

Package ‘gtsummary’

July 13, 2021

Title Presentation-Ready Data Summary and Analytic Result
Tables

Version 1.4.2

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.danieldsjoberg.com/gtsummary/>

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

Depends R (>= 3.4)

Imports broom (>= 0.7.8),
broom.helpers (>= 1.3.0),
cli (>= 2.3.0),
dplyr (>= 1.0.3),
forcats (>= 0.5.0),
glue (>= 1.4.1),
gt (>= 0.3.0),
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)

Suggests broom.mixed (>= 0.2.7),
car,
covr,
effectsize,
flextable (>= 0.5.10),

geepack,
 GGally (>= 2.1.0),
 Hmisc,
 huxtable (>= 5.0.0),
 kableExtra,
 lme4,
 mgcv,
 mice,
 nnet,
 officer,
 parameters,
 parsnip,
 rmarkdown,
 scales,
 spelling (>= 2.2),
 survey,
 testthat,
 workflows

VignetteBuilder knitr

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Config/testthat/edition 3

Config/testthat/parallel true

R topics documented:

add_difference	4
add_glance	5
add_global_p	7
add_n	9
add_n.tbl_summary	10
add_n.tbl_survfit	11
add_nevent	12
add_nevent.tbl_survfit	13
add_nevent_regression	13
add_n_regression	14
add_overall	15
add_p	16
add_p.tbl_cross	17
add_p.tbl_summary	18
add_p.tbl_survfit	20
add_p.tbl_svsummary	21
add_q	23
add_significance_stars	24
add_stat	26

add_stat_label	28
add_vif	30
as_flex_table	31
as_gt	33
as_hux_table	34
as_kable	35
as_kable_extra	36
as_tibble.gtsummary	37
bold_italicize_labels_levels	38
bold_p	39
combine_terms	40
custom_tidiers	41
inline_text	43
inline_text.gtsummary	44
inline_text.tbl_cross	45
inline_text.tbl_regression	46
inline_text.tbl_summary	47
inline_text.tbl_survfit	49
inline_text.tbl_uvregression	50
modify	52
modify_column_hide	55
modify_fmt_fun	55
modify_table_body	56
modify_table_styling	58
plot	59
print_gtsummary	60
remove_row_type	61
select_helpers	62
set_gtsummary_theme	63
sort_filter_p	64
style_number	65
style_percent	66
style_pvalue	67
style_ratio	68
style_sigfig	69
tbl_cross	70
tbl_merge	71
tbl_regression	73
tbl_regression_methods	75
tbl_stack	77
tbl_strata	79
tbl_summary	80
tbl_survfit	84
tbl_survfit_errors	87
tbl_svysummary	87
tbl_uvregression	91
tests	94
theme_gtsummary	96
trial	99

add_difference *Add difference between groups*

Description

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

Usage

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

Arguments

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

Example Output

Examples

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

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

 add_glance

Add Model Statistics

Description

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

Usage

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

```

add_glance_source_note(
  x,
  include = everything(),
  label = NULL,
  fmt_fun = NULL,
  glance_fun = broom::glance,
  text_interpret = c("md", "html"),
  sep1 = " = ",
  sep2 = "; "
)

```

Arguments

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

Value

gtsummary table

Default Labels

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

Statistic Name	Default Label
r.squared	R ²
adj.r.squared	Adjusted R ²
p.value	p-value
logLik	Log-likelihood
statistic	Statistic
df.residual	Residual df
null.deviance	Null deviance
df.null	Null df
nevent	N events

concordance	c-index
std.error.concordance	c-index SE
nobs	No. Obs.
deviance	Deviance
sigma	Sigma

Tips

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

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

Example Output

Examples

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

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

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

add_global_p

Add the global p-values

Description

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

Usage

```

add_global_p(x, ...)

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

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

```

Arguments

x	Object with class <code>tbl_regression</code> from the tbl_regression function
...	Additional arguments to be passed to <code>car::Anova</code>
include	Variables to calculate global p-value for. Input may be a vector of quoted or unquoted variable names. Default is <code>everything()</code>
type	Type argument passed to <code>car::Anova</code> . Default is "III"
keep	Logical argument indicating whether to also retain the individual p-values in the table output for each level of the categorical variable. Default is 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_q\(\)](#), [bold_italicize_labels_levels](#), [inline_text.tbl_uvregression\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_stack\(\)](#), [tbl_uvregression\(\)](#)

Other `tbl_regression` tools: [add_q\(\)](#), [bold_italicize_labels_levels](#), [combine_terms\(\)](#), [inline_text.tbl_regression](#), [modify](#), [tbl_merge\(\)](#), [tbl_regression\(\)](#), [tbl_stack\(\)](#)

Examples

```
# Example 1 -----
if (requireNamespace("car")) {
  tbl_lm_global_ex1 <-
    lm(marker ~ age + grade, trial) %>%
    tbl_regression() %>%
    add_global_p()
}

# Example 2 -----
if (requireNamespace("car")) {
  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_regression](#),
[add_n.tbl_uvregression](#)

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

<code>x</code>	Object with class <code>tbl_summary</code> from the tbl_summary function or with class <code>tbl_svysummary</code> from the tbl_svysummary function
<code>statistic</code>	String indicating the statistic to report. Default is the number of non-missing observation for each variable, <code>statistic = "{n}"</code> . Other statistics available to report include: <ul style="list-style-type: none"> • <code>"{N}"</code> total number of observations, • <code>"{n}"</code> number of non-missing observations, • <code>"{n_miss}"</code> number of missing observations, • <code>"{p}"</code> percent non-missing data, • <code>"{p_miss}"</code> percent missing data The argument uses glue::glue syntax and multiple statistics may be reported, e.g. <code>statistic = "{n} / {N} ({p}%)"</code>
<code>col_label</code>	String indicating the column label. Default is <code>"**N**"</code>
<code>footnote</code>	Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is <code>FALSE</code>

last	Logical indicator to include N column last in table. Default is FALSE, which will display N column first.
missing	DEPRECATED. Logical argument indicating whether to print N (missing = FALSE), or N missing (missing = TRUE). Default is FALSE
...	Not used

Value

A tbl_summary or tbl_svsummary 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_svsummary tools: [add_overall\(\)](#), [add_p.tbl_svsummary\(\)](#), [add_q\(\)](#), [add_stat_label\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_stack\(\)](#), [tbl_svsummary\(\)](#)

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

Add number of events to a regression table

Description

Adds a column of the number of events to tables created with [tbl_regression](#) or [tbl_uvregression](#). Supported model types are among 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](#), [add_nevent.tbl_survfit](#)

 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

<code>x</code>	object of class 'tbl_survfit'
<code>...</code>	Not used

Example Output

See Also

Other `tbl_survfit` tools: [add_n.tbl_survfit\(\)](#), [add_p.tbl_survfit\(\)](#), [modify.tbl_survfit\(\)](#), [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_regression *Add event N to regression table*

Description

Add event N to regression table

Usage

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

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

Arguments

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

Example Output**Examples**

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

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

Description

Add N to regression table

Usage

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

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

Arguments

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

Example Output**Examples**

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

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

add_overall

Add column with overall summary statistics

Description

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

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_svsummary'
add_overall(x, last = FALSE, col_label = NULL)
```

Arguments

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

Value

A `tbl_summary` object or a `tbl_svsummary` 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_svsummary` tools: [add_n.tbl_summary\(\)](#), [add_p.tbl_svsummary\(\)](#), [add_q\(\)](#), [add_stat_label\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_stack\(\)](#), [tbl_svsummary\(\)](#)

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 tbl_cross function
test	A string specifying statistical test to perform. Default is <code>"chisq.test"</code> when expected cell counts ≥ 5 and <code>"fisher.test"</code> when expected cell counts < 5 .
pvalue_fun	Function to round and format p-value. Default is style_pvalue , 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 <code>{gt}</code> table source notes rather than a column.
...	Not used

Example Output**Author(s)**

Karissa Whiting

See AlsoOther `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	<i>Adds p-values to summary tables</i>
-------------------	--

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 <code>tbl_summary</code> 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 tests 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 ≥ 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 <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code>).

group	Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is NULL. See tests for methods that utilize the group= argument.
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 everything().
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests("t.test") ~ list(var.equal = TRUE)
exclude	DEPRECATED
...	Not used

Value

A tbl_summary object

Example Output**Author(s)**

Daniel D. Sjoberg, Emily C. Zabor

See Also

See [tbl_summary vignette](#) for detailed examples

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)
  )
```

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

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

x	Object of class "tbl_survfit"
test	string indicating test to use. Must be one of "logrank", "survdiff", "petopeto_gehanwilcoxon", "coxph_lrt", "coxph_wald", "coxph_score". See details below
test.args	List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests("t.test") ~ list(var.equal = TRUE)
pvalue_fun	Function to round and format p-values. Default is style_pvalue . 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. pvalue_fun = function(x) style_pvalue(x, digits = 2) or equivalently, purrr::partial(style_pvalue, digits = 2)).
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 everything().
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Not used

test argument

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

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

Example Output**See Also**

Other `tbl_survfit` tools: [add_n.tbl_survfit\(\)](#), [add_nevent.tbl_survfit\(\)](#), [modify.tbl_merge\(\)](#), [tbl_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 `survdiff()`
add_p_tbl_survfit_ex2 <-
  gts_survfit %>%
  add_p(test = "survdiff", test.args = list(rho = 0.5))
```

`add_p.tbl_svsummary` *Adds p-values to svsummary tables*

Description

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

Usage

```
## S3 method for class 'tbl_svsummary'
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 <code>tbl_svysummary</code> 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"> "svy.t.test" for a t-test adapted to complex survey samples (cf. <code>survey::svyttest</code>), "svy.wilcox.test" for a Wilcoxon rank-sum test for complex survey samples (cf. <code>survey::svyranktest</code>), "svy.kruskal.test" for a Kruskal-Wallis rank-sum test for complex survey samples (cf. <code>survey::svyranktest</code>), "svy.vanderwaerden.test" for a van der Waerden's normal-scores test for complex survey samples (cf. <code>survey::svyranktest</code>), "svy.median.test" for a Mood's test for the median for complex survey samples (cf. <code>survey::svyranktest</code>), "svy.chisq.test" for a Chi-squared test with Rao & Scott's second-order correction (cf. <code>survey::svychisq</code>), "svy.adj.chisq.test" for a Chi-squared test adjusted by a design effect estimate (cf. <code>survey::svychisq</code>), "svy.wald.test" for a Wald test of independence for complex survey samples (cf. <code>survey::svychisq</code>), "svy.adj.wald.test" for an adjusted Wald test of independence for complex survey samples (cf. <code>survey::svychisq</code>), "svy.lincom.test" for a test of independence using the exact asymptotic distribution for complex survey samples (cf. <code>survey::svychisq</code>), "svy.saddlepoint.test" for a test of independence using a saddlepoint approximation for complex survey samples (cf. <code>survey::svychisq</code>), Tests default to "svy.wilcox.test" for continuous variables and "svy.chisq.test" for categorical variables.
<code>pvalue_fun</code>	Function to round and format p-values. Default is <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code>).
<code>include</code>	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or <code>tidyselect</code> 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>...</code>	Not used

Value

A `tbl_svysummary` object

Example Output**Author(s)**

Joseph Larmarange

See Also

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

Examples

```
# Example 1 -----
# A simple weighted dataset
add_p_svsummary_ex1 <-
  survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) %>%
  tbl_svsummary(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_svsummary_ex2 <-
  tbl_svsummary(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_svsummary_ex3 <-
  tbl_svsummary(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

*Add a column of q-values to account for multiple comparisons***Description**

Adjustments to p-values are performed with `stats::p.adjust`.

Usage

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

Arguments

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

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()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_stack()`

Other `tbl_uvregression` tools: `add_global_p()`, `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 -----
if (requireNamespace("car")) {
  add_q_ex2 <-
    trial[c("trt", "age", "grade", "response")] %>%
    tbl_uvregression(
      y = response,
      method = glm,
      method.args = list(family = binomial),
      exponentiate = TRUE
    ) %>%
    add_global_p() %>%
    add_q()
}
```

add_significance_stars

Add significance stars

Description

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

Usage

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

Arguments

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

Future Updates

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

Example Output**Examples**

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

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

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

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

add_stat

Add a custom statistic column

Description

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

Usage

```
add_stat(
  x,
  fns,
  location = NULL,
  fmt_fun = NULL,
  header = NULL,
  footnote = NULL,
  new_col_name = NULL
)
```

Arguments

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

Details

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

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

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

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

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

Example Output

Examples

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

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

```

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

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

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

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

```

add_stat_label

Add statistic labels

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 tbl_summary function or with class <code>tbl_svsummary</code> from the tbl_svsummary 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_svsummary` object

Tips

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

For example, the following two tables merge properly

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

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

Example Output**Author(s)**

Daniel D. Sjoberg

See Also

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_svsummary` tools: [add_n.tbl_summary\(\)](#), [add_overall\(\)](#), [add_p.tbl_svsummary\(\)](#), [add_q\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_stack\(\)](#), [tbl_svsummary\(\)](#)

Examples

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

# Example 1 -----
```

```

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

# Example 2 -----
add_stat_label_ex2 <-
  tbl %>%
  add_stat_label(
    # add a new column with statistic labels
    location = "column"
  )

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

```

add_vif

Add Variance Inflation Factor

Description

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

Usage

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

Arguments

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

Example Output

Examples

```
# Example 1 -----
if (requireNamespace("car")) {
  add_vif_ex1 <-
    lm(age ~ grade + marker, trial) %>%
    tbl_regression() %>%
    add_vif()
}

# Example 2 -----
if (requireNamespace("car")) {
  add_vif_ex2 <-
    lm(age ~ grade + marker, trial) %>%
    tbl_regression() %>%
    add_vif(c("aGVIF", "df"))
}
```

as_flex_table

*Convert gtsummary object to a flextable object***Description**

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

Usage

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

Arguments

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

Value

A flextable object

Details

The `as_flex_table()` functions converts the `gtsummary` object to a flextable, and prints it with the following styling functions.

1. `flextable::flextable()`
2. `flextable::set_header_labels()` to set column labels
3. `flextable::add_header_row()`, if applicable, to set spanning column header
4. `flextable::align()` to set column alignment
5. `flextable::padding()` to indent variable levels
6. `flextable::fontsize()` to set font size
7. `flextable::autofit()` to estimate the column widths
8. `flextable::footnote()` to add table footnotes and source notes
9. `flextable::bold()` to bold cells in data frame
10. `flextable::italic()` to italicize cells in data frame
11. `flextable::border()` to set all border widths to 1
12. `flextable::padding()` to set consistent header padding
13. `flextable::valign()` to ensure label column is top-left justified

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

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

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other `gtsummary` output types: [as_gt\(\)](#), [as_hux_table\(\)](#), [as_kable_extra\(\)](#), [as_kable\(\)](#), [as_tibble.gtsummary\(\)](#)

Examples

```
if (requireNamespace("flextable"))
  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. tbl_summary or tbl_regression)
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 gt::gt
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 = FALSE
)
```

Arguments

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

Value

A huxtable object

Details

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

1. huxtable::huxtable()
2. huxtable::insert_row() to insert header rows
3. huxtable::set_left_padding() to indent variable levels
4. huxtable::add_footnote() to add table footnotes and source notes
5. huxtable::set_bold() to bold cells

6. `huxtable::set_italic()` to italicize cells
7. `huxtable::set_top_border()` add horizontal line (when indicated)
8. `huxtable::set_na_string()` to use an em-dash for missing numbers
9. `huxtable::set_markdown()` use markdown for header rows
10. `huxtable::set_align()` to set column alignment

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

Author(s)

David Hugh-Jones

See Also

Other gtsummary output types: [as_flex_table\(\)](#), [as_gt\(\)](#), [as_kable_extra\(\)](#), [as_kable\(\)](#), [as_tibble.gtsummary\(\)](#)

Examples

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

as_kable

Convert gtsummary object to a kable object

Description

Function converts a gtsummary object to a knitr_kable object. This function 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. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. <code>tidyselect</code> and <code>gtsummary</code> select helper functions are also accepted. Default is <code>everything()</code> .
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED
...	Additional arguments passed to knitr::kable

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. Sjöberg

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`. Bold and italic cells are not supported for `kableExtra` output via `gtsummary`.

Usage

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

Arguments

<code>x</code>	Object created by a function from the <code>gtsummary</code> package (e.g. <code>tbl_summary</code> or <code>tbl_regression</code>)
<code>include</code>	Commands to include in output. Input may be a vector of quoted or unquoted names. <code>tidyselect</code> and <code>gtsummary</code> select helper functions are also accepted. Default is <code>everything()</code> .
<code>return_calls</code>	Logical. Default is <code>FALSE</code> . If <code>TRUE</code> , 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.

... Additional arguments passed to [knitr:kable](#)

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

```
if (requireNamespace("kableExtra"))
  tbl <-
    trial %>%
    tbl_summary(by = trt) %>%
    as_kable_extra()
```

as_tibble.gtsummary *Convert gtsummary object to a tibble*

Description

Function converts a gtsummary object to a tibble.

Usage

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

Arguments

x Object created by a function from the gtsummary package (e.g. [tbl_summary](#) or [tbl_regression](#))

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.

return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED
...	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_q\(\)](#), [combine_terms\(\)](#), [inline_text.tbl_regression\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_regression\(\)](#), [tbl_stack\(\)](#)

Other tbl_uvregression tools: [add_global_p\(\)](#), [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()
```

bold_p

Bold significant p-values or q-values

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_q\(\)](#), [bold_italicize_labels_levels](#), [inline_text.tbl_regression_modify](#), [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)

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

```

Arguments

<code>x</code>	a regression model object
<code>exponentiate</code>	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
<code>conf.level</code>	The confidence level to use for the confidence interval if <code>conf.int = TRUE</code> . Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
<code>conf.int</code>	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
<code>...</code>	arguments passed to method; <ul style="list-style-type: none"> • <code>pool_and_tidy_mice()</code>: <code>mice::tidy(x, ...)</code> • <code>tidy_standardize()</code>: <code>effectsize::standardize_parameters(x, ...)</code> • <code>tidy_bootstrap()</code>: <code>parameters::bootstrap_parameters(x, ...)</code>
<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)

if (requireNamespace("effectsize")) {
  tidy_standardize_ex1 <-
    tbl_merge(
      list(tbl_stnd, tbl),
      tab_spanner = c("**Standardized Model**", "**Original Model**")
    )
}

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

# Example 3 -----
# Multiple Imputation using the mice package
set.seed(1123)
if (requireNamespace("mice")) {
  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_svsummary](#), [inline_text.tbl_regression](#), [inline_text.tbl_uvregression](#), [inline_text.tbl_survfit](#), [inline_text.tbl_cross](#), [inline_text.gtsummary](#)

inline_text.gtsummary *Report statistics from summary tables inline*

Description

Report statistics from summary tables inline

Usage

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

Arguments

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

column + pattern

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

inline_text.tbl_cross *Report statistics from cross table inline*

Description

[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 <code>style_pvalue</code> . The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. <code>pvalue_fun = function(x) style_pvalue(x, digits = 2)</code> or equivalently, <code>purrr::partial(style_pvalue, digits = 2)</code>).
<code>...</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 tbl_regression
<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 glue::glue formatting. Default is <code>"{estimate} ({conf.level }% CI {conf.low}, {conf.high}; {p.value})"</code> . All columns from <code>x\$table_body</code> are available to print as well as the confidence level (<code>conf.level</code>). See below for details.
<code>estimate_fun</code>	function to style model coefficient estimates. Columns <code>'estimate'</code> , <code>'conf.low'</code> , and <code>'conf.high'</code> are formatted. Default is <code>x\$inputs\$estimate_fun</code>
<code>pvalue_fun</code>	function to style p-values and/or q-values. Default is <code>function(x) style_pvalue(x, prepend_p = TRUE)</code>
<code>...</code>	Not used

Value

A string reporting results from a `gtsummary` table

pattern argument

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

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

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

Author(s)

Daniel D. Sjoberg

See Also

Other `tbl_regression` tools: [add_global_p\(\)](#), [add_q\(\)](#), [bold_italicize_labels_levels](#), [combine_terms\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_regression\(\)](#), [tbl_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_svsummary'
inline_text(
  x,
  variable,
  column = NULL,
```

```

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

```

Arguments

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

Value

A string reporting results from a `gtsummary` table

Author(s)

Daniel D. Sjöberg

See Also

Other `tbl_summary` tools: `add_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

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

Value

A string reporting results from a gtsummary table

pattern argument

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

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

Author(s)

Daniel D. Sjoberg

See Also

Other `tbl_uvregression` tools: [add_global_p\(\)](#), [add_q\(\)](#), [bold_italicize_labels_levels](#), [modify](#), [tbl_merge\(\)](#), [tbl_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

Modify column headers, footnotes, spanning headers, and table captions

Description

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

Usage

```

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

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

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

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

show_header_names(x = NULL, 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".
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	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.
stat_by	DEPRECATED, use <code>update = all_stat_cols() ~ "<label>"</code> instead.
abbreviation	Logical indicating if an abbreviation is being updated.
caption	a string of the table caption/title

Value

Updated gtsummary object

tbl_summary(), tbl_svysummary(), and tbl_cross()

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

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

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

tbl_regression()

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

captions

Captions are assigned based on output type.

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

Example Output**Author(s)**

Daniel D. Sjoberg

See Also

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

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

Other `tbl_uvregression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `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, columns)

modify_column_unhide(x, columns)
```

Arguments

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

Example Output

See Also

Other Advanced modifiers: [modify_fmt_fun\(\)](#), [modify_table_body\(\)](#), [modify_table_styling\(\)](#)

Examples

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

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

Description

[Experimental] Use this function to update the way numeric columns and rows of .\$table_body are formatted

Usage

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

Arguments

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

Example Output**rows argument**

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

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

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

See Also

Other Advanced modifiers: [modify_column_hide\(\)](#), [modify_table_body\(\)](#), [modify_table_styling\(\)](#)

Examples

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

modify_table_body

Modify Table Body

Description

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

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

Usage

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

Arguments

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

Example Output**See Also**

```
modify_table_styling()
```

See [gtsummary internals vignette](#)

Other Advanced modifiers: [modify_column_hide\(\)](#), [modify_fmt_fun\(\)](#), [modify_table_styling\(\)](#)

Examples

```
# Example 1 -----
# Add number of cases and controls to regression table
modify_table_body_ex1 <-
  trial %>%
  select(response, age, marker) %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
    exponentiate = TRUE,
    hide_n = TRUE
  ) %>%
  # adding number of non-events to table
  modify_table_body(
    ~ .x %>%
      dplyr::mutate(N_nonevent = N_obs - N_event) %>%
      dplyr::relocate(c(N_event, N_nonevent), .before = estimate)
  ) %>%
  # assigning header labels
  modify_header(N_nonevent = "***Control N**", N_event = "***Case N**") %>%
  modify_fmt_fun(c(N_event, N_nonevent) ~ style_number)
```

 modify_table_styling *Modify Table Styling*

Description

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

Usage

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

Arguments

<code>x</code>	gtsummary object
<code>columns</code>	vector or selector of columns in <code>x\$table_body</code>
<code>rows</code>	predicate expression to select rows in <code>x\$table_body</code> . Can be used to style footnote, formatting functions, missing symbols, and text formatting. Default is <code>NULL</code> . See details below.
<code>label</code>	string of column label(s)
<code>spanning_header</code>	string with text for spanning header
<code>hide</code>	logical indicating whether to hide column from output
<code>footnote</code>	string with text for footnote
<code>footnote_abbrev</code>	string with abbreviation definition, e.g. "CI = Confidence Interval"
<code>align</code>	string indicating alignment of column, must be one of <code>c("left", "right", "center")</code>
<code>missing_symbol</code>	string indicating how missing values are formatted.
<code>fmt_fun</code>	function that formats the statistics in the columns/rows in <code>columns=</code> and <code>rows=</code>
<code>text_format</code>	string indicated which type of text formatting to apply to the rows and columns. Must be one of <code>c("bold", "italic", "indent", "indent2")</code> . Do not assign both "indent" and "indent2" to the same cell.

`undo_text_format` rarely used. Logical that undoes the indent, bold, and italic styling when TRUE

`text_interpret` string, must be one of "md" or "html"

`cols_merge_pattern` glue-syntax string indicating how to merge columns in `x$table_body`. For example, to construct a confidence interval use "{conf.low},{conf.high}". The first column listed in the pattern string must match the single column name passed in `columns=`.

Details

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

rows argument

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

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

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

cols_merge_pattern argument

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

See Also

`modify_table_body()`

See [gtsummary internals vignette](#)

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

Description

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

Usage

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

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

Arguments

```
x                'tbl_regression' or 'tbl_uvregression' object
remove_header_rows
                  logical indicating whether to remove header rows for categorical variables. Default is TRUE
remove_reference_rows
                  logical indicating whether to remove reference rows for categorical variables. Default is FALSE.
...              arguments passed to GGally::ggcoef_plot(...)
```

Details**[Experimental]****Value**

a ggplot

Examples

```
if (requireNamespace("GGally")) {
  glm(response ~ marker + grade, trial, family = binomial) %>%
    tbl_regression(
      add_estimate_to_reference_rows = TRUE,
      exponentiate = TRUE
    ) %>%
    plot()
}
```

print_gtsummary

*print and knit_print methods for gtsummary objects***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

<code>continuous2</code>	Logical indicating whether to include continuous2 variables. Default is TRUE
<code>dichotomous</code>	Logical indicating whether to include dichotomous variables. Default is TRUE
<code>tests</code>	string indicating the test type of the variables to select, e.g. select all variables being compared with "t.test"
<code>stat_0</code>	When FALSE, will not select the "stat_0" column. Default is TRUE
<code>contrasts_type</code>	type of contrast to select. When NULL, all variables with a contrast will be selected. Default is NULL. Select among contrast types <code>c("treatment", "sum", "poly", "helmert", "o</code>

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

Description

Style numbers

Usage

```

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

```

Arguments

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

Value

formatted character vector

See Also

Other style tools: [style_percent\(\)](#), [style_pvalue\(\)](#), [style_ratio\(\)](#), [style_sigfig\(\)](#)

Examples

```

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

```

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

Description

Style percentages

Usage

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

Arguments

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

Value

A character vector of styled percentages

Author(s)

Daniel D. Sjoberg

See Also

See Table Gallery [vignette](#) for example

Other style tools: [style_number\(\)](#), [style_pvalue\(\)](#), [style_ratio\(\)](#), [style_sigfig\(\)](#)

Examples

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

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

Description

Style p-values

Usage

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

Arguments

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

Value

A character vector of styled p-values

Author(s)

Daniel D. Sjoberg

See Also

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

Other style tools: [style_number\(\)](#), [style_percent\(\)](#), [style_ratio\(\)](#), [style_sigfig\(\)](#)

Examples

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

 style_ratio

Style significant figure-like rounding for ratios

Description

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

Usage

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

Arguments

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

Value

A character vector of styled ratios

Author(s)

Daniel D. Sjoberg

See Also

Other style tools: [style_number\(\)](#), [style_percent\(\)](#), [style_pvalue\(\)](#), [style_sigfig\(\)](#)

Examples

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

`style_sigfig`*Style significant figure-like rounding*

Description

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

Usage

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

Arguments

<code>x</code>	Numeric vector
<code>digits</code>	Integer specifying the minimum number of significant digits to display
<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>

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**

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 the rows of cross table.
col	A column name in data= to be used for the columns of cross table.
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute (<code>attr(data\$age, "label")</code>) is used. If attribute label is NULL, the variable name will be used.
statistic	A string with the statistic name in curly brackets to be replaced with the numeric statistic (see <code>glue::glue</code>). The default is <code>{n}</code> . If percent argument is "column", "row", or "cell", default is <code>{n} ({p}%)</code> .
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

Merge two or more gtsummary objects

Description

Merges two or more `tbl_regression`, `tbl_uvregression`, `tbl_stack`, `tbl_summary`, or `tbl_svsummary` 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 <code>gt::md</code> . 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_q\(\)](#), [bold_italicize_labels_levels](#), [combine_terms\(\)](#), [inline_text.tbl_regression\(\)](#), [modify.tbl_regression\(\)](#), [tbl_stack\(\)](#)

Other `tbl_uvregression` tools: [add_global_p\(\)](#), [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 = NULL,
  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 <code>tidyselect</code> select helper functions. Default is <code>everything()</code> .
show_single_row	By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here—quoted and unquoted variable name accepted.

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

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

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 examples

Other `tbl_regression` tools: `add_global_p()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `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 'model_fit'
tbl_regression(x, ...)

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

## 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 'brmsfit'
tbl_regression(
  x,
  tidy_fun = function(x, ...) broom.mixed::tidy(x, ..., effects = "fixed"),
  ...
)

```

```
)

## S3 method for class 'gam'
tbl_regression(x, tidy_fun = tidy_gam, ...)

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

Arguments

x	Regression model object
...	arguments passed to <code>tbl_regression.default()</code>
tidy_fun	Option to specify a particular tidier function for the model. Default is to use <code>broom::tidy</code> , but if an error occurs then tidying of the model is attempted with <code>parameters::model_parameters()</code> , if installed.

Methods

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

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

tbl_stack	<i>Stacks two or more gtsummary objects</i>
-----------	---

Description

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

Usage

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

Arguments

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

Value

A tbl_stack object

Example Output**Author(s)**

Daniel D. Sjoberg

See Also

[tbl_merge](#)

Other tbl_summary tools: [add_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_q\(\)](#), [bold_italicize_labels_levels](#), [combine_terms\(\)](#), [inline_text.tbl_regression\(\)](#), [modify](#), [tbl_merge\(\)](#), [tbl_regression\(\)](#)

Other tbl_uvregression tools: [add_global_p\(\)](#), [add_q\(\)](#), [bold_italicize_labels_levels](#), [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"))
```

tbl_strata

*Stratified gtsummary tables***Description**

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

Usage

```
tbl_strata(
  data,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",
  .combine_with = c("tbl_merge", "tbl_stack")
)
```

Arguments

data a data frame or survey object

strata character vector or tidy-selector of columns in data to stratify results by

.tbl_fun	A function or formula. If a <i>function</i> , it is used as is. If a formula, e.g. <code>~ .x %>% tbl_summary() %>% add_p()</code> , it is converted to a function. The stratified data frame is passed to this function.
...	Additional arguments passed on to the <code>.tbl_fun</code> function.
.sep	when more than one stratifying variable is passed, this string is used to separate the levels in the spanning header. Default is <code>" , "</code>
.combine_with	One of <code>c("tbl_merge", "tbl_stack")</code> . Names the function used to combine the stratified tables.

Tips

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

Example Output

Author(s)

Daniel D. Sjoberg

Examples

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

tbl_summary

Create a table of summary statistics

Description

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

Usage

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

Arguments

data	A data frame
by	A column name (quoted or unquoted) in data. Summary statistics will be calculated separately for each level of the by variable (e.g. by = trt). If NULL, summary statistics are calculated using all observations. To stratify a table by two or more variables, use <code>tbl_strata()</code>
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute (<code>attr(data\$age, "label")</code>) is used. If attribute label is NULL, the variable name will be used.
statistic	List of formulas specifying types of summary statistics to display for each variable. The default is <code>list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)")</code> . See below for details.
digits	List of formulas specifying the number of decimal places to round 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>
type	List of formulas specifying variable types. Accepted values are <code>c("continuous", "continuous2", ...)</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>
group	DEPRECATED. Migrated to add_p

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

x	a survfit object, list of survfit objects, or a data frame. If a data frame is passed, a list of survfit objects is constructed using each variable as a stratifying variable.
...	For <code>tbl_survfit.data.frame()</code> and <code>tbl_survfit.survfit()</code> the arguments are passed to <code>tbl_survfit.list()</code> . They are not used when <code>tbl_survfit.list()</code> is called directly.
times	numeric vector of times for which to return survival probabilities.
probs	numeric vector of probabilities with values in (0,1) specifying the survival quantiles to return
statistic	string defining the statistics to present in the table. Default is "{estimate} ({conf.low},{conf.high})"
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age, yrs", stage ~ "Path T Stage")</code> , or a string for a single variable table.
label_header	string specifying column labels above statistics. Default is "{prob} Percentile" for survival percentiles, and "Time {time}" for n-year survival estimates
estimate_fun	function to format the Kaplan-Meier estimates. Default is <code>style_percent()</code> for survival probabilities and <code>style_sigfig</code> for survival times
missing	text to fill when estimate is not estimable. Default is "--"
conf.level	Confidence level for confidence intervals. Default is 0.95
reverse	Flip the probability reported, i.e. 1 -estimate. Default is FALSE. Does not apply to survival quantile requests
quiet	Logical indicating whether to print messages in console. Default is FALSE
y	outcome call, e.g. <code>y = Surv(ttdeath, death)</code>
include	Variable to include as stratifying variables.

Example Output

Author(s)

Daniel D. Sjöberg

See Also

Other `tbl_survfit` tools: [add_n.tbl_survfit\(\)](#), [add_nevent.tbl_survfit\(\)](#), [add_p.tbl_survfit\(\)](#), [modify.tbl_survfit\(\)](#), [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)
set.seed(1123)
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)
trial2 <- trial %>%
  mutate(
    death_cr = case_when(
      death == 0 ~ "censor",
      runif(n()) < 0.5 ~ "death from cancer",
      TRUE ~ "death other causes"
    ) %>% factor()
  )

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

tbl_survfit_errors *Common Sources of Error with tbl_survfit()*

Description

When functions `add_n()` and `add_p()` are run after `tbl_survfit()`, the original call to `survival::survfit()` is extracted and the `formula=` and `data=` arguments are used to calculate the N or p-value.

When the values of the `formula=` and `data=` are unavailable, the functions cannot execute. Below are some tips to modify your code to ensure all functions run without issue.

1. Let `tbl_survfit()` construct the `survival::survfit()` for you by passing a data frame to `tbl_survfit()`. The `survfit` model will be constructed in a manner ensuring the `formula` and `data` are available. This only works if you have a stratified model.

Instead of the following line

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

Use this code

```
trial %>%
  select(ttdeath, death, trt) %>%
  tbl_survfit(y = Surv(ttdeath, death), times = c(12, 24))
```

2. Construct an expression of the `survival::survfit()` before evaluating it. Ensure the `formula` and `data` are available in the call by using the tidyverse bang-bang operator, `!!`.

Use this code

```
formula_arg <- Surv(ttdeath, death) ~ 1
data_arg <- trial
rlang::expr(survfit(!!formula_arg, !!data_arg)) %>%
  eval() %>%
  tbl_survfit(times = c(12, 24))
```

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

Description

The `tbl_svsummary` 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_svsummary(
  data,
  by = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
```

```

value = NULL,
missing = NULL,
missing_text = NULL,
sort = NULL,
percent = NULL,
include = everything()
)

```

Arguments

data	A survey object created with created with <code>survey::svydesign()</code>
by	A column name (quoted or unquoted) in data. Summary statistics will be calculated separately for each level of the by variable (e.g. <code>by = trt</code>). If NULL, summary statistics are calculated using all observations. To stratify a table by two or more variables, use <code>tbl_strata()</code>
label	List of formulas specifying variables labels, e.g. <code>list(age ~ "Age", stage ~ "Path T Stage")</code> . If a variable's label is not specified here, the label attribute (<code>attr(data\$age, "label")</code>) is used. If attribute label is NULL, the variable name will be used.
statistic	List of formulas specifying types of summary statistics to display for each variable. The default is <code>list(all_continuous() ~ "{median} ({p25},{p75})", all_categorical() ~ "{n} ({p}%")</code> . See below for details.
digits	List of formulas specifying the number of decimal places to round 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>
type	List of formulas specifying variable types. Accepted values are <code>c("continuous", "continuous2", "dichotomous")</code> , e.g. <code>type = list(age ~ "continuous", female ~ "dichotomous")</code> . If type not specified for a variable, the function will default to an appropriate summary type. See below for details.
value	List of formulas specifying the value to display for dichotomous variables. See below for details.
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
sort	List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. <code>sort = list(everything() ~ "frequency")</code>
percent	Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".
include	variables to include in the summary table. Default is <code>everything()</code>

Value

A `tbl_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

```
# 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_yesno = NULL,
  exclude = NULL
)
```

Arguments

<code>data</code>	Data frame to be used in univariate regression modeling. Data frame includes the outcome variable(s) and the independent variables. Survey design objects are also accepted.
<code>method</code>	Regression method (e.g. <code>lm</code> , <code>glm</code> , <code>survival::coxph</code> , <code>survey::svyglm</code> , and more).
<code>y</code>	Model outcome (e.g. <code>y = recurrence</code> or <code>y = Surv(time, recur)</code>). All other column in data will be regressed on <code>y</code> . Specify one and only one of <code>y</code> or <code>x</code>

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

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

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_q\(\)](#), [bold_italicize_labels_levels](#), [inline_text.tbl_uvregression](#), [modify](#), [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

*Tests/methods available in add_p() and add_difference()***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) %>%</code>
"kruskal.test"	Kruskal-Wallis test	<code>kruskal.test(data[[variable]], as.factor(by))</code>
"wilcox.test"	Wilcoxon rank-sum test	<code>wilcox.test(as.numeric(variable) ~ as.factor(by))</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 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.vanderwaerden.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_col
"paired.wilcox.test"	Paired Wilcoxon rank-sum test	rate difference	tidyr::pivot_wider(id_col
"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_survfi
<code>data=</code>	A data frame	A survey object	A survfi
<code>variable=</code>	String variable name	String variable name	NA
<code>by=</code>	String variable name	String variable name	NA
<code>group=</code>	String variable name	NA	NA
<code>type=</code>	Summary type	Summary type	NA
<code>conf.level=</code>	Confidence interval level	NA	NA
<code>adj.vars=</code>	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", "lancet", "nejm", "qjecon"),
  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", "is", "ja", "kr", "mr", "pt", "se",
    "zh-cn", "zh-tw"),
  decimal.mark = NULL,
  big.mark = NULL,
  iqr.sep = NULL,
  ci.sep = NULL,
  set_theme = TRUE
)

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

theme_gtsummary_mean_sd(set_theme = TRUE)

theme_gtsummary_eda(set_theme = TRUE)

```

Arguments

journal	String indicating the journal theme to follow. One of c("jama", "lancet", "nejm", "qjcon"). Details below.
set_theme	Logical indicating whether to set the theme. Default is TRUE. When FALSE the named list of theme elements is returned invisibly
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"
language	String indicating language. Must be one of "de" (German), "en" (English), "es" (Spanish), "fr" (French), "gu" (Gujarati), "hi" (Hindi), "is" (Icelandic), "ja" (Japanese), "kr" (Korean), "mr" (Marathi), "pt" (Portuguese), "se" (Swedish), "zh-c,n" (Chinese Simplified), "zh-tw" (Chinese Traditional) If a language is missing a translation for a word or phrase, please feel free to reach out on GitHub with the translated text!
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ",", except when decimal.mark = "." when the default is a space.
iqr.sep	string indicating separator for the default IQR in tbl_summary(). If decimal.mark= is NULL, iqr.sep= is ", ". The comma separator, however, can look odd when decimal.mark = ".", ". In this case the argument will default to an en dash
ci.sep	string indicating separator for confidence intervals. If decimal.mark= is NULL, ci.sep= is ", ". The comma separator, however, can look odd when decimal.mark = ".", ". In this case the argument will default to an en dash
statistic	Default statistic continuous variables

Themes

- theme_gtsummary_journal(journal=)

- "jama" *The Journal of the American Medical Association*
 - * Round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
 - * `tbl_summary()` Doesn't show percent symbol; use em-dash to separate IQR; run `add_stat_label()`
 - * `tbl_regression()/tbl_uvregression()` show coefficient and CI in same column
- "lancet" *The Lancet*
 - * Use mid-point as decimal separator; round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
 - * `tbl_summary()` Doesn't show percent symbol; use em-dash to separate IQR
- "nejm" *The New England Journal of Medicine*
 - * Round large p-values to 2 decimal places; separate confidence intervals with "ll to ul".
 - * `tbl_summary()` Doesn't show percent symbol; use em-dash to separate IQR
- "qjecon" *The Quarterly Journal of Economics* **Under Development**
 - * `tbl_summary()` all percentages rounded to one decimal place
 - * `tbl_regression()/tbl_uvregression()` add significance stars with `add_significance_stars()`; hides CI and p-value from output
- `theme_gtsummary_compact()`
 - tables printed with `gt`, `flextable`, `kableExtra`, or `huxtable` will be compact with smaller font size and reduced cell padding
- `theme_gtsummary_printer(print_engine=)`
 - Use this theme to permanently change the default printer.
- `theme_gtsummary_continuous2()`
 - Set all continuous variables to summary type "continuous2" by default
- `theme_gtsummary_mean_sd()`
 - Set default summary statistics to mean and standard deviation in `tbl_summary()`
 - Set default continuous tests in `add_p()` to t-test and ANOVA
- `theme_gtsummary_eda()`
 - Set all continuous variables to summary type "continuous2" by default
 - In `tbl_summary()` show the median, mean, IQR, SD, and Range by default

Use `reset_gtsummary_theme()` to restore the default settings

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

Example Output

See Also

[Themes vignette](#)

`set_gtsummary_theme()`, `reset_gtsummary_theme()`

Examples

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

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

# reset gtsummary themes
reset_gtsummary_theme()
```

 trial

Results from a simulated study of two chemotherapy agents

Description

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

Usage

```
trial
```

Format

A data frame with 200 rows—one row per patient

trt Chemotherapy Treatment

age Age

marker Marker Level (ng/mL)

stage T Stage

grade Grade

response Tumor Response

death Patient Died

ttdeath Months to Death/Censor

Index

- * **Advanced modifiers**
 - modify_column_hide, 55
 - modify_fmt_fun, 55
 - modify_table_body, 56
 - modify_table_styling, 58
 - * **datasets**
 - trial, 99
 - * **gtsummary output types**
 - as_flex_table, 31
 - as_gt, 33
 - as_hux_table, 34
 - as_kable, 35
 - as_kable_extra, 36
 - as_tibble.gtsummary, 37
 - * **style tools**
 - style_number, 65
 - style_percent, 66
 - style_pvalue, 67
 - style_ratio, 68
 - style_sigfig, 69
 - * **tbl_cross tools**
 - add_p.tbl_cross, 17
 - inline_text.tbl_cross, 45
 - tbl_cross, 70
 - * **tbl_regression tools**
 - add_global_p, 7
 - add_q, 23
 - bold_italicize_labels_levels, 38
 - combine_terms, 40
 - inline_text.tbl_regression, 46
 - modify, 52
 - tbl_merge, 71
 - tbl_regression, 73
 - tbl_stack, 77
 - * **tbl_summary tools**
 - add_n.tbl_summary, 10
 - add_overall, 15
 - add_p.tbl_summary, 18
 - add_q, 23
 - add_stat_label, 28
 - bold_italicize_labels_levels, 38
 - inline_text.tbl_summary, 47
 - inline_text.tbl_survfit, 49
 - modify, 52
 - tbl_merge, 71
 - tbl_stack, 77
 - tbl_summary, 80
 - * **tbl_survfit tools**
 - add_n.tbl_survfit, 11
 - add_nevent.tbl_survfit, 13
 - add_p.tbl_survfit, 20
 - modify, 52
 - tbl_merge, 71
 - tbl_stack, 77
 - tbl_survfit, 84
 - * **tbl_svysummary tools**
 - add_n.tbl_summary, 10
 - add_overall, 15
 - add_p.tbl_svysummary, 21
 - add_q, 23
 - add_stat_label, 28
 - modify, 52
 - tbl_merge, 71
 - tbl_stack, 77
 - tbl_svysummary, 87
 - * **tbl_uvregression tools**
 - add_global_p, 7
 - add_q, 23
 - bold_italicize_labels_levels, 38
 - inline_text.tbl_uvregression, 50
 - modify, 52
 - tbl_merge, 71
 - tbl_stack, 77
 - tbl_uvregression, 91
- add_difference, 4
- add_glance, 5
- add_glance_source_note (add_glance), 5
- add_glance_table (add_glance), 5
- add_global_p, 7, 24, 39, 41, 47, 51, 54, 72, 75, 78, 93
- add_n, 9
- add_n.tbl_regression, 9
- add_n.tbl_regression (add_n_regression), 14
- add_n.tbl_summary, 10, 16, 19, 23, 24, 29, 39, 48, 50, 54, 72, 78, 83, 90

- add_n.tbl_summary(), 9
- add_n.tbl_survfit, 11, 13, 21, 54, 72, 78, 86
- add_n.tbl_survfit(), 9
- add_n.tbl_svsummary
 - (add_n.tbl_summary), 10
- add_n.tbl_svsummary(), 9
- add_n.tbl_uvregression, 9
- add_n.tbl_uvregression
 - (add_n_regression), 14
- add_n_regression, 14
- add_nevent, 12
- add_nevent.tbl_regression, 12
- add_nevent.tbl_regression
 - (add_nevent_regression), 13
- add_nevent.tbl_survfit, 12, 13, 21, 54, 72, 78, 86
- add_nevent.tbl_uvregression, 12
- add_nevent.tbl_uvregression
 - (add_nevent_regression), 13
- add_nevent_regression, 13
- add_overall, 11, 15, 19, 23, 24, 29, 39, 48, 50, 54, 72, 78, 83, 90
- add_p, 16, 82
- add_p.tbl_cross, 17, 17, 45, 71
- add_p.tbl_summary, 11, 16, 17, 18, 24, 29, 39, 48, 50, 54, 72, 78, 83
- add_p.tbl_survfit, 12, 13, 17, 20, 54, 72, 78, 86
- add_p.tbl_svsummary, 11, 16, 17, 21, 24, 29, 54, 72, 78, 90
- add_q, 8, 11, 16, 19, 23, 23, 29, 39, 41, 47, 48, 50, 51, 54, 72, 75, 78, 83, 90, 93
- add_significance_stars, 24
- add_stat, 26
- add_stat_label, 11, 16, 19, 23, 24, 28, 39, 48, 50, 54, 72, 78, 83, 90
- add_vif, 30
- all_categorical(select_helpers), 62
- all_continuous(select_helpers), 62
- all_continuous2(select_helpers), 62
- all_contrasts(select_helpers), 62
- all_dichotomous(select_helpers), 62
- all_interaction(select_helpers), 62
- all_intercepts(select_helpers), 62
- all_stat_cols(select_helpers), 62
- all_tests(select_helpers), 62
- as_flex_table, 31, 33, 35–38
- as_gt, 32, 33, 35–38
- as_hux_table, 32, 33, 34, 36–38
- as_kable, 32, 33, 35, 35, 37, 38
- as_kable_extra, 32, 33, 35, 36, 36, 38
- as_tibble.gtsummary, 32, 33, 35–37, 37
- bold_italicize_labels_levels, 8, 11, 16, 19, 24, 29, 38, 41, 47, 48, 50, 51, 54, 72, 75, 78, 83, 93
- bold_labels
 - (bold_italicize_labels_levels), 38
- bold_labels(), 36
- bold_levels
 - (bold_italicize_labels_levels), 38
- bold_p, 39
- broom::tidy(), 75
- combine_terms, 8, 24, 39, 40, 47, 54, 72, 75, 78
- custom_tidiers, 41
- filter_p(sort_filter_p), 64
- glm, 91
- glue::glue, 10, 44, 46, 48, 51, 82, 89, 92
- glue::glue(), 53
- gt::gt, 33
- gt::html(), 6, 53
- gt::md(), 6, 53
- gtsummary themes, 63
- inline_text, 43
- inline_text.gtsummary, 44, 44
- inline_text.tbl_cross, 17, 44, 45, 71
- inline_text.tbl_regression, 8, 24, 39, 41, 44, 46, 54, 72, 75, 78
- inline_text.tbl_summary, 11, 16, 19, 24, 29, 39, 44, 47, 50, 54, 72, 78, 83
- inline_text.tbl_survfit, 11, 16, 19, 24, 29, 39, 44, 48, 49, 54, 72, 78, 83
- inline_text.tbl_svsummary, 44
- inline_text.tbl_svsummary
 - (inline_text.tbl_summary), 47
- inline_text.tbl_uvregression, 8, 24, 39, 44, 50, 54, 72, 78, 93
- italicize_labels
 - (bold_italicize_labels_levels), 38
- italicize_levels
 - (bold_italicize_labels_levels), 38
- italicize_levels(), 36
- knit_print.gtsummary(print_gtsummary), 60
- knitr::kable, 35–37
- lm, 91

- modifications, [74, 77, 93](#)
- modify, [8, 11–13, 16, 19, 21, 23, 24, 29, 39, 41, 47, 48, 50, 51, 52, 72, 75, 78, 83, 86, 90, 93](#)
- modify_caption(modify), [52](#)
- modify_column_hide, [55, 56, 57, 59](#)
- modify_column_unhide
(modify_column_hide), [55](#)
- modify_fmt_fun, [55, 55, 57, 59](#)
- modify_footnote(modify), [52](#)
- modify_header(modify), [52](#)
- modify_spanning_header(modify), [52](#)
- modify_table_body, [55, 56, 56, 59](#)
- modify_table_styling, [55–57, 58](#)
- plot, [59](#)
- pool_and_tidy_mice(custom_tidiers), [41](#)
- print.gtsummary(print_gtsummary), [60](#)
- print_gtsummary, [60](#)
- remove_row_type, [61](#)
- reset_gtsummary_theme
(set_gtsummary_theme), [63](#)
- reset_gtsummary_theme(), [98](#)
- select_helpers, [62](#)
- set_gtsummary_theme, [63](#)
- set_gtsummary_theme(), [98](#)
- show_header_names(modify), [52](#)
- sort_filter_p, [64](#)
- sort_p(sort_filter_p), [64](#)
- stats::anova, [40](#)
- stats::anova(), [40](#)
- stats::glm, [12](#)
- stats::p.adjust, [23](#)
- stats::update, [40](#)
- style_number, [65, 66–69](#)
- style_percent, [65, 66, 67–69](#)
- style_percent(), [85](#)
- style_pvalue, [4, 17, 18, 20, 22, 23, 45, 48, 49, 65, 66, 67, 68, 69, 74, 92](#)
- style_ratio, [65–67, 68, 69, 74, 92](#)
- style_sigfig, [65–68, 69, 74, 85, 92](#)
- style_sigfig(), [4, 30](#)
- survival::coxph, [12, 91](#)
- tbl_cross, [17, 45, 70](#)
- tbl_merge, [8, 11–13, 16, 19, 21, 23, 24, 29, 39, 41, 47, 48, 50, 51, 54, 61, 71, 75, 78, 83, 86, 90, 93](#)
- tbl_regression, [8, 12, 24, 31, 33–37, 39, 41, 46, 47, 54, 61, 72, 73, 78, 91](#)
- tbl_regression.brmsfit
(tbl_regression_methods), [75](#)
- tbl_regression.default(), [75](#)
- tbl_regression.gam
(tbl_regression_methods), [75](#)
- tbl_regression.glmerMod
(tbl_regression_methods), [75](#)
- tbl_regression.glmmladmb
(tbl_regression_methods), [75](#)
- tbl_regression.glmmlTMB
(tbl_regression_methods), [75](#)
- tbl_regression.lmerMod
(tbl_regression_methods), [75](#)
- tbl_regression.mipo
(tbl_regression_methods), [75](#)
- tbl_regression.mira
(tbl_regression_methods), [75](#)
- tbl_regression.model_fit
(tbl_regression_methods), [75](#)
- tbl_regression.multinom
(tbl_regression_methods), [75](#)
- tbl_regression.stanreg
(tbl_regression_methods), [75](#)
- tbl_regression.survreg
(tbl_regression_methods), [75](#)
- tbl_regression.workflow
(tbl_regression_methods), [75](#)
- tbl_regression_methods, [75](#)
- tbl_stack, [8, 11–13, 16, 19, 21, 23, 24, 29, 39, 41, 47, 48, 50, 51, 54, 61, 72, 75, 77, 83, 86, 90, 93](#)
- tbl_strata, [79](#)
- tbl_summary, [10, 11, 16, 18, 19, 24, 28, 29, 31, 33–37, 39, 48, 50, 54, 61, 72, 78, 80](#)
- tbl_summary(), [87, 89](#)
- tbl_survfit, [12, 13, 21, 49, 54, 72, 78, 84](#)
- tbl_survfit.data.frame(), [85](#)
- tbl_survfit.list(), [85](#)
- tbl_survfit.survfit(), [85](#)
- tbl_survfit_errors, [87](#)
- tbl_svsummary, [10, 11, 16, 22–24, 28, 29, 54, 72, 78, 87](#)
- tbl_uvregression, [8, 12, 24, 39, 51, 54, 61, 72, 78, 91](#)
- tests, [4, 18, 19, 94](#)
- theme_gtsummary, [96](#)
- theme_gtsummary_compact
(theme_gtsummary), [96](#)
- theme_gtsummary_continuous2
(theme_gtsummary), [96](#)
- theme_gtsummary_eda(theme_gtsummary), [96](#)
- theme_gtsummary_journal

(theme_gtsummary), 96
theme_gtsummary_language
 (theme_gtsummary), 96
theme_gtsummary_mean_sd
 (theme_gtsummary), 96
theme_gtsummary_printer
 (theme_gtsummary), 96
tibble, 38
tidy_bootstrap(custom_tidiers), 41
tidy_gam(custom_tidiers), 41
tidy_standardize(custom_tidiers), 41
trial, 99