable, by, ...) {
  data <- data[c(variable, by)] %>% dplyr::filter(complete.cases(.))
  t.test(data[[variable]] ~ factor(data[[by]]), var.equal = TRUE) %>%
  broom::tidy()
}

trial[c("age", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p(test = age ~ "ttest_common_variance")
```

A custom `add_difference()` is similar, and accepts arguments `conf.level=` and `adj.vars=` as well.

```
ttest_common_variance <- function(data, variable, by, conf.level, ...) {
  data <- data[c(variable, by)] %>% dplyr::filter(complete.cases(.))
  t.test(data[[variable]] ~ factor(data[[by]]), conf.level = conf.level, var.equal = TRUE) %>%
  broom::tidy()
}
```

### Function Arguments:

For `tbl_summary()` objects, the custom function will be passed the following arguments: `custom_pvalue_fun(data=)`. While your function may not utilize each of these arguments, these arguments are passed and the function must accept them. We recommend including `...` to future-proof against updates where additional arguments are added.

The following table describes the argument inputs for each gtsummary table type.

argument	tbl_summary	tbl_svysummary	tbl_survfit
data=	A data frame	A survey object	A survfit object
variable=	String variable name	String variable name	NA
by=	String variable name	String variable name	NA
group=	String variable name	NA	NA
type=	Summary type	Summary type	NA
conf.level=	Confidence interval level	NA	NA
adj.vars=	Character vector of adjustment variable names (e.g. used in ANCOVA)	NA	NA

---

theme\_gtsummary

*Available gtsummary themes*

---

## Description

**[Experimental]** 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", "nejm", "lancet"),
  set_theme = TRUE
)

theme_gtsummary_compact(set_theme = TRUE)

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", "ja", "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)

```

**Arguments**

journal	String indicating the journal theme to follow. <ul style="list-style-type: none"> <li>• "jama" Journal of the American Medical Association</li> <li>• "nejm" New England Journal of Medicine</li> <li>• "lancet" The Lancet</li> </ul>
set_theme	Logical indicating whether to set the theme. Default is TRUE. When FALSE the named list of theme elements is returned invisibly
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble" #' @export
language	String indicating language. Must be one of "de" (German), "en" (English), "es" (Spanish), "fr" (French), "gu" (Gujarati), "hi" (Hindi), "ja" (Japanese), "mr" (Marathi), "pt" (Portuguese), "se" (Swedish), "zh-cn" (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 <code>tbl_summary()</code> . If <code>decimal.mark=</code> is NULL, <code>iqr.sep=</code> is <code>" , "</code> . The comma separator, however, can look odd when <code>decimal.mark = " , "</code> . In this case the argument will default to an en dash
ci.sep	string indicating separator for confidence intervals. If <code>decimal.mark=</code> is NULL, <code>ci.sep=</code> is <code>" , "</code> . The comma separator, however, can look odd when <code>decimal.mark = " , "</code> . 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
  - `"nejm"` The New England Journal of Medicine
  - `"lancet"` The Lancet
- `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=)`
  - `"gt"` sets the `gt` package as the default print engine
  - `"flextable"` sets the `flextable` package as the default print engine
  - `"huxtable"` sets the `huxtable` package as the default print engine
  - `"kable"` sets the `knitr::kable()` function as the default print engine
  - `"kable_extra"` sets the `kableExtra` package as the default print engine
  - `"tibble"` returns output as tibble
- `theme_gtsummary_continuous2()`
  - Set all continuous variables to summary type `"continuous2"` by default
  - Use the `statistic=` argument to set the default continuous variable summary statistics
- `theme_gtsummary_mean_sd()`
  - Set default summary statistics to mean and standard deviation in `tbl_summary()`
  - Set default tests in `add_p.tbl_summary()` to t-tests and ANOVA

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) %>%
    add_stat_label() %>%
    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

- \* **datasets**
    - trial, [88](#)
  - \* **gtsummary output types**
    - as\_flex\_table, [27](#)
    - as\_gt, [29](#)
    - as\_hux\_table, [30](#)
    - as\_kable, [31](#)
    - as\_kable\_extra, [32](#)
    - as\_tibble.gtsummary, [33](#)
  - \* **style tools**
    - style\_number, [57](#)
    - style\_percent, [58](#)
    - style\_pvalue, [59](#)
    - style\_ratio, [60](#)
    - style\_sigfig, [61](#)
  - \* **tbl\_cross tools**
    - add\_p.tbl\_cross, [16](#)
    - inline\_text.tbl\_cross, [40](#)
    - tbl\_cross, [62](#)
  - \* **tbl\_regression tools**
    - add\_global\_p, [6](#)
    - add\_nevent.tbl\_regression, [11](#)
    - add\_q, [22](#)
    - bold\_italicize\_labels\_levels, [34](#)
    - combine\_terms, [36](#)
    - inline\_text.tbl\_regression, [41](#)
    - modify, [47](#)
    - tbl\_merge, [63](#)
    - tbl\_regression, [65](#)
    - tbl\_stack, [69](#)
  - \* **tbl\_summary tools**
    - add\_n.tbl\_summary, [8](#)
    - add\_overall, [14](#)
    - add\_p.tbl\_summary, [17](#)
    - add\_q, [22](#)
    - add\_stat\_label, [25](#)
    - bold\_italicize\_labels\_levels, [34](#)
    - inline\_text.tbl\_summary, [42](#)
    - inline\_text.tbl\_survfit, [44](#)
    - modify, [47](#)
    - tbl\_merge, [63](#)
    - tbl\_stack, [69](#)
    - tbl\_summary, [70](#)
  - \* **tbl\_survfit tools**
    - add\_n.tbl\_survfit, [10](#)
    - add\_nevent.tbl\_survfit, [12](#)
    - add\_p.tbl\_survfit, [18](#)
    - modify, [47](#)
    - tbl\_merge, [63](#)
    - tbl\_stack, [69](#)
    - tbl\_survfit, [74](#)
  - \* **tbl\_svsummary tools**
    - add\_n.tbl\_summary, [8](#)
    - add\_overall, [14](#)
    - add\_p.tbl\_svsummary, [20](#)
    - add\_q, [22](#)
    - add\_stat\_label, [25](#)
    - modify, [47](#)
    - tbl\_merge, [63](#)
    - tbl\_stack, [69](#)
    - tbl\_svsummary, [77](#)
  - \* **tbl\_uvregression tools**
    - add\_global\_p, [6](#)
    - add\_nevent.tbl\_uvregression, [13](#)
    - add\_q, [22](#)
    - bold\_italicize\_labels\_levels, [34](#)
    - inline\_text.tbl\_uvregression, [45](#)
    - modify, [47](#)
    - tbl\_merge, [63](#)
    - tbl\_stack, [69](#)
    - tbl\_uvregression, [80](#)
- add\_difference, [3](#)
- add\_glance\_source\_note, [5](#)
- add\_global\_p, [6](#), [12](#), [13](#), [23](#), [35](#), [37](#), [42](#), [46](#),  
[49](#), [64](#), [67](#), [69](#), [70](#), [82](#)
- add\_n, [8](#)
- add\_n.tbl\_summary, [8](#), [15](#), [18](#), [21](#), [22](#), [26](#), [35](#),  
[43](#), [45](#), [48](#), [64](#), [69](#), [73](#), [80](#)
- add\_n.tbl\_summary(), [8](#)
- add\_n.tbl\_survfit, [10](#), [12](#), [19](#), [49](#), [64](#), [70](#), [76](#)
- add\_n.tbl\_survfit(), [8](#)
- add\_n.tbl\_svsummary  
(add\_n.tbl\_summary), [8](#)
- add\_n.tbl\_svsummary(), [8](#)
- add\_nevent, [11](#)

- `add_nevent.tbl_regression`, 7, 11, 11, 23, 35, 37, 42, 49, 64, 67, 69
- `add_nevent.tbl_survfit`, 10, 12, 19, 49, 64, 70, 76
- `add_nevent.tbl_uvregression`, 7, 11, 13, 23, 35, 46, 49, 64, 70, 82
- `add_overall`, 10, 14, 18, 21, 22, 26, 35, 43, 45, 48, 64, 69, 73, 80
- `add_p`, 15, 72
- `add_p.tbl_cross`, 15, 16, 40, 63
- `add_p.tbl_summary`, 10, 15, 17, 22, 26, 35, 43, 45, 48, 64, 69, 73
- `add_p.tbl_survfit`, 10, 12, 15, 18, 49, 64, 70, 76
- `add_p.tbl_svsummary`, 10, 15, 20, 22, 26, 48, 64, 69, 80
- `add_q`, 7, 10, 12, 13, 15, 18, 21, 22, 26, 35, 37, 42, 43, 45, 46, 48, 49, 64, 67, 69, 70, 73, 80, 82
- `add_stat`, 23
- `add_stat_label`, 10, 15, 18, 21, 22, 25, 35, 43, 45, 48, 64, 69, 73, 80
- `all_categorical` (`select_helpers`), 54
- `all_continuous` (`select_helpers`), 54
- `all_continuous2` (`select_helpers`), 54
- `all_contrasts` (`select_helpers`), 54
- `all_dichotomous` (`select_helpers`), 54
- `all_interaction` (`select_helpers`), 54
- `all_intercepts` (`select_helpers`), 54
- `all_stat_cols` (`select_helpers`), 54
- `all_tests` (`select_helpers`), 54
- `as_flex_table`, 27, 29, 31–34
- `as_flex_table()`, 28
- `as_gt`, 28, 29, 31–34
- `as_hux_table`, 28, 29, 30, 32–34
- `as_kable`, 28, 29, 31, 31, 33, 34
- `as_kable_extra`, 28, 29, 31, 32, 32, 34
- `as_tibble.gtsummary`, 28, 29, 31–33, 33
- `bold_italicize_labels_levels`, 7, 10, 12, 13, 15, 18, 22, 23, 26, 34, 37, 42, 43, 45, 46, 48, 49, 64, 67, 69, 70, 73, 82
- `bold_labels` (`bold_italicize_labels_levels`), 34
- `bold_labels()`, 32
- `bold_levels` (`bold_italicize_labels_levels`), 34
- `bold_p`, 35
- `broom::tidy()`, 67
- `car::Anova`, 7
- `combine_terms`, 7, 12, 23, 35, 36, 42, 49, 64, 67, 69
- `custom_tidiers`, 37
- `filter_p` (`sort_filter_p`), 56
- `flextable::add_header_row()`, 28
- `flextable::align()`, 28
- `flextable::autofit()`, 28
- `flextable::bold()`, 28
- `flextable::border()`, 28
- `flextable::flextable()`, 28
- `flextable::fontsize()`, 28
- `flextable::footnote()`, 28
- `flextable::italic()`, 28
- `flextable::padding()`, 28
- `flextable::set_header_labels()`, 28
- `flextable::valign()`, 28
- `flextable::width()`, 28
- `geepack::geeglm`, 11, 13
- `glm`, 80, 81
- `glue::glue`, 9, 41, 43, 46, 72, 78, 81
- `glue::glue()`, 48
- `gt::gt`, 29
- `gt::html()`, 47
- `gt::md()`, 47
- `gtsummary` themes, 56
- `huxtable::add_footnote()`, 30
- `huxtable::align()`, 30
- `huxtable::huxtable()`, 30
- `huxtable::insert_row()`, 30
- `huxtable::set_bold()`, 31
- `huxtable::set_italic()`, 31
- `huxtable::set_left_padding()`, 30
- `huxtable::set_na_string()`, 31
- `inline_text`, 11, 13, 39
- `inline_text.tbl_cross`, 16, 40, 63
- `inline_text.tbl_regression`, 7, 12, 23, 35, 37, 39, 41, 49, 64, 67, 69
- `inline_text.tbl_summary`, 10, 15, 18, 22, 26, 35, 39, 42, 45, 48, 64, 69, 73
- `inline_text.tbl_survfit`, 10, 15, 18, 22, 26, 35, 39, 43, 44, 48, 64, 69, 73
- `inline_text.tbl_svsummary` (`inline_text.tbl_summary`), 42
- `inline_text.tbl_uvregression`, 7, 13, 23, 35, 39, 45, 49, 64, 70, 82
- `italicize_labels` (`bold_italicize_labels_levels`), 34

- `italicize_levels`  
  (`bold_italicize_labels_levels`),  
  34
- `italicize_levels()`, 32
- `knit_print.gtsummary` (`print_gtsummary`),  
  53
- `knitr::kable`, 31–33
- `lm`, 80, 81
- `lme4::glmer`, 11, 13
- `modifications`, 66, 68, 82
- `modify`, 7, 10, 12, 13, 15, 18, 19, 21–23, 26,  
  35, 37, 42, 43, 45, 46, 47, 64, 67, 69,  
  70, 73, 76, 80, 82
- `modify_caption` (`modify`), 47
- `modify_column_hide`, 49
- `modify_column_unhide`  
  (`modify_column_hide`), 49
- `modify_footnote` (`modify`), 47
- `modify_header` (`modify`), 47
- `modify_spanning_header` (`modify`), 47
- `modify_table_body`, 50
- `modify_table_header`, 51
- `pool_and_tidy_mice` (`custom_tidiers`), 37
- `print.gtsummary` (`print_gtsummary`), 53
- `print_gtsummary`, 53
- `remove_row_type`, 53
- `reset_gtsummary_theme`  
  (`set_gtsummary_theme`), 55
- `reset_gtsummary_theme()`, 87
- `select_helpers`, 54
- `set_gtsummary_theme`, 55
- `set_gtsummary_theme()`, 87
- `show_header_names` (`modify`), 47
- `sort_filter_p`, 56
- `sort_p` (`sort_filter_p`), 56
- `stats::anova`, 36
- `stats::anova()`, 36
- `stats::glm`, 11, 13
- `stats::p.adjust`, 22
- `stats::update`, 36
- `style_number`, 57, 59–61
- `style_percent`, 58, 58, 59–61
- `style_percent()`, 75
- `style_pvalue`, 4, 16, 17, 19, 21, 22, 40, 43,  
  44, 58, 59, 59, 60, 61, 66, 81
- `style_ratio`, 44, 58, 59, 60, 61, 66, 81
- `style_sigfig`, 44, 58–60, 61, 66, 75, 81
- `style_sigfig()`, 4
- `survey::svychisq`, 20, 21
- `survey::svydesign()`, 77
- `survey::svyranktest`, 20
- `survey::svyttest`, 20
- `survival::coxph`, 11, 13, 81
- `tbl_cross`, 16, 40, 62
- `tbl_merge`, 7, 10, 12, 13, 15, 18, 19, 21–23,  
  26, 35, 37, 42, 43, 45, 46, 48, 49, 53,  
  63, 67, 69, 70, 73, 76, 80, 82
- `tbl_regression`, 7, 11, 12, 23, 27, 29–33, 35,  
  37, 41, 42, 49, 53, 64, 65, 69, 80
- `tbl_regression.default()`, 67
- `tbl_regression.glmerMod`  
  (`tbl_regression_methods`), 67
- `tbl_regression.glmmadmb`  
  (`tbl_regression_methods`), 67
- `tbl_regression.glmmTMB`  
  (`tbl_regression_methods`), 67
- `tbl_regression.lmerMod`  
  (`tbl_regression_methods`), 67
- `tbl_regression.mipo`  
  (`tbl_regression_methods`), 67
- `tbl_regression.mira`  
  (`tbl_regression_methods`), 67
- `tbl_regression.multinom`  
  (`tbl_regression_methods`), 67
- `tbl_regression.stanreg`  
  (`tbl_regression_methods`), 67
- `tbl_regression.survreg`  
  (`tbl_regression_methods`), 67
- `tbl_regression_methods`, 67
- `tbl_stack`, 7, 10, 12, 13, 15, 18, 19, 21–23,  
  26, 35, 37, 42, 43, 45, 46, 48, 49, 53,  
  64, 67, 69, 73, 76, 80, 82
- `tbl_summary`, 9, 10, 14, 15, 17, 18, 22, 26, 27,  
  29–33, 35, 43, 45, 48, 53, 64, 69, 70
- `tbl_summary()`, 77, 78
- `tbl_survfit`, 10, 12, 19, 44, 49, 64, 70, 74
- `tbl_survfit.data.frame()`, 75
- `tbl_survfit.list()`, 75
- `tbl_survfit.survfit()`, 75
- `tbl_svysummary`, 9, 10, 14, 15, 20–22, 26, 48,  
  64, 69, 77
- `tbl_uvregression`, 7, 11, 13, 23, 35, 46, 49,  
  53, 64, 70, 80
- `tests`, 4, 17, 83
- `theme_gtsummary`, 85
- `theme_gtsummary_compact`  
  (`theme_gtsummary`), 85
- `theme_gtsummary_continuous2`  
  (`theme_gtsummary`), 85

theme\_gtsummary\_journal  
    (theme\_gtsummary), 85  
theme\_gtsummary\_language  
    (theme\_gtsummary), 85  
theme\_gtsummary\_mean\_sd  
    (theme\_gtsummary), 85  
theme\_gtsummary\_printer  
    (theme\_gtsummary), 85  
tibble, 34  
tidy\_bootstrap(custom\_tidiers), 37  
tidy\_standardize(custom\_tidiers), 37  
trial, 88  
vetted model, 66, 68, 81