Package 'gtsummary'

May 8, 2026

Title Presentation-Ready Data Summary and Analytic Result Tables

Version 2.5.0

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

License MIT + file LICENSE

URL https://github.com/ddsjoberg/gtsummary, https://www.danieldsjoberg.com/gtsummary/

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

Depends R (>= 4.2)

Imports cards (>= 0.7.1), cardx (>= 0.3.1), cli (>= 3.6.3), dplyr (>= 1.1.3), glue (>= 1.8.0), gt (>= 0.11.1), lifecycle (>= 1.0.3), rlang (>= 1.1.1), tidyr (>= 1.3.0), vctrs (>= 0.6.4)

Suggests aod (>= 1.3.3), broom (>= 1.0.5), broom.helpers (>= 1.20.0), broom.mixed (>= 0.2.9), car (>= 3.0-11), cmprsk, effectsize (>= 0.6.0), emmeans (>= 1.7.3), flextable (>= 0.8.1), geepack (>= 1.3.10), ggstats (>= 0.2.1), huxtable (>= 5.4.0), insight (>= 0.15.0), kableExtra (>= 1.3.4), knitr (>= 1.37), lme4 (>= 1.1-31), mice (>= 3.10.0), nnet, officer, openxlsx, parameters (>= 0.20.2), parsnip (>= 0.1.7), rmarkdown, smd (>= 0.6.6), spelling, survey (>= 4.2), survival (>= 3.6-4), testthat (>= 3.2.0), withr (>= 2.5.0), workflows (>= 0.2.4)

VignetteBuilder knitr

Config/Needs/check hms

Config/Needs/website forcats, sandwich, scales

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Daniel D. Sjoberg [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0862-2018>),
Joseph Larmarange [aut] (ORCID:
<https://orcid.org/0000-0001-7097-700X>),
Michael Curry [aut] (ORCID: <https://orcid.org/0000-0002-0261-4044>),
Emily de la Rua [aut] (ORCID: <https://orcid.org/0009-0000-8738-5561>),
Jessica Lavery [aut] (ORCID: <https://orcid.org/0000-0002-2746-5647>),
Karissa Whiting [aut] (ORCID: <https://orcid.org/0000-0002-4683-1868>),
Emily C. Zabor [aut] (ORCID: <https://orcid.org/0000-0002-1402-4498>),
Xing Bai [ctb],
Malcolm Barrett [ctb] (ORCID: <https://orcid.org/0000-0003-0299-5825>),
Esther Drill [ctb] (ORCID: <https://orcid.org/0000-0002-3315-4538>),
Jessica Flynn [ctb] (ORCID: <https://orcid.org/0000-0001-8310-6684>),
Margie Hannum [ctb] (ORCID: <https://orcid.org/0000-0002-2953-0449>),
Stephanie Lobaugh [ctb],
Shannon Pileggi [ctb] (ORCID: <https://orcid.org/0000-0002-7732-4164>),
Amy Tin [ctb] (ORCID: <https://orcid.org/0000-0002-8005-0694>),
Gustavo Zapata Wainberg [ctb] (ORCID:
<https://orcid.org/0000-0002-2524-3637>)

Maintainer Daniel D. Sjoberg <danield.sjoberg@gmail.com>

Repository CRAN

Date/Publication 2025-12-05 11:40:07 UTC

Contents

add_ci *Add CI Column*

Description

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

Usage

```
add_ci(x, ...)

## S3 method for class 'tbl_summary'
add_ci(
  x,
  method = list(all_continuous() ~ "t.test", all_categorical() ~ "wilson"),
  include = everything(),
 statistic = list(all_continuous() ~ "{conf.low}, {conf.high}", all_categorical() ~
    "{conf.low}%, {conf.high}%"),
  conf.level = 0.95,
  style_fun = list(all_continuous() ~ label_style_sigfig(), all_categorical() ~
    label_style_sigfig(scale = 100)),
  pattern = NULL,
  ...
)
```

Arguments

x	(tbl_summary) a summary table of class 'tblsummary'
...	These dots are for future extensions and must be empty.
method	(formula-list-selector) Confidence interval method. Default is list(all_continuous() ~ "t.test", all_categorical() ~ "wilson"). See details below.
include	(tidy-select) Variables to include in the summary table. Default is everything().
statistic	(formula-list-selector) Indicates how the confidence interval will be displayed. Default is list(all_continuous() ~ "{conf.low}, {conf.high}", all_categorical() ~ "{conf.low}%, {conf.high}%")
conf.level	(scalar real) Confidence level. Default is 0.95
style_fun	(function) Function to style upper and lower bound of confidence interval. Default is list(all_continuous() ~ label_style_sigfig(), all_categorical() ~ label_style_sigfig(s = 100)).

`pattern` (`string`)
Indicates the pattern to use to merge the CI with the statistics cell. The default is NULL, where no columns are merged. The two columns that will be merged are the statistics column, represented by `"{stat}"` and the CI column represented by `"{ci}"`, e.g. `pattern = "{stat} ({ci})"` will merge the two columns with the CI in parentheses. Default is `NULL`, and no merging is performed.

Value

gtsummary table

method argument

Must be one of

- `"wilson"`, `"wilson.no.correct"` calculated via `prop.test(correct = c(TRUE, FALSE))` for **categorical** variables
- `"exact"` calculated via `stats::binom.test()` for **categorical** variables
- `"wald"`, `"wald.no.correct"` calculated via `cardx::proportion_ci_wald(correct = c(TRUE, FALSE)` for **categorical** variables
- `"agresti.coull"` calculated via `cardx::proportion_ci_agresti_coull()` for **categorical** variables
- `"jeffreys"` calculated via `cardx::proportion_ci_jeffreys()` for **categorical** variables
- `"t.test"` calculated via `stats::t.test()` for **continuous** variables
- `"wilcox.test"` calculated via `stats::wilcox.test()` for **continuous** variables

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(
    missing = "no",
    statistic = all_continuous() ~ "{mean} ({sd})",
    include = c(marker, response, trt)
  ) |>
  add_ci()

# Example 2 ----------------------------------
trial |>
  select(response, grade) %>%
  tbl_summary(
    statistic = all_categorical() ~ "{p}%",
    missing = "no",
    include = c(response, grade)
  ) |>
  add_ci(pattern = "{stat} ({ci})") |>
  remove_footnote_header(everything())
```

add_ci.tbl_svysummary *Add CI Column*

Description

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

Usage

```
## S3 method for class 'tbl_svysummary'
add_ci(
  x,
 method = list(all_continuous() ~ "svymean", all_categorical() ~ "svyprop.logit"),
  include = everything(),
 statistic = list(all_continuous() ~ "{conf.low}, {conf.high}", all_categorical() ~
    "{conf.low}%, {conf.high}%"),
  conf.level = 0.95,
  style_fun = list(all_continuous() ~ label_style_sigfig(), all_categorical() ~
    label_style_sigfig(scale = 100)),
  pattern = NULL,
  df = survey::degf(x$inputs$data),
  ...
)
```

Arguments

x	(tbl_summary) a summary table of class 'tblsummary'
method	(formula-list-selector) Confidence interval method. Default is list(all_continuous() ~ "svymean", all_categorical() ~ "svyprop.logit"). See details below.
include	(tidy-select) Variables to include in the summary table. Default is everything().
statistic	(formula-list-selector) Indicates how the confidence interval will be displayed. Default is list(all_continuous() ~ "{conf.low}, {conf.high}", all_categorical() ~ "{conf.low}%, {conf.high}%")
conf.level	(scalar real) Confidence level. Default is 0.95
style_fun	(function) Function to style upper and lower bound of confidence interval. Default is list(all_continuous() ~ label_style_sigfig(), all_categorical() ~ label_style_sigfig(s = 100)).
pattern	(string) Indicates the pattern to use to merge the CI with the statistics cell. The default is NULL, where no columns are merged. The two columns that will be merged are

	the statistics column, represented by `"{stat}"` and the CI column represented by `"{ci}"`, e.g. `pattern = "{stat} ({ci})"` will merge the two columns with the CI in parentheses. Default is `NULL`, and no merging is performed.
`df`	(`numeric`) denominator degrees of freedom, passed to `survey::svyciprop(df)` or `confint(df)`. Default is `survey::degf(x$inputs$data)`.
`...`	These dots are for future extensions and must be empty.

Value

gtsummary table

method argument

Must be one of

- `"svyprop.logit"`, `"svyprop.likelihood"`, `"svyprop.asin"`, `"svyprop.beta"`, `"svyprop.mean"`, `"svyprop.xlogit"` calculated via `survey::svyciprop()` for **categorical** variables
- `"svymean"` calculated via `survey::svymean()` for **continuous** variables
- `"svymedian.mean"`, `"svymedian.beta"`, `"svymedian.xlogit"`, `"svymedian.asin"`, `"svymedian.score"` calculated via `survey::svyquantile(quantiles = 0.5)` for **continuous** variables

Examples

```
data(api, package = "survey")
survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc) |>
  tbl_svysummary(
    by = "both",
    include = c(api00, stype),
    statistic = all_continuous() ~ "{mean} ({sd})"
  ) |>
  add_stat_label() |>
  add_ci(pattern = "{stat} (95% CI {ci})") |>
  modify_header(all_stat_cols() ~ "**{level}**") |>
  modify_spanning_header(all_stat_cols() ~ "**Survived**")
```

---

add_difference.tbl_summary

Add differences between groups

Description

Adds difference to tables created by `tbl_summary()`. The difference between two groups (typically mean or rate difference) is added to the table along with the difference's confidence interval and a p-value (when applicable).

Usage

```
## S3 method for class 'tbl_summary'
add_difference(
  x,
  test = NULL,
  group = NULL,
  adj.vars = NULL,
  test.args = NULL,
  conf.level = 0.95,
  include = everything(),
  pvalue_fun = label_style_pvalue(digits = 1),
 estimate_fun = list(c(all_continuous(), all_categorical(FALSE)) ~ label_style_sigfig(),
   all_dichotomous() ~ label_style_sigfig(scale = 100, suffix = "%"), all_tests("smd")
    ~ label_style_sigfig()),
  ...
)
```

Arguments

x	(tbl_summary) table created with tbl_summary()
test	(formula-list-selector) Specifies the tests/methods to perform for each variable, e.g. list(all_continuous() ~ "t.test", all_dichotomous() ~ "prop.test", all_categorical(FALSE) ~ "smd"). See below for details on default tests and ?tests for details on available tests and creating custom tests.
group	(tidy-select) Variable name 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.
adj.vars	(tidy-select) Variables to include in adjusted calculations (e.g. in ANCOVA models). Default is NULL.
test.args	(formula-list-selector) 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).
conf.level	(numeric) a scalar in the interval (0, 1) indicating the confidence level. Default is 0.95
include	(tidy-select) Variables to include in output. Default is everything().
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).

estimate_fun	(formula-list-selector) List of formulas specifying the functions to round and format differences and confidence limits.
...	These dots are for future extensions and must be empty.

Value

a gtsummary table of class "tbl_summary"

Examples

```
# Example 1 ----------------------------------
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
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_difference.tbl_svysummary

Add differences between groups

Description

Adds difference to tables created by tbl_summary(). The difference between two groups (typically mean or rate difference) is added to the table along with the difference's confidence interval and a p-value (when applicable).

Usage

```
## S3 method for class 'tbl_svysummary'
add_difference(
  x,
  test = NULL,
  group = NULL,
  adj.vars = NULL,
  test.args = NULL,
  conf.level = 0.95,
  include = everything(),
  pvalue_fun = label_style_pvalue(digits = 1),
 estimate_fun = list(c(all_continuous(), all_categorical(FALSE)) ~ label_style_sigfig(),
   all_dichotomous() ~ label_style_sigfig(scale = 100, suffix = "%"), all_tests("smd")
    ~ label_style_sigfig()),
  ...
)
```

Arguments

x	(tbl_summary) table created with tbl_summary()
test	(formula-list-selector) Specifies the tests/methods to perform for each variable, e.g. list(all_continuous() ~ "svy.t.test", all_dichotomous() ~ "emmeans", all_categorical(FALSE) ~ "svy.chisq.test"). See below for details on default tests and ?tests for details on available tests and creating custom tests.
group	(tidy-select) Variable name 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.
adj.vars	(tidy-select) Variables to include in adjusted calculations (e.g. in ANCOVA models). Default is NULL.
test.args	(formula-list-selector) 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).
conf.level	(numeric) a scalar in the interval (0, 1) indicating the confidence level. Default is 0.95
include	(tidy-select) Variables to include in output. Default is everything().
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).

estimate_fun (formula-list-selector)
List of formulas specifying the functions to round and format differences and confidence limits. Default is `list(c(all_continuous(), all_categorical(FALSE)) ~ label_style`

... These dots are for future extensions and must be empty.

Value

a gtsummary table of class "tbl_summary"

Examples

`add_difference_row.tbl_summary`
Add differences rows between groups

Description

[Experimental]
Adds difference to tables created by `tbl_summary()` as additional rows. This function is often useful when there are more than two groups to compare.

Pairwise differences are calculated relative to the specified by variable's specified reference level.

Usage

```
## S3 method for class 'tbl_summary'
add_difference_row(
  x,
  reference,
  statistic = everything() ~ "{estimate}",
  test = NULL,
  group = NULL,
  header = NULL,
  adj.vars = NULL,
  test.args = NULL,
  conf.level = 0.95,
  include = everything(),
  pvalue_fun = label_style_pvalue(digits = 1),
 estimate_fun = list(c(all_continuous(), all_categorical(FALSE)) ~ label_style_sigfig(),
   all_dichotomous() ~ label_style_sigfig(scale = 100, suffix = "%"), all_tests("smd")
    ~ label_style_sigfig()),
  ...
)
```

Arguments

x	(tbl_summary) table created with tbl_summary()
reference	(scalar) Value of the tbl_summary(by) variable value that is the reference for each of the difference calculations. For factors, use the character level.
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is everything() ~ "{estimate}". The statistics available to include will depend on the method specified in the test argument, but are generally "estimate", "std.error", "parameter", "statistic", "conf.low", "conf.high", "p.value".
test	(formula-list-selector) Specifies the tests/methods to perform for each variable, e.g. list(all_continuous() ~ "t.test", all_dichotomous() ~ "prop.test", all_categorical(FALSE) ~ "smd"). See below for details on default tests and ?tests for details on available tests and creating custom tests.
group	(tidy-select) Variable name 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.
header	(string) When supplied, a header row will appear above the difference statistics.
adj.vars	(tidy-select) Variables to include in adjusted calculations (e.g. in ANCOVA models). Default is NULL.
test.args	(formula-list-selector) 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).
conf.level	(numeric) a scalar in the interval (0, 1) indicating the confidence level. Default is 0.95
include	(tidy-select) Variables to include in output. Default is everything().
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
estimate_fun	(formula-list-selector) List of formulas specifying the functions to round and format differences and confidence limits.
...	These dots are for future extensions and must be empty.

Details

The default labels for the statistic rows will often *not* be what you need to display. In cases like this, use `modify_table_body()` to directly update the label rows. Use `show_header_names()` to print the underlying column names to identify the columns to target when changing the label, which in this case will always be the `'label'` column. See Example 2.

Value

a gtsummary table of class `"tbl_summary"`

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(
    by = grade,
    include = c(age, response),
    missing = "no",
    statistic = all_continuous() ~ "{mean} ({sd})"
  ) |>
  add_stat_label() |>
  add_difference_row(
    reference = "I",
    statistic = everything() ~ c("{estimate}", "{conf.low}, {conf.high}", "{p.value}")
  )

# Example 2 ----------------------------------
# Function to build age-adjusted logistic regression and put results in ARD format
ard_odds_ratio <- \(data, variable, by, ...) {
  cardx::construct_model(
    data = data,
   formula = reformulate(response = variable, termlabels = c(by, "age")), # adjusting model for age
    method = "glm",
    method.args = list(family = binomial)
  ) |>
    cardx::ard_regression_basic(exponentiate = TRUE) |>
    dplyr::filter(.data$variable == .env$by)
}

trial |>
  tbl_summary(by = trt, include = response, missing = "no") |>
  add_stat_label() |>
  add_difference_row(
    reference = "Drug A",
    statistic = everything() ~ c("{estimate}", "{conf.low}, {conf.high}", "{p.value}"),
    test = everything() ~ ard_odds_ratio,
    estimate_fun = everything() ~ label_style_ratio()
  ) |>
  # change the default label for the 'Odds Ratio'
  modify_table_body(
    ~ .x |>
      dplyr::mutate(
```

```
      label = ifelse(label == "Coefficient", "Odds Ratio", label)
    )
) |>
# add footnote about logistic regression
modify_footnote_body(
  footnote = "Age-adjusted logistic regression model",
  column = "label",
  rows = variable == "response-row_difference"
)
```

---

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 = list(everything() ~ label_style_sigfig(digits = 3), any_of("p.value") ~
    label_style_pvalue(digits = 1), c(where(is.integer), starts_with("df")) ~
    label_style_number()),
  glance_fun = glance_fun_s3(x$inputs$x)
)

add_glance_source_note(
  x,
  include = everything(),
  label = NULL,
 fmt_fun = list(everything() ~ label_style_sigfig(digits = 3), any_of("p.value") ~
    label_style_pvalue(digits = 1), c(where(is.integer), starts_with("df")) ~
    label_style_number()),
  glance_fun = glance_fun_s3(x$inputs$x),
  text_interpret = c("md", "html"),
  sep1 = " = ",
  sep2 = "; "
)
```

Arguments

x	(tbl_regression) a 'tbl_regression' object

include (tidy-select)
names of statistics to include in output. Must be column names of the tibble returned by broom::glance() or from the glance_fun argument. The include argument can also be used to specify the order the statistics appear in the table.

label (formula-list-selector)
specifies statistic labels, e.g. list(r.squared = "R2", p.value = "P")

fmt_fun (formula-list-selector)
Specifies the the functions used to format/round the glance 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)
function that returns model statistics. Default is glance_fun() (which is broom::glance() for most model objects). Custom functions must return a single row tibble.

text_interpret (string)
String indicates whether source note text will be interpreted with gt::md() or gt::html(). Must be "md" (default) or "html".

sep1 (string)
Separator between statistic name and statistic. Default is " = ", e.g. "R2 = 0.456"

sep2 (string)
Separator between statistics. Default is "; "

Value

gtsummary table

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 |> dplyr::arrange(row_type == "glance_statistic"))
```

Examples

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

# Example 1 ----------------------------------
mod |>
  add_glance_table(
    label = list(sigma = "\U03C3"),
    include = c(r.squared, AIC, sigma)
  )

# Example 2 ----------------------------------
```

```
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()` (by default) to calculate global p-values for model covariates. 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(),
  keep = FALSE,
  anova_fun = global_pvalue_fun,
  type = "III",
  quiet,
  ...
)

## S3 method for class 'tbl_uvregression'
add_global_p(
  x,
  include = everything(),
  keep = FALSE,
  anova_fun = global_pvalue_fun,
  type = "III",
  quiet,
  ...
)
```

Arguments

`x`	(`tbl_regression`, `tbl_uvregression`) Object with class `'tbl_regression'` or `'tbl_uvregression'`
`...`	Additional arguments to be passed to `car::Anova`, `aod::wald.test()` or `anova_fun` (if specified)

include	(tidy-select) Variables to calculate global p-value for. Default is everything()
keep	(scalar logical) 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.
anova_fun	(function) Function used to calculate global p-values. Default is generic global_pvalue_fun(), which wraps car::Anova() for most models. The type argument is passed to this function. See help file for details. To pass a custom function, it must accept as its first argument is a model. Note that anything passed in ... will be passed to this function. The function must return an object of class 'cards' (see cardx::ard_car_anova() as an example), or a tibble with columns 'term' and 'p.value' (e.g. \(x, type, ...) car::Anova(x, type, ...) |>
type	Type argument passed to anova_fun. Default is "III"
quiet	**[Deprecated]**

Author(s)

Daniel D. Sjoberg

Examples

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

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

---

add_n.tbl_survfit *Add N*

Description

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

Examples

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

# Example 1 ----------------------------------
list(fit1, fit2) |>
  tbl_survfit(times = c(12, 24)) |>
  add_n()
```

---

add_nevent.tbl_survfit

Add event N

Description

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

Usage

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

Arguments

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

See Also

Other tbl_survfit tools: `add_p.tbl_survfit()`

Examples

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

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

---

add_nevent_regression *Add event N*

Description

Add event N

Usage

```
add_nevent(x, ...)

## S3 method for class 'tbl_regression'
add_nevent(x, location = "label", ...)

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

Arguments

x	(tbl_regression, tbl_uvregression) a tbl_regression or tbl_uvregression table
...	These dots are for future extensions and must be empty.
location	(character) location to place Ns. Select one or more of c('label', 'level'). Default is 'label'. 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.

Examples

```
# Example 1 ----------------------------------
trial |>
  select(response, trt, grade) |>
  tbl_uvregression(
```

```
    y = response,
    exponentiate = TRUE,
    method = glm,
    method.args = list(family = binomial),
  ) |>
  add_nevent()

# Example 2 ----------------------------------
glm(response ~ age + grade, trial, family = binomial) |>
  tbl_regression(exponentiate = TRUE) |>
  add_nevent(location = "level")
```

`add_n_regression` *Add N to regression table*

Description

Add N to regression table

Usage

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

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

Arguments

`x`	(`tbl_regression`, `tbl_uvregression`) a `tbl_regression` or `tbl_uvregression` table
`location`	(`character`) location to place Ns. Select one or more of `c('label', 'level')`. Default is `'label'`. 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.
`...`	These dots are for future extensions and must be empty.

Examples

```
# Example 1 ----------------------------------
trial |>
  select(response, age, grade) |>
  tbl_uvregression(
    y = response,
    exponentiate = TRUE,
```

```
    method = glm,
    method.args = list(family = binomial),
    hide_n = TRUE
  ) |>
  add_n(location = "label")

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

---

add_n_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_nonmiss}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  ...
)

## S3 method for class 'tbl_svysummary'
add_n(
  x,
  statistic = "{N_nonmiss}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  ...
)

## S3 method for class 'tbl_likert'
add_n(
  x,
  statistic = "{N_nonmiss}",
  col_label = "**N**",
  footnote = FALSE,
```

```
  last = FALSE,
  ...
)
```

Arguments

x	(tbl_summary) Object with class 'tbl_summary' created with tbl_summary() function.
statistic	(string) String indicating the statistic to report. Default is the number of non-missing observation for each variable, statistic = "{N_nonmiss}". All statistics available to report include: • "{N_obs}" total number of observations, • "{N_nonmiss}" number of non-missing observations, • "{N_miss}" number of missing observations, • "{p_nonmiss}" percent non-missing data, • "{p_miss}" percent missing data The argument uses glue::glue() syntax and multiple statistics may be reported, e.g. statistic = "{N_nonmiss} / {N_obs} ({p_nonmiss}%)"
col_label	(string) String indicating the column label. Default is "**N**"
footnote	(scalar logical) Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is FALSE
last	(scalar logical) Logical indicator to include N column last in table. Default is FALSE, which will display N column first.
...	These dots are for future extensions and must be empty.

Value

A table of class c('tbl_summary', 'gtsummary')

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(by = trt, include = c(trt, age, grade, response)) |>
  add_n()

# Example 2 ----------------------------------
survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) |>
  tbl_svysummary(by = Survived, percent = "row", include = c(Class, Age)) |>
  add_n()
```

add_overall *Add overall column*

Description

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

Usage

```
add_overall(x, ...)

## S3 method for class 'tbl_summary'
add_overall(
  x,
  last = FALSE,
  col_label = "**Overall**  \nN = {style_number(N)}",
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_continuous'
add_overall(
  x,
  last = FALSE,
  col_label = "**Overall**  \nN = {style_number(N)}",
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_svysummary'
add_overall(
  x,
  last = FALSE,
  col_label = "**Overall**  \nN = {style_number(N)}",
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_custom_summary'
add_overall(
  x,
  last = FALSE,
  col_label = "**Overall**  \nN = {style_number(N)}",
```

```
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_hierarchical'
add_overall(
  x,
  last = FALSE,
  col_label = "**Overall**  \nN = {style_number(N)}",
  statistic = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'tbl_hierarchical_count'
add_overall(
  x,
  last = FALSE,
  col_label = ifelse(rlang::is_empty(x$inputs$denominator), "**Overall**",
    "**Overall**  \nN = {style_number(N)}"),
  statistic = NULL,
  digits = NULL,
  ...
)
```

Arguments

x	(tbl_summary, tbl_svysummary, tbl_continuous, tbl_custom_summary) A stratified 'gtsummary' table
...	These dots are for future extensions and must be empty.
last	(scalar logical) Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.
col_label	(string) String indicating the column label. Default is "**Overall** \nN = {style_number(N)}"
statistic	(formula-list-selector) Override the statistic argument in initial tbl_* function call. Default is NULL.
digits	(formula-list-selector) Override the digits argument in initial tbl_* function call. Default is NULL.

Value

A gtsummary of same class as x

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(include = c(age, grade), by = trt) |>
  add_overall()

# Example 2 ----------------------------------
trial |>
  tbl_summary(
    include = grade,
    by = trt,
    percent = "row",
    statistic = ~"{p}%",
    digits = ~1
  ) |>
  add_overall(
    last = TRUE,
    statistic = ~"{p}% (n={n})",
    digits = ~ c(1, 0)
  )

# Example 3 ----------------------------------
trial |>
  tbl_continuous(
    variable = age,
    by = trt,
    include = grade
  ) |>
  add_overall(last = TRUE)
```

---

add_overall_ard *ARD add overall column*

Description

Adds a column with overall summary statistics to tables created by `tbl_ard_summary()`.

Usage

```
## S3 method for class 'tbl_ard_summary'
add_overall(
  x,
  cards,
  last = FALSE,
  col_label = "**Overall**",
  statistic = NULL,
  ...
)
```

Arguments

x	(tbl_ard_summary) A stratified 'gtsummary' table
cards	(card) An ARD object of class "card" typically created with cards::ard_*() functions.
last	(scalar logical) Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.
col_label	(string) String indicating the column label. Default is "**Overall**"
statistic	(formula-list-selector) Override the statistic argument in initial tbl_* function call. Default is NULL.
...	These dots are for future extensions and must be empty.

Value

A gtsummary of same class as x

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
# build primary table
tbl <-
  cards::ard_stack(
    trial,
    .by = trt,
    cards::ard_summary(variables = age),
    cards::ard_tabulate(variables = grade),
    .missing = TRUE,
    .attributes = TRUE,
    .total_n = TRUE
  ) |>
  tbl_ard_summary(by = trt)

# create ARD with overall results
ard_overall <-
  cards::ard_stack(
    trial,
    cards::ard_summary(variables = age),
    cards::ard_tabulate(variables = grade),
    .missing = TRUE,
    .attributes = TRUE,
    .total_n = TRUE
  )
```

```
# add an overall column
tbl |>
  add_overall(cards = ard_overall)
```

`add_p.tbl_continuous` *Add p-values*

Description

Add p-values

Usage

```
## S3 method for class 'tbl_continuous'
add_p(
  x,
  test = NULL,
  pvalue_fun = label_style_pvalue(digits = 1),
  include = everything(),
  test.args = NULL,
  group = NULL,
  ...
)
```

Arguments

`x`	(`tbl_continuous`) table created with `tbl_continuous()`
`test`	List of formulas specifying statistical tests to perform for each variable. Default is two-way ANOVA when `by=` is not `NULL`, and has the same defaults as `add_p.tbl_continuous()` when `by = NULL`. See tests for details, more tests, and instruction for implementing a custom test.
`pvalue_fun`	(`function`) Function to round and format p-values. Default is `label_style_pvalue()`. The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = label_style_pvalue(digits = 2)`).
`include`	(`tidy-select`) Variables to include in output. Default is `everything()`.
`test.args`	(`formula-list-selector`) 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)`.

group (tidy-select)
Variable name 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.

... These dots are for future extensions and must be empty.

Value

'tbl_continuous' object

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_continuous(variable = age, by = trt, include = grade) |>
  add_p(pvalue_fun = label_style_pvalue(digits = 2))

# Example 2 ----------------------------------
trial |>
  tbl_continuous(variable = age, include = grade) |>
  add_p(test = everything() ~ "kruskal.test")
```

---

add_p.tbl_cross *Add p-value*

Description

Calculate and add a p-value comparing the two variables in the cross table. If missing levels are included in the tables, they are also included in p-value calculation.

Usage

```
## S3 method for class 'tbl_cross'
add_p(
  x,
  test = NULL,
 pvalue_fun = ifelse(source_note, label_style_pvalue(digits = 1, prepend_p = TRUE),
    label_style_pvalue(digits = 1)),
  source_note = FALSE,
  test.args = NULL,
  ...
)
```

Arguments

x	(tbl_cross) Object with class tbl_cross created with the tbl_cross() function
test	(string) A string specifying statistical test to perform. Default is "chisq.test" when expected cell counts >=5 and "fisher.test" when expected cell counts <5.
pvalue_fun	(function) Function to round and format p-value. Default is label_style_pvalue(digits = 1), except when source_note = TRUE when the default is label_style_pvalue(digits = 1, prepend_p = TRUE)
source_note	(scalar logical) Logical value indicating whether to show p-value in the {gt} table source notes rather than a column.
test.args	(named list) Named list containing additional arguments to pass to the test (if it accepts additional arguments). For example, add an argument for a chi-squared test with test.args = list(correct = TRUE)
...	These dots are for future extensions and must be empty.

Author(s)

Karissa Whiting, Daniel D. Sjoberg

Examples

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

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

---

add_p.tbl_summary *Add p-values*

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 = label_style_pvalue(digits = 1),
  group = NULL,
  include = everything(),
  test.args = NULL,
  adj.vars = NULL,
  ...
)
```

Arguments

x	(tbl_summary) table created with tbl_summary()
test	(formula-list-selector) Specifies the statistical tests to perform for each variable, e.g. list(all_continuous() ~ "t.test", all_categorical() ~ "fisher.test"). See below for details on default tests and ?tests for details on available tests and creating custom tests.
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
group	(tidy-select) Variable name 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	(tidy-select) Variables to include in output. Default is everything().
test.args	(formula-list-selector) 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).
adj.vars	(tidy-select) Variables to include in adjusted calculations (e.g. in ANCOVA models). Default is NULL.
...	These dots are for future extensions and must be empty.

Value

a gtsummary table of class "tbl_summary"

test argument

See the ?tests help file for details on available tests and creating custom tests. The ?tests help file also includes pseudo-code for each test to be clear precisely how the calculation is performed.

The default test used in `add_p()` primarily depends on these factors:

- whether the variable is categorical/dichotomous vs continuous
- number of levels in the `tbl_summary(by)` variable
- whether the `add_p(group)` argument is specified
- whether the `add_p(adj.vars)` argument is specified

Specified neither `add_p(group)` nor `add_p(adj.vars)`:

- `"wilcox.test"` when by variable has two levels and variable is continuous.
- `"kruskal.test"` when by variable has more than two levels and variable is continuous.
- `"chisq.test.no.correct"` for categorical variables with all expected cell counts >=5, and `"fisher.test"` for categorical variables with any expected cell count <5.

Specified `add_p(group)` and not `add_p(adj.vars)`:

- `"lme4"` when by variable has two levels for all summary types.

There is no default for grouped data when by *variable has more than two levels. Users must create custom tests for this scenario.*

Specified `add_p(adj.vars)` and not `add_p(group)`:

- `"ancova"` when variable is continuous and by variable has two levels.

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(by = trt, include = c(age, grade)) |>
  add_p()

# Example 2 ----------------------------------
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` *Add p-value*

Description

Calculate and add a p-value to stratified `tbl_survfit()` tables.

Usage

```
## S3 method for class 'tbl_survfit'
add_p(
  x,
  test = "logrank",
  test.args = NULL,
  pvalue_fun = label_style_pvalue(digits = 1),
  include = everything(),
  quiet,
  ...
)
```

Arguments

`x`	(`tbl_survfit`) Object of class `"tbl_survfit"`
`test`	(`string`) string indicating test to use. Must be one of `"logrank"`, `"tarone"`, `"survdiff"`, `"petopeto_gehanwilcoxon"`, `"coxph_lrt"`, `"coxph_wald"`, `"coxph_score"`. See details below
`test.args`	(named `list`) named list of arguments that will be passed to the method specified in the `test` argument. Default is `NULL`.
`pvalue_fun`	(`function`) Function to round and format p-values. Default is `label_style_pvalue()`. The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = label_style_pvalue(digits = 2)`).
`include`	(`tidy-select`) Variables to include in output. Default is `everything()`.
`quiet`	**[Deprecated]**
`...`	These dots are for future extensions and must be empty.

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 in 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))
```

Note

To calculate the p-values, the formula is re-constructed from the the call in the original `survfit()` object. When the `survfit()` object is created a for loop, `lapply()`, `purrr::map()` setting the call *may not* reflect the true formula which may result in an error or an incorrect calculation.

To ensure correct results, the call formula in `survfit()` must represent the formula that will be used in `survival::survdiff()`. If you utilize the `tbl_survfit.data.frame()` S3 method, this is handled for you.

See Also

Other tbl_survfit tools: `add_nevent.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 ----------------------------------
gts_survfit |>
  add_p()

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

---

add_p.tbl_svysummary *Add p-values*

Description

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

Usage

```
## S3 method for class 'tbl_svysummary'
add_p(
  x,
 test = list(all_continuous() ~ "svy.wilcox.test", all_categorical() ~ "svy.chisq.test"),
  pvalue_fun = label_style_pvalue(digits = 1),
  include = everything(),
  test.args = NULL,
  ...
)
```

Arguments

x	(tbl_svysummary) table created with tbl_svysummary()
test	(formula-list-selector) List of formulas specifying statistical tests to perform. Default is list(all_continuous() ~ "svy.wilcox.test", all_categorical() ~ "svy.chisq.test"). See below for details on default tests and ?tests for details on available tests and creating custom tests.
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
include	(tidy-select) Variables to include in output. Default is everything().
test.args	(formula-list-selector) 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).
...	These dots are for future extensions and must be empty.

Value

a gtsummary table of class "tbl_svysummary"

Examples

```
# Example 1 ----------------------------------
# A simple weighted dataset
survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) |>
  tbl_svysummary(by = Survived, include = c(Sex, Age)) |>
  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 ----------------------------------
tbl_svysummary(d_clust, by = both, include = c(api00, api99)) |>
  add_p()

# Example 3 ----------------------------------
# change tests to svy t-test and Wald test
tbl_svysummary(d_clust, by = both, include = c(api00, api99, stype)) |>
  add_p(
    test = list(
      all_continuous() ~ "svy.t.test",
      all_categorical() ~ "svy.wald.test"
    )
  )
```

---

add_q *Add multiple comparison adjustment*

Description

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

Usage

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

Arguments

`x`	(`gtsummary`) a `gtsummary` object with a column named `"p.value"`
`method`	(`string`) String indicating method to be used for p-value adjustment. Methods from `stats::p.adjust()` are accepted. Default is `method='fdr'`. Must be one of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr', 'none'
`pvalue_fun`	(`function`) Function to round and format q-values. Default is the function specified to round the existing `'p.value'` column.
`quiet`	**[Deprecated]**

Author(s)

Daniel D. Sjoberg, Esther Drill

Examples

```
# Example 1 ----------------------------------
add_q_ex1 <-
  trial |>
  tbl_summary(by = trt, include = c(trt, age, grade, response)) |>
  add_p() |>
  add_q()

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

---

add_significance_stars

Add significance stars

Description

Add significance stars to estimates with small p-values

Usage

```
add_significance_stars(
  x,
  pattern = ifelse(inherits(x, c("tbl_regression", "tbl_uvregression")),
    "{estimate}{stars}", "{p.value}{stars}"),
  thresholds = c(0.001, 0.01, 0.05),
  hide_ci = TRUE,
  hide_p = inherits(x, c("tbl_regression", "tbl_uvregression")),
  hide_se = FALSE
)
```

Arguments

x	(gtsummary) A 'gtsummary' object with a 'p.value' column
pattern	(string) glue-syntax string indicating what to display in formatted column. Default is "{estimate}{stars}" for regression summaries and "{p.value}{stars}"

otherwise. A footnote is placed on the first column listed in the pattern. Other common patterns are `"{estimate}{stars} ({conf.low}, {conf.high})"` and `"{estimate} ({conf.low} to {conf.high}){stars}"`

`thresholds` (`numeric`)
Thresholds for significance stars. Default is `c(0.001, 0.01, 0.05)`

`hide_ci` (scalar `logical`)
logical whether to hide confidence interval. Default is `TRUE`

`hide_p` (scalar `logical`)
logical whether to hide p-value. Default is `TRUE` for regression summaries, and `FALSE` otherwise.

`hide_se` (scalar `logical`)
logical whether to hide standard error. Default is `FALSE`

Value

a 'gtsummary' table

Examples

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

# Example 1 ----------------------------------
tbl |>
  add_significance_stars(hide_ci = FALSE, hide_p = FALSE)

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

# Example 3 ----------------------------------
# Use '  \n' to put a line break between beta and SE
tbl |>
  add_significance_stars(
    hide_se = TRUE,
    pattern = "{estimate}{stars}  \n({std.error})"
  ) |>
  modify_header(estimate = "**Beta  \n(SE)**") |>
  modify_abbreviation("SE = Standard Error") |>
  as_gt() |>
  gt::fmt_markdown(columns = everything()) |>
  gt::tab_style(
    style = "vertical-align:top",
    locations = gt::cells_body(columns = label)
```

```
  )

# Example 4 ---------------------------------
lm(marker ~ stage + grade, data = trial) |>
  tbl_regression() |>
  add_global_p() |>
  add_significance_stars(
    hide_p = FALSE,
    pattern = "{p.value}{stars}"
  )
```

`add_stat` *Add a custom statistic*

Description

The function allows a user to add a new column (or columns) of statistics to an existing `tbl_summary`, `tbl_svysummary`, or `tbl_continuous` object.

Usage

```
add_stat(x, fns, location = everything() ~ "label")
```

Arguments

`x`	(`tbl_summary/tbl_svysummary/tbl_continuous`) A gtsummary table of class `'tbl_summary'`, `'tbl_svysummary'`, or `'tbl_continuous'`.
`fns`	(`formula-list-selector`) Indicates the functions that create the statistic. See details below.
`location`	(`formula-list-selector`) Indicates the location the new statistics are placed. The values must be one of `c("label", "level", "missing")`. When `"label"`, a single statistic is placed on the variable label row. When `"level"` the statistics are placed on the variable level rows. The length of the vector of statistics returned from the `fns` function must match the dimension of levels. Default is to place the new statistics on the label row.

Value

A 'gtsummary' of the same class as the input

Details

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

1. Each function must return a tibble or a vector. If a vector is returned, it will be converted to a tibble with one column and number of rows equal to the length of the vector.

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

The user-defined function 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_header()` 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()`.

Examples

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

trial |>
  tbl_summary(
    by = trt,
    include = c(trt, age, marker),
    missing = "no"
  ) |>
  add_stat(fns = everything() ~ my_ttest) |>
  modify_header(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() %>%
    dplyr::mutate(
    stat = glue::glue("t={style_sigfig(statistic)}, {style_pvalue(p.value, prepend_p = TRUE)}")
    ) %>%
    dplyr::pull(stat)
}
```

```
trial |>
  tbl_summary(
    by = trt,
    include = c(trt, age, marker),
    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)
}

trial |>
  tbl_summary(
    by = trt,
    include = c(trt, age, marker),
    missing = "no"
  ) |>
  add_stat(fns = everything() ~ my_ttest3) |>
  modify_header(statistic = "**t-statistic**", p.value = "**p-value**") |>
 modify_fmt_fun(statistic = label_style_sigfig(), p.value = label_style_pvalue(digits = 2))
```

`add_stat_label` *Add statistic labels*

Description

[Questioning]
Adds or modifies labels describing the summary statistics presented for each variable in a `tbl_summary()` table.

Usage

```
add_stat_label(x, ...)

## S3 method for class 'tbl_summary'
add_stat_label(x, location = c("row", "column"), label = NULL, ...)

## S3 method for class 'tbl_svysummary'
add_stat_label(x, location = c("row", "column"), label = NULL, ...)

## S3 method for class 'tbl_ard_summary'
add_stat_label(x, location = c("row", "column"), label = NULL, ...)
```

Arguments

x	(tbl_summary) Object with class 'tbl_summary' or with class 'tbl_svysummary'
...	These dots are for future extensions and must be empty.
location	(string) 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	(formula-list-selector) indicates the updates to the statistic label, e.g. label = all_categorical() ~ "No. (%)". When not specified, the default statistic labels are used.

Value

A tbl_summary or tbl_svysummary object

Tips

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

For example, the following two tables merge properly

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

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

The addition of the new "stat_label" column requires a default labels for categorical variables, which is "No. (%)". This can be changed to either desired text or left blank using NA_character_. The blank option is useful in the location="row" case to keep the output for categorical variables identical what was produced without a "add_stat_label()" function call.

Author(s)

Daniel D. Sjoberg

Examples

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

# Example 1 ---------------------------------
```

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

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

# Example 3 ----------------------------------
trial |>
  select(age, grade, trt) |>
  tbl_summary(
    by = trt,
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c("{median} ({p25}, {p75})", "{min} - {max}"),
  ) |>
  add_stat_label(label = age ~ c("IQR", "Range"))
```

`add_variable_group_header` *Variable Group Header*

Description

Some data are inherently grouped, and should be reported together. Grouped variables are all indented together. This function indents the variables that should be reported together while adding a header above the group.

Usage

```
add_variable_group_header(x, header, variables, indent = 4L)
```

Arguments

`x`	(`tbl_summary`) gtsummary object of class `'tbl_summary'`
`header`	(`string`) string of the header to place above the variable group
`variables`	(`tidy-select`) Variables to group that appear in `x$table_body`. Selected variables should be appear consecutively in table.
`indent`	(`integer`) An integer indicating how many space to indent text. All rows in the group will be indented by this amount. Default is `4`.

Details

This function works by inserting a row into the `x$table_body` and indenting the group of selected variables. This function cannot be used in conjunction with all functions in gtsummary; for example, `bold_labels()` will bold the incorrect rows after running this function.

Value

a gtsummary table

Examples

```
# Example 1 ----------------------------------
set.seed(11234)
data.frame(
  exclusion_age = sample(c(TRUE, FALSE), 20, replace = TRUE),
  exclusion_mets = sample(c(TRUE, FALSE), 20, replace = TRUE),
  exclusion_physician = sample(c(TRUE, FALSE), 20, replace = TRUE)
) |>
  tbl_summary(
    label = list(exclusion_age = "Age",
                 exclusion_mets = "Metastatic Disease",
                 exclusion_physician = "Physician")
  ) |>
  add_variable_group_header(
    header = "Exclusion Reason",
    variables = starts_with("exclusion_")
  ) |>
  modify_caption("**Study Exclusion Criteria**")

# Example 2 ----------------------------------
lm(marker ~ trt + grade + age, data = trial) |>
  tbl_regression() |>
  add_global_p(keep = TRUE, include = grade) |>
  add_variable_group_header(
    header = "Treatment:",
    variables = trt
  ) |>
  add_variable_group_header(
    header = "Covariate:",
    variables = -trt
  ) |>
  # indent levels 8 spaces
  modify_indent(
    columns = "label",
    rows = row_type == "level",
    indent = 8L
  )
```

add_vif *Add Variance Inflation Factor*

Description

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 = label_style_sigfig(digits = 2))
```

Arguments

`x`	`'tbl_regression'` object
`statistic`	`"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 `car::vif()` for details.
`estimate_fun`	Default is `label_style_sigfig(digits = 2)`.

See Also

Review list, formula, and selector syntax used throughout gtsummary

Examples

```
# Example 1 ----------------------------------
lm(age ~ grade + marker, trial) |>
  tbl_regression() |>
  add_vif()

# Example 2 ----------------------------------
lm(age ~ grade + marker, trial) |>
  tbl_regression() |>
  add_vif(c("aGVIF", "df"))
```

assign_summary_digits *Assign Default Digits*

Description

Used to assign the default formatting for variables summarized with `tbl_summary()`.

Usage

```
assign_summary_digits(data, statistic, type, digits = NULL)
```

Arguments

`data`	(`data.frame`) a data frame
`statistic`	(named list) a named list; notably, *not* a `formula-list-selector`
`type`	(named list) a named list; notably, *not* a `formula-list-selector`
`digits`	(named list) a named list; notably, *not* a `formula-list-selector`. Default is `NULL`

Value

a named list

Examples

```
assign_summary_digits(
  mtcars,
  statistic = list(mpg = "{mean}"),
  type = list(mpg = "continuous")
)
```

assign_summary_type *Assign Default Summary Type*

Description

Function inspects data and assigns a summary type when not specified in the `type` argument.

Usage

```
assign_summary_type(data, variables, value, type = NULL, cat_threshold = 10L)
```

Arguments

data	(data.frame) a data frame
variables	(character) character vector of column names in data
value	(named list) named list of values to show for dichotomous variables, where the names are the variables
type	(named list) named list of summary types, where names are the variables
cat_threshold	(integer) for base R numeric classes with fewer levels than this threshold will default to a categorical summary. Default is 10L

Value

named list

Examples

```
assign_summary_type(
  data = trial,
  variables = c("age", "grade", "response"),
  value = NULL
)
```

---

assign_tests *Assign Test*

Description

This function is used to assign default tests for add_p() and add_difference().

Usage

```
assign_tests(x, ...)

## S3 method for class 'tbl_summary'
assign_tests(
  x,
  include,
  by = x$inputs$by,
  test = NULL,
  group = NULL,
  adj.vars = NULL,
  summary_type = x$inputs$type,
```

```
  calling_fun = c("add_p", "add_difference"),
  ...
)

## S3 method for class 'tbl_svysummary'
assign_tests(
  x,
  include,
  by = x$inputs$by,
  test = NULL,
  group = NULL,
  adj.vars = NULL,
  summary_type = x$inputs$type,
  calling_fun = c("add_p", "add_difference"),
  ...
)

## S3 method for class 'tbl_continuous'
assign_tests(x, include, by, cont_variable, test = NULL, group = NULL, ...)

## S3 method for class 'tbl_survfit'
assign_tests(x, include, test = NULL, ...)
```

Arguments

x	(gtsummary) a table of class 'gtsummary'
...	Passed to rlang::abort(), rlang::warn() or rlang::inform().
include	(character) Character vector of column names to assign a default tests.
by	(string) a single stratifying column name
test	(named list) a named list of tests.
group	(string) a variable name indicating the grouping column for correlated data. Default is NULL.
adj.vars	(character) Variables to include in adjusted calculations (e.g. in ANCOVA models).
summary_type	(named list) named list of summary types
calling_fun	(string) Must be one of 'add_p' and 'add_difference'. Depending on the context, different defaults are set.
cont_variable	(string) a column name of the continuous summary variable in tbl_continuous()

Value

A table of class 'gtsummary'

Examples

```
trial |>
  tbl_summary(
    by = trt,
    include = c(age, stage)
  ) |>
  assign_tests(include = c("age", "stage"), calling_fun = "add_p")
```

---

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, ...)
```

Arguments

x	(gtsummary) An object of class "gtsummary"
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.
...	Not used

Details

The as_flex_table() function supports bold and italic markdown syntax in column headers and spanning headers ('**' and '_' only). Text wrapped in double stars ('**bold**') will be made bold, and text between single underscores ('_italic_') will be made italic. No other markdown syntax is supported and the double-star and underscore cannot be combined. To further style your table, you may convert the table to flextable with as_flex_table(), then utilize any of the flextable functions.

Value

A 'flextable' object

Author(s)

Daniel D. Sjoberg

Examples

```
trial |>
  select(trt, age, grade) |>
  tbl_summary(by = trt) |>
  add_p() |>
  as_flex_table()
```

---

as_gt *Convert gtsummary object to gt*

Description

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

Usage

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

Arguments

x	(gtsummary) An object of class "gtsummary"
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(...)

Value

A gt_tbl object

Note

As of 2024-08-15, line breaks (e.g. '\n') do not render properly for PDF output. For now, these line breaks are stripped when rendering to PDF with Quarto and R markdown.

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ---------------------------------
trial |>
  tbl_summary(by = trt, include = c(age, grade, response)) |>
  as_gt()
```

`as_gtsummary` *Create gtsummary table*

Description

This function ingests a data frame and adds the infrastructure around it to make it a gtsummary object.

Usage

```
as_gtsummary(table_body, ...)
```

Arguments

`table_body`	(`data.frame`) a data frame that will be added as the gtsummary object's `table_body`
`...`	other objects that will be added to the gtsummary object list

Details

Function uses `table_body` to create a gtsummary object

Value

gtsummary object

Examples

```
mtcars[1:2, 1:2] |>
  as_gtsummary()
```

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)

as_hux_xlsx(x, file, include = everything(), bold_header_rows = TRUE)
```

Arguments

x	(gtsummary) An object of class "gtsummary"
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.
file	File path for the output.
bold_header_rows	(scalar logical) logical indicating whether to bold header rows. Default is TRUE

Value

A {huxtable} object

Excel Output

Use the as_hux_xlsx() function to save a copy of the table in an excel file. The file is saved using huxtable::quick_xlsx().

Author(s)

David Hugh-Jones, Daniel D. Sjoberg

Examples

```
trial |>
  tbl_summary(by = trt, include = c(age, grade)) |>
  add_p() |>
  as_hux_table()
```

as_kable *Convert gtsummary object to a kable object*

Description

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.

Line breaks (\n) are removed from column headers and table cells.

Usage

```
as_kable(x, ..., include = everything(), return_calls = FALSE)
```

Arguments

x	(gtsummary) Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
...	Additional arguments passed to knitr::kable()
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.

Details

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

Value

A knitr_kable object

Author(s)

Daniel D. Sjoberg

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. This allows the customized formatting available via `knitr::kable()` and {kableExtra}; `as_kable_extra()` supports arguments in `knitr::kable()`. `as_kable_extra()` output via gtsummary supports bold and italic cells for table bodies. Users are encouraged to leverage `as_kable_extra()` for enhanced pdf printing; for html output options there is better support via `as_gt()`.

Usage

```
as_kable_extra(
  x,
  escape = FALSE,
  format = NULL,
  ...,
  include = everything(),
  addtl_fmt = TRUE,
  return_calls = FALSE
)
```

Arguments

`x`	(`gtsummary`) Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
`format, escape, ...`	arguments passed to `knitr::kable()`. Default is `escape = FALSE`, and the format is auto-detected.
`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()`.
`addtl_fmt`	logical indicating whether to include additional formatting. Default is `TRUE`. This is primarily used to escape special characters, convert markdown to LaTeX, and remove line breaks from the footnote.
`return_calls`	Logical. Default is `FALSE`. If `TRUE`, the calls are returned as a list of expressions.

Value

A {kableExtra} table

PDF/LaTeX

This section shows options intended for use with `output: pdf_document` in yaml of `.Rmd`.

When the default values of `as_kable_extra(escape = FALSE, addtl_fmt = TRUE)` are utilized, the following formatting occurs.

- Markdown bold, italic, and underline syntax in the headers, spanning headers, caption, and footnote will be converted to escaped LaTeX code
- Special characters in the table body, headers, spanning headers, caption, and footnote will be escaped with `.escape_latex()` or `.escape_latex2()`
- The `"\n"` symbol will be recognized as a line break in the table headers, spanning headers, caption, and the table body
- The `"\n"` symbol is removed from the footnotes

To suppress *these* additional formats, set `as_kable_extra(addtl_fmt = FALSE)`

Additional styling is available with `kableExtra::kable_styling()` as shown in Example 2, which implements row striping and repeated column headers in the presence of page breaks.

HTML

This section discusses options intended for use with `output: html_document` in yaml of `.Rmd`.

When the default values of `as_kable_extra(escape = FALSE, addtl_fmt = TRUE)` are utilized, the following formatting occurs.

- The default markdown syntax in the headers and spanning headers is removed
- Special characters in the table body, headers, spanning headers, caption, and footnote will be escaped with `.escape_html()`
- The `"\n"` symbol is removed from the footnotes

To suppress the additional formatting, set `as_kable_extra(addtl_fmt = FALSE)`

Author(s)

Daniel D. Sjoberg

Examples

```
# basic gtsummary tbl to build upon
as_kable_extra_base <-
  trial |>
  tbl_summary(by = trt, include = c(age, stage)) |>
  bold_labels()

# Example 1 (PDF via LaTeX) ---------------------
# add linebreak in table header with '\n'
as_kable_extra_ex1_pdf <-
  as_kable_extra_base |>
  modify_header(all_stat_cols() ~ "**{level}**  \n*N = {n}*") |>
  as_kable_extra()
```

```
# Example 2 (PDF via LaTeX) ---------------------
# additional styling in `knitr::kable()` and with
#   call to `kableExtra::kable_styling()`
as_kable_extra_ex2_pdf <-
  as_kable_extra_base |>
  as_kable_extra(
    booktabs = TRUE,
    longtable = TRUE,
    linesep = ""
  ) |>
  kableExtra::kable_styling(
    position = "left",
    latex_options = c("striped", "repeat_header"),
    stripe_color = "gray!15"
  )
```

`as_tibble.gtsummary` *Convert gtsummary object to a tibble*

Description

Function converts a gtsummary object to a tibble.

Usage

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

## S3 method for class 'gtsummary'
as.data.frame(...)
```

Arguments

x	(gtsummary) An object of class "gtsummary"
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	(scalar logical) 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.
fmt_missing	(scalar logical) Logical argument adding the missing value formats.
...	Arguments passed on to gt::gt(...)

Value

a tibble

Author(s)

Daniel D. Sjoberg

Examples

```
tbl <-
  trial |>
  tbl_summary(by = trt, include = c(age, grade, response))

as_tibble(tbl)

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

---

bold_italicize_labels_levels

Bold or Italicize

Description

Bold or italicize labels or levels in gtsummary tables

Usage

```
bold_labels(x)

italicize_labels(x)

bold_levels(x)

italicize_levels(x)

## S3 method for class 'gtsummary'
bold_labels(x)

## S3 method for class 'gtsummary'
bold_levels(x)
```

```
## S3 method for class 'gtsummary'
italicize_labels(x)

## S3 method for class 'gtsummary'
italicize_levels(x)

## S3 method for class 'tbl_cross'
bold_labels(x)

## S3 method for class 'tbl_cross'
bold_levels(x)

## S3 method for class 'tbl_cross'
italicize_labels(x)

## S3 method for class 'tbl_cross'
italicize_levels(x)
```

Arguments

x	(gtsummary) An object of class 'gtsummary'

Value

Functions return the same class of gtsummary object supplied

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
tbl_summary(trial, include = c("trt", "age", "response")) |>
  bold_labels() |>
  bold_levels() |>
  italicize_labels() |>
  italicize_levels()
```

---

bold_p *Bold significant p-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	(gtsummary) Object created using gtsummary functions
t	(scalar numeric) Threshold below which values will be bold. Default is 0.05.
q	(scalar logical) When TRUE will bold the q-value column rather than the p-value. Default is FALSE.

Author(s)

Daniel D. Sjoberg, Esther Drill

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(by = trt, include = c(response, marker, trt), missing = "no") |>
  add_p() |>
  bold_p(t = 0.1)

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

---

brdg_continuous *Continuous Summary Table Bridges*

Description

Bridge function for converting tbl_continuous() cards to basic gtsummary objects. This bridge function converts the 'cards' object to a format suitable to pass to brdg_summary(): no pier_*() functions required.

Usage

```
brdg_continuous(cards, by = NULL, statistic, include, variable, type)
```

Arguments

cards	(card) An ARD object of class "card" typically created with cards::ard_*() functions.
by	(string) string indicating the stratifying column

statistic	(named list) named list of summary statistic names
include	(tidy-select) Variables to include in the summary table. Default is everything().
variable	(tidy-select) A single column from data. Variable name of the continuous column to be summarized.
type	(named list) named list of summary types

Value

a gtsummary object

Examples

```
library(cards)

bind_ard(
  # the primary ARD with the results
  ard_summary(trial, by = grade, variables = age),
  # add missing and attributes ARD
  ard_missing(trial, by = grade, variables = age),
  ard_attributes(trial, variables = c(grade, age))
) |>
  # adding the column name
  dplyr::mutate(
    gts_column =
      ifelse(!context %in% "attributes", "stat_0", NA_character_)
  ) |>
  brdg_continuous(
    variable = "age",
    include = "grade",
    statistic = list(grade = "{median} ({p25}, {p75})"),
    type = list(grade = "categorical")
 ) |>
 as_tibble()
```

---

brdg_hierarchical *Hierarchy table bridge*

Description

Bridge function for converting tbl_hierarchical() (and similar) cards to basic gtsummary objects. All bridge functions begin with prefix brdg_*().

This file also contains helper functions for constructing the bridge, referred to as the piers (supports for a bridge) and begin with pier_*().

- `brdg_hierarchical()`: The bridge function ingests an ARD data frame and returns a gtsummary table that includes `.$table_body` and a basic `.$table_styling`. The `.$table_styling$header` data frame includes the header statistics. Based on context, this function adds a column to the ARD data frame named `"gts_column"`. This column is used during the reshaping in the `pier_*()` functions defining column names.
- `pier_*()`: these functions accept a cards tibble and returns a tibble that is a piece of the `.$table_body`. Typically these will be stacked to construct the final table body data frame. The ARD object passed here will have two primary parts: the calculated summary statistics and the attributes ARD. The attributes ARD is used for labeling. The ARD data frame passed to this function must include a `"gts_column"` column, which is added in `brdg_hierarchical()`.

Usage

```
brdg_hierarchical(
  cards,
  variables,
  by,
  include,
  statistic,
  overall_row,
  count,
  is_ordered,
  label
)

pier_summary_hierarchical(cards, variables, include, statistic)
```

Arguments

`cards`	(`card`) an ARD object of class `"card"` created with `cards::ard_hierarchical_stack()`.
`variables`	(`character`) character list of hierarchy variables.
`by`	(`string`) string indicating the stratifying column.
`include`	(`character`) character list of hierarchy variables to include summary statistics for.
`statistic`	(named `list`) named list of summary statistic names.
`overall_row`	(scalar `logical`) whether an overall summary row should be included at the top of the table. The default is `FALSE`.
`count`	(scalar `logical`) whether `tbl_hierarchical_count()` (`TRUE`) or `tbl_hierarchical()` (`FALSE`) is being applied.
`is_ordered`	(scalar `logical`) whether the last variable in `variables` is ordered.

`label` (named `list`)
named list of hierarchy variable labels.

Value

a gtsummary object

See Also

Review list, formula, and selector syntax used throughout gtsummary

---

`brdg_summary` *Summary table bridge*

Description

Bridge function for converting `tbl_summary()` (and similar) cards to basic gtsummary objects. All bridge functions begin with prefix `brdg_*()`.

This file also contains helper functions for constructing the bridge, referred to as the piers (supports for a bridge) and begin with `pier_*()`.

- `brdg_summary()`: The bridge function ingests an ARD data frame and returns a gtsummary table that includes `.$table_body` and a basic `.$table_styling`. The `.$table_styling$header` data frame includes the header statistics. Based on context, this function adds a column to the ARD data frame named `"gts_column"`. This column is used during the reshaping in the `pier_*()` functions defining column names.
- `pier_*()`: these functions accept a cards tibble and returns a tibble that is a piece of the `.$table_body`. Typically these will be stacked to construct the final table body data frame. The ARD object passed here will have two primary parts: the calculated summary statistics and the attributes ARD. The attributes ARD is used for labeling. The ARD data frame passed to this function must include a `"gts_column"` column, which is added in `brdg_summary()`.

Usage

```
brdg_summary(
  cards,
  variables,
  type,
  statistic,
  by = NULL,
  missing = "no",
  missing_stat = "{N_miss}",
  missing_text = "Unknown"
)

pier_summary_dichotomous(cards, variables, statistic)
```

```
pier_summary_categorical(cards, variables, statistic)

pier_summary_continuous2(cards, variables, statistic)

pier_summary_continuous(cards, variables, statistic)

pier_summary_missing_row(
  cards,
  variables,
  missing = "no",
  missing_stat = "{N_miss}",
  missing_text = "Unknown"
)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.
`variables`	(`character`) character list of variables
`type`	(named `list`) named list of summary types
`statistic`	(named `list`) named list of summary statistic names
`by`	(`string`) string indicating the stratifying column
`missing`, `missing_text`, `missing_stat`	Arguments dictating how and if missing values are presented: • `missing`: must be one of `c("ifany", "no", "always")`. • `missing_text`: string indicating text shown on missing row. Default is `"Unknown"`. • `missing_stat`: statistic to show on missing row. Default is `"{N_miss}"`. Possible values are `N_miss`, `N_obs`, `N_nonmiss`, `p_miss`, `p_nonmiss`.

Value

a gtsummary object

Examples

```
library(cards)

# first build ARD data frame
cards <-
  ard_stack(
    mtcars,
    ard_summary(variables = c("mpg", "hp")),
```

```
    ard_tabulate(variables = "cyl"),
    ard_tabulate_value(variables = "am"),
    .missing = TRUE,
    .attributes = TRUE
  ) |>
  # this column is used by the `pier_*()` functions
  dplyr::mutate(gts_column = ifelse(context == "attributes", NA, "stat_0"))

brdg_summary(
  cards = cards,
  variables = c("cyl", "am", "mpg", "hp"),
  type =
    list(
      cyl = "categorical",
      am = "dichotomous",
      mpg = "continuous",
      hp = "continuous2"
    ),
  statistic =
    list(
      cyl = "{n} / {N}",
      am = "{n} / {N}",
      mpg = "{mean} ({sd})",
      hp = c("{median} ({p25}, {p75})", "{mean} ({sd})")
    )
) |>
  as_tibble()

pier_summary_dichotomous(
  cards = cards,
  variables = "am",
  statistic = list(am = "{n} ({p})")
)

pier_summary_categorical(
  cards = cards,
  variables = "cyl",
  statistic = list(cyl = "{n} ({p})")
)

pier_summary_continuous2(
  cards = cards,
  variables = "hp",
  statistic = list(hp = c("{median}", "{mean}"))
)

pier_summary_continuous(
  cards = cards,
  variables = "mpg",
  statistic = list(mpg = "{median}")
)
```

`brdg_wide_summary` *Wide summary table bridge*

Description

Bridge function for converting `tbl_wide_summary()` (and similar) cards to basic gtsummary objects. All bridge functions begin with prefix `brdg_*()`.

Usage

```
brdg_wide_summary(cards, variables, statistic, type)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.
`variables`	(`character`) character list of variables
`statistic`	(named `list`) named list of summary statistic names
`type`	(named `list`) named list of summary types

Value

a gtsummary object

Examples

```
library(cards)

bind_ard(
  ard_summary(trial, variables = c(age, marker)),
  ard_attributes(trial, variables = c(age, marker))
) |>
  brdg_wide_summary(
    variables = c("age", "marker"),
    statistic = list(age = c("{mean}", "{sd}"), marker = c("{mean}", "{sd}")),
    type = list(age = "continuous", marker = "continuous")
  )
```

combine_terms *Combine terms*

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, ...)
```

Arguments

x	(tbl_regression) A tbl_regression object
formula_update	(formula) formula update passed to the `stats::update()`. This updated formula is used to construct a reduced model, and is subsequently passed to `stats::anova()` to calculate the p-value for the group of removed terms. See the `stats::update()` function's `formula.=` argument for proper syntax.
label	(string) Optional string argument labeling the combined rows
quiet	**[Deprecated]**
...	Additional arguments passed to stats::anova

Value

tbl_regression object

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
# Logistic Regression Example, LRT p-value
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 *Custom tidiers*

Description

Collection of tidiers that can be utilized in gtsummary. See details below.

Usage

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

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

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

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

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

tidy_wald_test(x, tidy_fun = NULL, vcov = stats::vcov(x), ...)
```

Arguments

x	(model) Regression model object

exponentiate	(scalar logical) Logical indicating whether to exponentiate the coefficient estimates. Default is FALSE.
conf.level	(scalar real) Confidence level for confidence interval/credible interval. Defaults to 0.95.
conf.int	(scalar logical) Logical indicating whether or not to include a confidence interval in the output. Default is TRUE.
...	Arguments passed to method; • pool_and_tidy_mice(): mice::tidy(x, ...) • tidy_standardize(): parameters::standardize_parameters(x, ...) • tidy_bootstrap(): parameters::bootstrap_parameters(x, ...) • tidy_robust(): parameters::model_parameters(x, ...)
quiet	**[Deprecated]**
vcov, vcov_args	• tidy_robust(): Arguments passed to parameters::model_parameters(). At least one of these arguments **must** be specified. • tidy_wald_test(): vcov is the covariance matrix of the model with default stats::vcov().
pool.args	(named list) Named list of arguments passed to mice::pool() in pool_and_tidy_mice(). Default is NULL
tidy_fun	(function) Tidier function for the model. Default is to use broom::tidy(). If an error occurs, the tidying of the model is attempted with parameters::model_parameters(), if installed.

Regression Model Tidiers

These tidiers are passed to tbl_regression() and tbl_uvregression() to obtain modified results.

- tidy_standardize() tidier to report standardized coefficients. The parameters package includes a wonderful function to estimate standardized coefficients. The tidier uses the output from parameters::standardize_parameters(), and merely takes the result and puts it in broom::tidy() format.
- tidy_bootstrap() tidier to report bootstrapped coefficients. The parameters package includes a wonderful function to estimate bootstrapped coefficients. The tidier uses the output from parameters::bootstrap_parameters(test = "p"), and merely takes the result and puts it in broom::tidy() format.
- tidy_robust() tidier to report robust standard errors, confidence intervals, and p-values. The parameters package includes a wonderful function to calculate robust standard errors, confidence intervals, and p-values The tidier uses the output from parameters::model_parameters(), and merely takes the result and puts it in broom::tidy() format. To use this function with tbl_regression(), pass a function with the arguments for tidy_robust() populated.
- 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.

Other Tidiers

- `tidy_wald_test()` tidier to report Wald p-values, wrapping the `aod::wald.test()` function. Use this tidier with `add_global_p(anova_fun = tidy_wald_test)`

Examples

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

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

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

# Example 2 ----------------------------------
# use "posthoc" method for coef calculation
tbl_regression(mod, tidy_fun = \(x, ...) tidy_standardize(x, method = "posthoc", ...))

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

---

filter_hierarchical *Filter Hierarchical Tables*

Description

[Experimental]

This function is used to filter hierarchical table rows. Filters are not applied to summary or overall rows.

Usage

```
filter_hierarchical(x, ...)

## S3 method for class 'tbl_hierarchical'
filter_hierarchical(
  x,
```

```
  filter,
  var = NULL,
  keep_empty = FALSE,
  quiet = FALSE,
  ...
)

## S3 method for class 'tbl_hierarchical_count'
filter_hierarchical(
  x,
  filter,
  var = NULL,
  keep_empty = FALSE,
  quiet = FALSE,
  ...
)

## S3 method for class 'tbl_ard_hierarchical'
filter_hierarchical(
  x,
  filter,
  var = NULL,
  keep_empty = FALSE,
  quiet = FALSE,
  ...
)
```

Arguments

x	(tbl_hierarchical, tbl_hierarchical_count, tbl_ard_hierarchical) A hierarchical gtsummary table of class 'tbl_hierarchical', 'tbl_hierarchical_count', or 'tbl_ard_hierarchical'.
...	These dots are for future extensions and must be empty.
filter	(expression) An expression that is used to filter rows of the table. See the Details section below.
var	(tidy-select) Hierarchy variable from x to perform filtering on. The variable must be present in x$inputs$include. If NULL, the last hierarchy variable from x (dplyr::last(x$inputs$include)) will be used.
keep_empty	(scalar logical) Logical argument indicating whether to retain summary rows corresponding to table hierarchy sections that have had all rows filtered out. Default is FALSE.
quiet	(logical) Logical indicating whether to suppress any messaging. Default is FALSE.

Details

The `filter` argument can be used to filter out rows of a table which do not meet the criteria provided as an expression. Rows can be filtered on the values of any of the possible statistics (`n`, `p`, and `N`) provided they are included at least once in the table, as well as the values of any by variables.

Additionally, filters can be applied on individual column values (if a by variable was specified) via the `n_XX`, `N_XX`, and `p_XX` statistics, where each `XX` represents the index of the column to select the statistic from. For example, `filter = n_1 > 5` will check whether `n` values in the first column of the table are greater than 5 in each row.

Overall statistics for each row can be used in filters via the `n_overall`, `N_overall`, and `p_overall` statistics. If used in `filter`, overall statistics are derived within the filtering function. `n_overall` can only be derived if `n` statistic is present in the table for the filter variable, `N_overall` if the `N` statistic is present for the filter variable, and `p_overall` if both the `n` and `N` statistics are present for the filter variable.

By default, filters will be applied at the level of the innermost hierarchy variable, i.e. the last variable supplied to `variables`. If filters should instead be applied at the level of one of the outer hierarchy variables, the `var` parameter can be used to specify a different variable to filter on. When `var` is set to a different (outer) variable and a level of the variable does not meet the filtering criteria then the section corresponding to that variable level - including summary rows - and all sub-sections within that section will be removed.

If an overall column was added to the table (via `add_overall()`) this column will not be used in any filters (i.e. `n_overall` will not include the overall `n` in a given row).

Some examples of possible filters:

- `filter = n > 5`: keep rows where one of the treatment groups observed more than 5 AEs
- `filter = n == 2 & p < 0.05`: keep rows where one of the treatment groups observed exactly 2 AEs and one of the treatment groups observed a proportion less than 5%.
- `filter = n_overall >= 4`: keep rows where there were 4 or more AEs observed across the row
- `filter = mean(n) > 4 | n > 3`: keep rows where the mean number of AEs is 4 or more across the row or one of the treatment groups observed more than 3 AEs
- `filter = n_2 > 2`: keep rows where the `"Xanomeline High Dose"` treatment group observed more than 2 AEs

Value

a gtsummary table of the same class as `x`.

See Also

`sort_hierarchical()`

Examples

```
ADAE_subset <- cards::ADAE |>
  dplyr::filter(AEBODSYS %in% c("SKIN AND SUBCUTANEOUS TISSUE DISORDERS",
                                "EAR AND LABYRINTH DISORDERS")) |>
```

```
  dplyr::filter(.by = AEBODSYS, dplyr::row_number() < 20)

tbl <-
  tbl_hierarchical(
    data = ADAE_subset,
    variables = c(AEBODSYS, AEDECOD),
    by = TRTA,
    denominator = cards::ADSL,
    id = USUBJID,
    overall_row = TRUE
  )

# Example 1 ----------------------------------
# Keep rows where less than 2 AEs are observed across the row
filter_hierarchical(tbl, sum(n) < 2)

# Example 2 ----------------------------------
# Keep rows where at least one treatment group in the row has at least 2 AEs observed
filter_hierarchical(tbl, n >= 2)

# Example 3 ----------------------------------
# Keep rows where AEs across the row have an overall prevalence of greater than 0.5%
filter_hierarchical(tbl, p_overall > 0.005)

# Example 4 ----------------------------------
# Keep rows where SOCs across the row have an overall prevalence of greater than 20
filter_hierarchical(tbl, n_overall > 20, var = AEBODSYS)

# Example 5 ----------------------------------
# Keep AEs that have a difference in prevalence of greater than 3% between reference group with
# `TRTA = "Xanomeline High Dose"` and comparison group with `TRTA = "Xanomeline Low Dose"`
filter_hierarchical(tbl, abs(p_2 - p_3) > 0.03)
```

`gather_ard` *Extract ARDs*

Description

Extract the ARDs from a gtsummary table. If needed, results may be combined with `cards::bind_ard()`.

Usage

```
gather_ard(x)
```

Arguments

`x`	(`gtsummary`) a gtsummary table.

Value

list

Examples

```
tbl_summary(trial, by = trt, include = age) |>
  add_overall() |>
  add_p() |>
  gather_ard()

glm(response ~ trt, data = trial, family = binomial()) |>
  tbl_regression() |>
  gather_ard()
```

---

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) gtsummary object
variable	(tidy-select) A single variable name of statistic to present
level	(string) Level of the variable to display for categorical variables. Default is NULL
column	(tidy-select) Column name to return from x$table_body.
pattern	(string) String indicating the statistics to return. Uses glue::glue() formatting. Default is NULL
...	These dots are for future extensions and must be empty.

Value

A string

column + pattern

Some gtsummary tables report multiple statistics in a single cell, e.g. `"{mean} ({sd})"` in `tbl_summary()` or `tbl_svysummary()`. 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()`. *This is not supported for all tables.*

---

inline_text.tbl_continuous

Report statistics from summary tables inline

Description

Extracts and returns statistics from a `tbl_continuous()` object for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

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

Arguments

`x`	(`tbl_continuous`) Object created from `tbl_continuous()`
`variable`	(`tidy-select`) A single variable name of statistic to present
`column`	(`tidy-select`) Column name to return from `x$table_body`. Can also pass the level of a by variable.
`level`	(`string`) Level of the variable to display for categorical variables. Default is `NULL`
`pattern`	(`string`) String indicating the statistics to return. Uses `glue::glue()` formatting. Default is `NULL`

pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
...	These dots are for future extensions and must be empty.

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

Examples

```
t1 <- trial |>
  tbl_summary(by = trt, include = grade) |>
  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_cross *Report statistics from cross table inline*

Description

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,
  row_level = NULL,
  pvalue_fun = label_style_pvalue(prepend_p = TRUE),
  ...
)
```

Arguments

x	(tbl_cross) A tbl_cross object
col_level	(string) Level of the column variable to display. Can also specify "p.value" for the p-value and "stat_0" for Total column.
row_level	(string) Level of the row variable to display.
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
...	These dots are for future extensions and must be empty.

Value

A string reporting results from a gtsummary table

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 = x$inputs$estimate_fun,
  pvalue_fun = label_style_pvalue(prepend_p = TRUE),
  ...
)
```

Arguments

x	(tbl_regression) Object created by tbl_regression()
variable	(tidy-select) A single variable name of statistic to present
level	(string) Level of the variable to display for categorical variables. Default is NULL
pattern	(string) 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$table_body are available to print as well as the confidence level (conf.level). See below for details.
estimate_fun	(function) Function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is x$inputs$estimate_fun
pvalue_fun	function to style p-values and/or q-values. Default is label_style_pvalue(prepend_p = TRUE)
...	These dots are for future extensions and must be empty.

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

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 = label_style_pvalue(prepend_p = TRUE),
  ...
)

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

Arguments

`x`	(`tbl_summary`) Object created from `tbl_summary()` or `tbl_svysummary()`

variable	(tidy-select) A single variable name of statistic to present
column	(tidy-select) Column name to return from x$table_body. Can also pass the level of a by variable.
level	(string) Level of the variable to display for categorical variables. Default is NULL
pattern	(string) String indicating the statistics to return. Uses glue::glue() formatting. Default is NULL
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
...	These dots are for future extensions and must be empty.

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

Examples

```
t1 <- trial |>
  tbl_summary(by = trt, include = grade) |>
  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

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 = label_style_pvalue(prepend_p = TRUE),
  ...
)
```

Arguments

x	(tbl_survfit) Object created from tbl_survfit()
variable	(tidy-select) Variable name of statistic to present.
level	(string) Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is NULL
pattern	(string) String indicating the statistics to return.
time, prob	(numeric scalar) time or probability for which to return result
column	(tidy-select) column to print from x$table_body. Columns may be selected with time or prob arguments as well.
estimate_fun	(function) Function to round and format estimate and confidence limits. Default is the same function used in tbl_survfit()
pvalue_fun	(function) Function to round and format p-values. Default is label_style_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 2)).
...	These dots are for future extensions and must be empty.

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

Examples

```
library(survival)

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

# sumarize 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, time = 24, level = "Drug B",
            pattern = "{estimate} [95% CI {conf.low}, {conf.high}]")
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 = x$inputs$estimate_fun,
  pvalue_fun = label_style_pvalue(prepend_p = TRUE),
  ...
)
```

Arguments

x	(tbl_uvregression) Object created by tbl_uvregression()
variable	(tidy-select) A single variable name of statistic to present
level	(string) Level of the variable to display for categorical variables. Default is NULL
pattern	(string) String indicating the statistics to return. Uses glue::glue() formatting. Default is NULL
estimate_fun	(function) 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 label_style_pvalue(prepend_p = TRUE)
...	These dots are for future extensions and must be empty.

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

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")
```

---

label_style *Style Functions*

Description

Similar to the `style_*()` family of functions, but these functions return a `style_*()` **function** rather than performing the styling.

Usage

```
label_style_number(
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  scale = 1,
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)

label_style_sigfig(
  digits = 2,
  scale = 1,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)

label_style_pvalue(
```

```
  digits = 1,
  prepend_p = FALSE,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  na = NA_character_,
  ...
)

label_style_ratio(
  digits = 2,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)

label_style_percent(
  prefix = "",
  suffix = "",
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  na = NA_character_,
  ...
)
```

Arguments

digits, big.mark, decimal.mark, scale, prepend_p, prefix, suffix, na, ...
: arguments passed to the style_*() functions

Value

a function

See Also

Other style tools: style_sigfig()

Examples

```
my_style <- label_style_number(digits = 1)
my_style(3.14)
```

modify *Modify column headers and spanning headers*

Description

These functions assist with modifying the aesthetics/style of a table.

- `modify_header()` update column headers
- `modify_spanning_header()` update/add spanning headers

The functions often require users to know the underlying column names. Run `show_header_names()` to print the column names to the console.

Usage

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

modify_spanning_header(
  x,
  ...,
  text_interpret = c("md", "html"),
  level = 1L,
  quiet,
  update
)

remove_spanning_header(x, columns = everything(), level = 1L)

show_header_names(x, show_hidden = FALSE, include_example, quiet)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`...`	dynamic-dots Used to assign updates to headers and spanning headers. Use `modify_*(colname='new header')` to update a single column. Using a formula will invoke tidyselect, e.g. `modify_*(all_stat_cols() ~ "**{level}**")`. The dynamic dots allow syntax like `modify_header(x, !!!list(label = "Variable"))`. See examples below. Use the `show_header_names()` to see the column names that can be modified.
`text_interpret`	(`string`) String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`. Applies to tables printed with `{gt}`.
`update, quiet`	**[Deprecated]**

level (integer)
An integer specifying which level to place the spanning header.

columns (tidy-select)
Columns from which to remove spanning headers.

show_hidden (scalar logical)
Logical indicating whether to print hidden columns as well as printed columns. Default is FALSE.

include_example
[Deprecated]

Value

Updated gtsummary object

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

When assigning column headers and spanning headers, 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).

Author(s)

Daniel D. Sjoberg

Examples

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

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

# Example 1 ----------------------------------
# updating column headers
tbl |>
  modify_header(label = "**Variable**", p.value = "**P**")

# Example 2 ----------------------------------
# updating headers add spanning header
tbl |>
```

```
  modify_header(all_stat_cols() ~ "**{level}**, N = {n} ({style_percent(p)}%)") |>
  modify_spanning_header(all_stat_cols() ~ "**Treatment Received**")
```

`modify_abbreviation` *Modify Abbreviations*

Description

All abbreviations will be coalesced when printing the final table into a single specialized source note.

Usage

```
modify_abbreviation(x, abbreviation, text_interpret = c("md", "html"))

remove_abbreviation(x, abbreviation = NULL)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`abbreviation`	(`string`) a string. In `remove_abbreviation()`, the default value is `NULL`, which will remove all abbreviation source notes.
`text_interpret`	(`string`) String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`. Applies to tables printed with `{gt}`.

Value

Updated gtsummary object

See Also

Footnotes vs Source Notes vs Abbreviations

Examples

```
# Example 1 ----------------------------------
tbl_summary(
  trial,
  by = trt,
  include = age,
  type = age ~ "continuous2"
) |>
  modify_table_body(~dplyr::mutate(.x, label = sub("Q1, Q3", "IQR", x = label))) |>
  modify_abbreviation("IQR = Interquartile Range")
```

```
# Example 2 ---------------------------------
lm(marker ~ trt, trial) |>
  tbl_regression() |>
  remove_abbreviation("CI = Confidence Interval")
```

`modify_bold_italic` *Modify Bold and Italic*

Description

Add or remove bold and italic styling to a cell in a table. By default, the remove functions will remove all bold/italic styling.

Usage

```
modify_bold(x, columns, rows)

remove_bold(x, columns = everything(), rows = TRUE)

modify_italic(x, columns, rows)

remove_italic(x, columns = everything(), rows = TRUE)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`columns`	(`tidy-select`) Selector of columns in `x$table_body`
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Review rows argument details.

Value

Updated gtsummary object

Examples

```
# Example 1 ---------------------------------
tbl <- trial |>
  tbl_summary(include = grade) |>
  modify_bold(columns = label, rows = row_type == "label") |>
  modify_italic(columns = label, rows = row_type == "level")
tbl
```

```
# Example 2 ---------------------------------
tbl |>
  remove_bold(columns = label, rows = row_type == "label") |>
  remove_italic(columns = label, rows = row_type == "level")
```

`modify_caption` *Modify table caption*

Description

Captions are assigned based on output type.

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

Usage

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

Arguments

`x`	(`gtsummary`) A gtsummary object
`caption`	(`string/character`) A string for the table caption/title. NOTE: The `gt` print engine supports a vector of captions. But not every print engine supports this feature, and for those outputs, only a string is accepted.
`text_interpret`	(`string`) String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`. Applies to tables printed with `{gt}`.

Value

Updated gtsummary object

Examples

```
trial |>
  tbl_summary(by = trt, include = c(marker, stage)) |>
  modify_caption(caption = "**Baseline Characteristics** N = {N}")
```

modify_column_alignment *Modify column alignment*

Description

Update column alignment/justification in a gtsummary table.

Usage

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

Arguments

x	(gtsummary) gtsummary object
columns	(tidy-select) Selector of columns in x$table_body
align	(string) String indicating alignment of column, must be one of c("left", "right", "center")

Examples

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

modify_column_hide *Modify hidden columns*

Description

Use these functions to hide or unhide columns in a gtsummary table. Use show_header_names(show_hidden=TRUE) to print available columns to update.

Usage

```
modify_column_hide(x, columns)

modify_column_unhide(x, columns)
```

Arguments

x	(gtsummary) gtsummary object
columns	(tidy-select) Selector of columns in x$table_body

Author(s)

Daniel D. Sjoberg

Examples

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

---

modify_column_merge *Modify Column Merging*

Description

Merge two or more columns in a gtsummary table. Use show_header_names() to print underlying column names.

Usage

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

remove_column_merge(x, columns = everything())
```

Arguments

x	(gtsummary) A gtsummary object
pattern	(string) glue syntax string indicating how to merge columns in x$table_body. For example, to construct a confidence interval use "{conf.low}, {conf.high}".
rows	(predicate expression) Predicate expression to select rows in x$table_body. Review rows argument details.
columns	(tidy-select) Selector of columns in x$table_body

Value

gtsummary table

Details

1. Calling this function merely records the instructions to merge columns. The actual merging occurs when the gtsummary table is printed or converted with a function like `as_gt()`.
2. Because the column merging is delayed, it is recommended to perform major modifications to the table, such as those with `tbl_merge()` and `tbl_stack()`, before assigning merging instructions. Otherwise, unexpected formatting may occur in the final table.
3. If this functionality is used in conjunction with `tbl_stack()` (which includes `tbl_uvregression()`), there may be potential issues with printing. When columns are stack AND when the column-merging is defined with a quosure, you may run into issues due to the loss of the environment when 2 or more quosures are combined. If the expression version of the quosure is the same as the quosure (i.e. no evaluated objects), there should be no issues.

This function is used internally with care, and **it is *not* recommended for users**.

Future Updates

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

See Also

Other Advanced modifiers: `modify_indent()`, `modify_table_styling()`

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(by = trt, missing = "no", include = c(age, marker, trt)) |>
  add_p(all_continuous() ~ "t.test", pvalue_fun = label_style_pvalue(prepend_p = TRUE)) |>
  modify_fmt_fun(statistic ~ label_style_sigfig()) |>
  modify_column_merge(pattern = "t = {statistic}; {p.value}") |>
  modify_header(statistic = "**t-test**")

# Example 2 ----------------------------------
lm(marker ~ age + grade, trial) |>
  tbl_regression() |>
  modify_column_merge(
    pattern = "{estimate} ({conf.low}, {conf.high})",
    rows = !is.na(estimate)
  )
```

`modify_fmt_fun` *Modify formatting functions*

Description

Use this function to update the way numeric columns and rows of `.$table_body` are formatted

Usage

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

Arguments

`x`	(`gtsummary`) A gtsummary object
`...`	dynamic-dots Used to assign updates to formatting functions. Use `modify_fmt_fun(colname = <fmt fun>)` to update a single column. Using a formula will invoke tidyselect, e.g. `modify_fmt_fun(c(estimate, conf.low, conf.high) ~ <fm` Use the `show_header_names()` to see the column names that can be modified.
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Can be used to style footnote, formatting functions, missing symbols, and text formatting. Default is `NULL`. See details below.
`update, quiet`	**[Deprecated]**

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`.

Examples

```
# Example 1 ----------------------------------
# show 'grade' p-values to 3 decimal places and estimates to 4 sig figs
lm(age ~ marker + grade, trial) |>
  tbl_regression() %>%
```

```
  modify_fmt_fun(
    p.value = label_style_pvalue(digits = 3),
    c(estimate, conf.low, conf.high) ~ label_style_sigfig(digits = 4),
    rows = variable == "grade"
  )
```

---

modify_footnote2 *Modify Footnotes*

Description

Modify Footnotes

Usage

```
modify_footnote_header(
  x,
  footnote,
  columns,
  replace = TRUE,
  text_interpret = c("md", "html")
)

modify_footnote_body(
  x,
  footnote,
  columns,
  rows,
  replace = TRUE,
  text_interpret = c("md", "html")
)

modify_footnote_spanning_header(
  x,
  footnote,
  columns,
  level = 1L,
  replace = TRUE,
  text_interpret = c("md", "html")
)

remove_footnote_header(x, columns = everything())

remove_footnote_body(x, columns = everything(), rows = TRUE)

remove_footnote_spanning_header(x, columns = everything(), level = 1L)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`footnote`	(`string`) a string
`columns`	(`tidy-select`) columns to add footnote. For `modify_footnote_spanning_header()`, pass a single column name where the spanning header begins. If multiple column names are passed, only the first is used.
`replace`	(scalar `logical`) Logical indicating whether to replace any existing footnotes in the specified location with the specified footnote, or whether the specified should be added to the existing footnote(s) in the header/cell. Default is to replace existing footnotes.
`text_interpret`	(`string`) String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`. Applies to tables printed with `{gt}`.
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Review rows argument details.
`level`	(`integer`) An integer specifying which level to place the spanning header footnote.

Value

Updated gtsummary object

See Also

Footnotes vs Source Notes vs Abbreviations

Examples

```
# Example 1 ----------------------------------
tbl <- trial |>
  tbl_summary(by = trt, include = c(age, grade), missing = "no") |>
  modify_footnote_header(
    footnote = "All but four subjects received both treatments in a crossover design",
    columns = all_stat_cols(),
    replace = FALSE
  ) |>
  modify_footnote_body(
    footnote = "Tumor grade was assessed _before_ treatment began",
    columns = "label",
    rows = variable == "grade" & row_type == "label"
  )
tbl
```

```
# Example 2 ---------------------------------
# remove all footnotes
tbl |>
  remove_footnote_header(columns = all_stat_cols()) |>
  remove_footnote_body(columns = label, rows = variable == "grade" & row_type == "label")
```

`modify_indent` *Modify column indentation*

Description

Add, increase, or reduce indentation for columns.

Usage

```
modify_indent(x, columns, rows = NULL, indent = 4L, double_indent, undo)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`columns`	(`tidy-select`) Selector of columns in `x$table_body`
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Review rows argument details.
`indent`	(`integer`) An integer indicating how many space to indent text
`double_indent, undo`	**[Deprecated]**

Value

a gtsummary table

See Also

Other Advanced modifiers: `modify_column_merge()`, `modify_table_styling()`

Examples

```
# remove indentation from `tbl_summary()`
trial |>
  tbl_summary(include = grade) |>
  modify_indent(columns = label, indent = 0L)

# increase indentation in `tbl_summary`
trial |>
  tbl_summary(include = grade) |>
  modify_indent(columns = label, rows = !row_type %in% 'label', indent = 8L)
```

`modify_missing_symbol` *Modify Missing Substitution*

Description

Specify how missing values will be represented in the printed table. By default, a blank space is printed for all `NA` values.

Usage

```
modify_missing_symbol(x, symbol, columns, rows)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`symbol`	(`string`) string indicating how missing values are formatted.
`columns`	(`tidy-select`) columns to add missing symbol.
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Review rows argument details.

Value

Updated gtsummary object

Examples

```
# Use the abbreivation "Ref." for reference rows instead of the em-dash
lm(marker ~ trt, data = trial) |>
  tbl_regression() |>
  modify_missing_symbol(
    symbol = "Ref.",
    columns = c(estimate, conf.low, conf.high),
```

```
  rows = reference_row == TRUE
)
```

`modify_post_fmt_fun` *Modify post formatting*

Description

[Experimental]
Apply a formatting function after the primary formatting functions have been applied. The function is similar to `gt::text_transform()`.

Usage

```
modify_post_fmt_fun(x, fmt_fun, columns, rows = TRUE)
```

Arguments

`x`	(`gtsummary`) A gtsummary object
`fmt_fun`	(`function`) a function that will be applied to the specified columns and rows.
`columns`	(`tidy-select`) Selector of columns in `x$table_body`
`rows`	(predicate `expression`) Predicate expression to select rows in `x$table_body`. Review rows argument details.

Value

Updated gtsummary object

Examples

```
# Example 1 ----------------------------------
data.frame(x = FALSE) |>
  tbl_summary(type = x ~ "categorical") |>
  modify_post_fmt_fun(
    fmt_fun = ~ifelse(. == "0 (0%)", "0", .),
    columns = all_stat_cols()
  )
```

`modify_source_note` *Modify source note*

Description

Add and remove source notes from a table. Source notes are similar to footnotes, expect they are not linked to a cell in the table.

Usage

```
modify_source_note(x, source_note, text_interpret = c("md", "html"))

remove_source_note(x, source_note_id = NULL)
```

Arguments

`x`	(`gtsummary`) A gtsummary object.
`source_note`	(`string`) A string to add as a source note.
`text_interpret`	(`string`) String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`. Applies to tables printed with `{gt}`.
`source_note_id`	(`integers`) Integers specifying the IDs of the source notes to remove. Source notes are indexed sequentially at the time of creation. Default is `NULL`, which removes all source notes.

Details

Source notes are not supported by `as_kable_extra()`.

Value

gtsummary object

See Also

Footnotes vs Source Notes vs Abbreviations

Examples

```
# Example 1 ----------------------------------
tbl <- tbl_summary(trial, include = c(marker, grade), missing = "no") |>
  modify_source_note("Results as of June 26, 2015")
tbl
```

```
# Example 2 ---------------------------------
remove_source_note(tbl, source_note_id = 1)
```

`modify_table_body` *Modify Table Body*

Description

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()` or call `modify_column_unhide()`.

Usage

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

Arguments

`x`	(`gtsummary`) A 'gtsummary' object
`fun`	(`function`) A function or formula. If a *function*, it is used as is. If a *formula*, e.g. `fun = ~ .x |> arrange(variable)`, it is converted to a function. The argument passed to `fun` is `x$table_body`.
`...`	Additional arguments passed on to the function

Value

A 'gtsummary' object

Examples

```
# Example 1 ---------------------------------
# Add number of cases and controls to regression table
trial |>
 tbl_uvregression(
   y = response,
   include = c(age, marker),
   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)
```

`plot` *Plot Regression Coefficients*

Description

The `plot()` function extracts `x$table_body` and passes the it to `ggstats::ggcoef_plot()` along with 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`, `tbl_uvregression`) A 'tbl_regression' or 'tbl_uvregression' object
`remove_header_rows`	(scalar `logical`) logical indicating whether to remove header rows for categorical variables. Default is `TRUE`
`remove_reference_rows`	(scalar `logical`) logical indicating whether to remove reference rows for categorical variables. Default is `FALSE`.
`...`	arguments passed to `ggstats::ggcoef_plot(...)`

Value

a ggplot

Examples

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

`proportion_summary` *Summarize a proportion*

Description

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

Usage

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

Arguments

`variable`	(`string`) String indicating the name of the variable from which the proportion will be computed.
`value`	(`scalar`) Value (or list of values) of `variable` to be taken into account in the numerator.
`weights`	(`string`) Optional string indicating the name of a frequency weighting variable. If `NULL`, all observations will be assumed to have a weight equal to 1.
`na.rm`	(scalar `logical`) Should missing values be removed before computing the proportion? (default is `TRUE`)
`conf.level`	(scalar `numeric`) Confidence level for the returned confidence interval. Must be strictly greater than 0 and less than 1. Default to 0.95, which corresponds to a 95 percent confidence interval.

`method` (`string`)
Confidence interval method. Must be one of `c("wilson", "wilson.no.correct", "wald", "wald.no.correct", "exact", "agresti.coull", "jeffreys")`. See `add_ci()` for details.

Details

Computed statistics:

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

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

Author(s)

Joseph Larmarange

Examples

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

`ratio_summary` *Summarize the ratio of two variables*

Description

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

Usage

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

Arguments

`numerator`	(`string`) String indicating the name of the variable to be summed for computing the numerator.
`denominator`	(`string`) String indicating the name of the variable to be summed for computing the denominator.
`na.rm`	(scalar `logical`) Should missing values be removed before summing the numerator and the denominator? (default is `TRUE`)
`conf.level`	(scalar `numeric`) Confidence level for the returned confidence interval. Must be strictly greater than 0 and less than 1. Default to 0.95, which corresponds to a 95 percent confidence interval.

Details

Computed statistics:

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

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

Author(s)

Joseph Larmarange

Examples

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

---

remove_row_type *Remove rows*

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", "level", "all"),
  level_value = NULL
)
```

Arguments

x	(gtsummary) A gtsummary object
variables	(tidy-select) Variables to to remove rows from. Default is everything()
type	(string) Type of row to remove. Must be one of c("header", "reference", "missing", "level", "all")
level_value	(string) When type='level' you can specify the *character* value of the level to remove. When NULL all levels are removed.

Value

Modified gtsummary table

Examples

```
# Example 1 ----------------------------------
trial |>
  dplyr::mutate(
    age60 = ifelse(age < 60, "<60", "60+")
  ) |>
  tbl_summary(by = trt, missing = "no", include = c(trt, age, age60)) |>
  remove_row_type(age60, type = "header")
```

---

select_helpers *Select helper functions*

Description

Set of functions to supplement the {tidyselect} 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_svysummary` 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)

all_intercepts()

all_interaction()

all_contrasts(
```

```
  contrasts_type = c("treatment", "sum", "poly", "helmert", "sdif", "other")
)

all_stat_cols(stat_0 = TRUE)
```

Arguments

continuous2	(scalar logical) Logical indicating whether to include continuous2 variables. Default is TRUE
dichotomous	(scalar logical) Logical indicating whether to include dichotomous variables. Default is TRUE
tests	(character) character vector indicating the test type of the variables to select, e.g. select all variables being compared with "t.test".
contrasts_type	(character) type of contrast to select. Select among contrast types c("treatment", "sum", "poly", "helmert", "sdif", "other"). Default is all contrast types.
stat_0	(scalar logical) When FALSE, will not select the "stat_0" column. Default is TRUE

Value

A character vector of column names selected

See Also

Review list, formula, and selector syntax used throughout gtsummary

Examples

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

---

separate_p_footnotes *Create footnotes for individual p-values*

Description

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

Usage

```
separate_p_footnotes(x)
```

Arguments

x	(tbl_summary, tbl_svysummary) Object with class "tbl_summary" or "tbl_svysummary"

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_summary(by = trt, include = c(age, grade)) |>
  add_p() |>
  separate_p_footnotes()
```

---

set_gtsummary_theme *Set gtsummary theme*

Description

Functions to **set**, **reset**, **get**, and evaluate **with** gtsummary themes.

- `set_gtsummary_theme()` set a theme
- `reset_gtsummary_theme()` reset themes
- `get_gtsummary_theme()` get a named list with all active theme elements
- `with_gtsummary_theme()` evaluate an expression with a theme temporarily set
- `check_gtsummary_theme()` checks if passed theme is valid

Usage

```
set_gtsummary_theme(x, quiet)

reset_gtsummary_theme()

get_gtsummary_theme()

with_gtsummary_theme(
  x,
  expr,
  env = rlang::caller_env(),
  msg_ignored_elements = NULL
)

check_gtsummary_theme(x)
```

Arguments

`x`	(named `list`) A named list defining a gtsummary theme.
`quiet`	**[Deprecated]**
`expr`	(`expression`) Expression to be evaluated with the theme specified in `x=` loaded
`env`	(`environment`) The environment in which to evaluate `expr=`
`msg_ignored_elements`	(`string`) Default is `NULL` with no message printed. Pass a string that will be printed with `cli::cli_alert_info()`. The `"{elements}"` object contains vector of theme elements that will be overwritten and ignored.

Details

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.

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 |>
  tbl_summary(by = trt, include = c(age, grade, trt)) |>
  add_stat_label() |>
  as_gt()

# reset gtsummary theme
reset_gtsummary_theme()
```

`sort_filter_p` *Sort/filter by p-values*

Description

Sort/filter by p-values

Usage

```
sort_p(x, q = FALSE)

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

Arguments

`x`	(`gtsummary`) An object created using gtsummary functions
`q`	(scalar `logical`) When `TRUE` will check the q-value column rather than the p-value. Default is `FALSE`.
`t`	(scalar `numeric`) Threshold below which values will be retained. Default is 0.05.

Author(s)

Karissa Whiting, Daniel D. Sjoberg

Examples

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

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

sort_hierarchical *Sort Hierarchical Tables*

Description

[Experimental]

This function is used to sort hierarchical tables. Options for sorting criteria are:

1. Descending - within each section of the hierarchy table, event rate sums are calculated for each row and rows are sorted in descending order by sum (default).
2. Alphanumeric - rows are ordered alphanumerically (i.e. A to Z) by label text. By default, `tbl_hierarchical()` sorts tables in alphanumeric order.

Usage

```
sort_hierarchical(x, ...)

## S3 method for class 'tbl_hierarchical'
sort_hierarchical(x, sort = everything() ~ "descending", ...)

## S3 method for class 'tbl_hierarchical_count'
sort_hierarchical(x, sort = everything() ~ "descending", ...)

## S3 method for class 'tbl_ard_hierarchical'
sort_hierarchical(x, sort = everything() ~ "descending", ...)
```

Arguments

`x`	(`tbl_hierarchical`, `tbl_hierarchical_count`, `tbl_ard_hierarchical`) a hierarchical gtsummary table of class `'tbl_hierarchical'`, `'tbl_hierarchical_count'`, or `'tbl_ard_hierarchical'`.
`...`	These dots are for future extensions and must be empty.
`sort`	(`formula-list-selector`, `string`) a named list, a list of formulas, a single formula where the list element is a named list of functions (or the RHS of a formula), or a string specifying the types of sorting to perform at each hierarchy level. If the sort method for any variable is not specified then the method will default to `"descending"`. If a single unnamed string is supplied it is applied to all hierarchy levels. For each variable, the value specified must be one of: • `"alphanumeric"` - at the specified hierarchy level, groups are ordered alphanumerically (i.e. A to Z) by `variable_level` text. • `"descending"` - at the specified hierarchy level, count sums are calculated for each row and rows are sorted in descending order by sum. If `sort` is `"descending"` for a given variable and `n` is included in `statistic` for the variable then `n` is used to calculate row sums, otherwise `p` is used. If neither `n` nor `p` are present in `x` for the variable, an error will occur.

Defaults to `everything() ~ "descending"`.

Details

If you do not want to display rates for a hierarchy variable from table `x` but you would like to sort by descending frequency for that variable, the recommended method is to keep the variable in `include` when calling `tbl_hierarchical()` so that rates for this variable are available, sort the table as needed using `sort_hierarchical()`, and then finally remove these rates from the table using `modify_table_body()`.

For example, to remove rates from the `AESOC` rows in hierarchy table `x`, you can call:

```
x |>
  modify_table_body(
   \(df) mutate(df, dplyr::across(all_stat_cols(), ~ifelse(variable %in% "AESOC", NA, .)))
  )
```

Value

a gtsummary table of the same class as `x`.

Note

When sorting a table that includes an overall column `add_overall()` must be called to add the overall column *before* `sort_hierarchical()` is called.

See Also

`filter_hierarchical()`

Examples

```
theme_gtsummary_compact()
ADAE_subset <- cards::ADAE |>
  dplyr::filter(AEBODSYS %in% c("SKIN AND SUBCUTANEOUS TISSUE DISORDERS",
                                "EAR AND LABYRINTH DISORDERS")) |>
  dplyr::filter(.by = AEBODSYS, dplyr::row_number() < 20)

tbl <-
  tbl_hierarchical(
    data = ADAE_subset,
    variables = c(AEBODSYS, AEDECOD),
    by = TRTA,
    denominator = cards::ADSL,
    id = USUBJID,
    overall_row = TRUE
  ) |>
  add_overall()

# Example 1 ----------------------------------------------
# Sort all variables by descending frequency (default)
sort_hierarchical(tbl)
```

```
# Example 2 ---------------------------------------------
# Sort all variables alphanumerically
sort_hierarchical(tbl, sort = everything() ~ "alphanumeric")

# Example 3 ---------------------------------------------
# Sort `AEBODSYS` alphanumerically, `AEDECOD` by descending frequency
sort_hierarchical(tbl, sort = list(AEBODSYS = "alphanumeric", AEDECOD = "descending"))

reset_gtsummary_theme()
```

`style_number` *Style numbers*

Description

Style numbers

Usage

```
style_number(
  x,
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  scale = 1,
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)
```

Arguments

`x`	(`numeric`) Numeric vector
`digits`	(non-negative `integer`) Integer or vector of integers specifying the number of decimals to round `x`. When vector is passed, each integer is mapped 1:1 to the numeric values in `x`
`big.mark`	(`string`) 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`	(`string`) The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`

scale	(scalar numeric) A scaling factor: x will be multiplied by scale before formatting.
prefix	(string) Additional text to display before the number.
suffix	(string) Additional text to display after the number.
na	(NA/string) Character to replace NA values with. Default is NA_character
...	Arguments passed on to base::format()

Value

formatted character vector

Examples

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

`style_percent` *Style percentages*

Description

Style percentages

Usage

```
style_percent(
  x,
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  prefix = "",
  suffix = "",
  symbol,
  na = NA_character_,
  ...
)
```

Arguments

x	numeric vector of percentages
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	(string) 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	(string) The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
prefix	(string) Additional text to display before the number.
suffix	(string) Additional text to display after the number.
symbol	Logical indicator to include percent symbol in output. Default is FALSE.
na	(NA/string) Character to replace NA values with. Default is NA_character
...	Arguments passed on to base::format()

Value

A character vector of styled percentages

Author(s)

Daniel D. Sjoberg

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, suffix = "%", digits = 1)
```

---

style_pvalue *Style p-values*

Description

Style p-values

Usage

```
style_pvalue(
  x,
  digits = 1,
  prepend_p = FALSE,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  na = NA_character_,
  ...
)
```

Arguments

x	(numeric) Numeric vector of p-values.
digits	(integer) Number of digits large p-values are rounded. Must be 1, 2, or 3. Default is 1.
prepend_p	(scalar logical) Logical. Should 'p=' be prepended to formatted p-value. Default is FALSE
big.mark	(string) 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	(string) The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
na	(NA/string) Character to replace NA values with. Default is NA_character
...	Arguments passed on to base::format()

Value

A character vector of styled p-values

Author(s)

Daniel D. Sjoberg

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 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 = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)
```

Arguments

x	(numeric) Numeric vector
digits	(integer) Integer specifying the number of significant digits to display for numbers below 1. Numbers larger than 1 will be be digits + 1. Default is digits = 2.
big.mark	(string) 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	(string) The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
prefix	(string) Additional text to display before the number.
suffix	(string) Additional text to display after the number.
na	(NA/string) Character to replace NA values with. Default is NA_character
...	Arguments passed on to base::format()

Value

A character vector of styled ratios

Author(s)

Daniel D. Sjoberg

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 1.2x10^2).

Usage

```
style_sigfig(
  x,
  digits = 2,
  scale = 1,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  prefix = "",
  suffix = "",
  na = NA_character_,
  ...
)
```

Arguments

`x`	Numeric vector
`digits`	Integer specifying the minimum number of significant digits to display
`scale`	(scalar `numeric`) A scaling factor: `x` will be multiplied by scale before formatting.
`big.mark`	(`string`) 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`	(`string`) The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`prefix`	(`string`) Additional text to display before the number.
`suffix`	(`string`) Additional text to display after the number.
`na`	(`NA`/`string`) Character to replace `NA` values with. Default is `NA_character`
`...`	Arguments passed on to `base::format()`

Value

A character vector of styled numbers

Details

- Scientific notation output is avoided.
- If 2 significant figures are requested, the number is rounded to no more than 2 decimal places. For example, a number will be rounded to 2 decimals places when `abs(x) < 1`, 1 decimal place when `abs(x) >= 1 & abs(x) < 10`, and to the nearest integer when `abs(x) >= 10`.
- 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 1.2x10^2).

Author(s)

Daniel D. Sjoberg

See Also

Other style tools: `label_style`

Examples

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

---

tbl_ard_continuous *Summarize continuous variable*

Description

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

Usage

```
tbl_ard_continuous(
  cards,
  variable,
  include,
  by = NULL,
  label = NULL,
  statistic = everything() ~ "{median} ({p25}, {p75})",
  value = NULL
)
```

Arguments

cards	(card) An ARD object of class "card" typically created with cards::ard_*() functions.
variable	(string) A single variable name of the continuous variable being summarized.
include	(character) Character vector of the categorical variables to
by	(string) A single variable name of the stratifying variable.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is everything() ~ "{median} ({p25}, {p75})".
value	(formula-list-selector) Supply a value to display a variable on a single row, printing the results for the variable associated with the value (similar to a 'dichotomous' display in tbl_summary()).

Value

a gtsummary table of class "tbl_ard_summary"

Examples

```
library(cards)

# Example 1 ----------------------------------
# the primary ARD with the results
ard_summary(
  # the order variables are passed is important for the `by` variable.
  # 'trt' is the column stratifying variable and needs to be listed first.
  trial, by = c(trt, grade), variables = age
) |>
  # adding OPTIONAL information about the summary variables
  bind_ard(
    # add univariate trt tabulation
    ard_tabulate(trial, variables = trt),
    # add missing and attributes ARD
    ard_missing(trial, by = c(trt, grade), variables = age),
    ard_attributes(trial, variables = c(trt, grade, age))
  ) |>
  tbl_ard_continuous(by = "trt", variable = "age", include = "grade")

# Example 2 ----------------------------------
```

```
# the primary ARD with the results
ard_summary(trial, by = grade, variables = age) |>
  # adding OPTIONAL information about the summary variables
  bind_ard(
    # add missing and attributes ARD
    ard_missing(trial, by = grade, variables = age),
    ard_attributes(trial, variables = c(grade, age))
  ) |>
  tbl_ard_continuous(variable = "age", include = "grade")
```

`tbl_ard_hierarchical` *ARD Hierarchical Table*

Description

[Experimental]
This is an preview of this function. There will be changes in the coming releases, and changes will not undergo a formal deprecation cycle.

Constructs tables from nested or hierarchical data structures (e.g. adverse events).

Usage

```
tbl_ard_hierarchical(
  cards,
  variables,
  by = NULL,
  include = everything(),
  statistic = ~"{n} ({p}%)",
  label = NULL
)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.
`variables`	(`tidy-select`) character vector or tidy-selector of columns in `data` used to create a hierarchy. Hierarchy will be built with variables in the order given.
`by`	(`tidy-select`) a single column from `data`. Summary statistics will be stratified by this variable. Default is `NULL`.
`include`	(`tidy-select`) columns from the `variables` argument for which summary statistics should be returned (on the variable label rows). Including the last element of `variables` has no effect since each level has its own row for this variable. The default is `everything()`.

`statistic` (formula-list-selector)
used to specify the summary statistics to display for all variables in `tbl_hierarchical()`. The default is `everything() ~ "{n} ({p})"`.

`label` (formula-list-selector)
used to override default labels in hierarchical table, e.g. `list(AESOC = "System Organ Class")`. The default for each variable is the column label attribute, `attr(., 'label')`. If no label has been set, the column name is used.

Value

a gtsummary table of class `"tbl_ard_hierarchical"`

Examples

```
ADAE_subset <- cards::ADAE |>
  dplyr::filter(
    AESOC %in% unique(cards::ADAE$AESOC)[1:5],
    AETERM %in% unique(cards::ADAE$AETERM)[1:5]
  )

# Example 1: Event Rates  --------------------
# First, build the ARD
ard <-
  cards::ard_stack_hierarchical(
    data = ADAE_subset,
    variables = c(AESOC, AETERM),
    by = TRTA,
    denominator = cards::ADSL,
    id = USUBJID
  )

# Second, build table from the ARD
tbl_ard_hierarchical(
  cards = ard,
  variables = c(AESOC, AETERM),
  by = TRTA
)

# Example 2: Event Counts  -------------------
ard <-
  cards::ard_stack_hierarchical_count(
    data = ADAE_subset,
    variables = c(AESOC, AETERM),
    by = TRTA,
    denominator = cards::ADSL
  )

tbl_ard_hierarchical(
  cards = ard,
  variables = c(AESOC, AETERM),
  by = TRTA,
  statistic = ~"{n}"
```

```
)
```

`tbl_ard_strata` *Stratified gtsummary tables from ARD*

Description

[Experimental]
Similar to `tbl_strata()`, except the function accepts an ARD instead of a data frame.

Usage

```
tbl_ard_strata(
  cards,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",
  .combine_with = c("tbl_merge", "tbl_stack"),
  .combine_args = NULL,
  .header = ifelse(.combine_with == "tbl_merge", "**{strata}**", "{strata}")
)

tbl_ard_strata2(
  cards,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",
  .combine_with = c("tbl_merge", "tbl_stack"),
  .combine_args = NULL,
  .header = ifelse(.combine_with == "tbl_merge", "**{strata}**", "{strata}")
)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.
`strata`	(`tidy-select`) the grouping columns to stratify by. Must select `'group#'` and `'group#_level'` pairs. Importantly, the function expects the `'group#'` columns to be the same variable, e.g. stratifying by a single variable. The `'group#_level'` value is available to place in header (and more) via the `{strata}` element.

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

`.header`	(`string`)
	String indicating the headers that will be placed. Default is `"**{strata}**"` when `.combine_with = "tbl_merge"` and `"{strata}"` when `.combine_with = "tbl_stack"`. Items placed in curly brackets will be evaluated according to `glue::glue()` syntax.
	`- \`strata\` stratum levels`
	`- \`n\` N within stratum`
	`- \`N\` Overall N`
	The evaluated value of `.header` is also available within `tbl_strata2(.tbl_fun)`

Value

a 'gtsummary' table

Examples

```
cards::ADLB |>
  dplyr::filter(
    AVISIT %in% c("Baseline", "Week 12", "Week 24"),
    PARAMCD %in% c("ALB", "BUN")
  ) |>
  cards::ard_summary(
    strata = PARAM,
    by = TRTA,
    variables = AVAL
  ) |>
  tbl_ard_strata2(
    strata = c(group2, group2_level),
    ~ .x |>
      tbl_ard_summary(by = TRTA, label = list(AVAL = .y)),
    .combine_with = "tbl_stack",
    .combine_args = list(group_header = NULL)
  )
```

`tbl_ard_summary` *ARD summary table*

Description

[Experimental]
The `tbl_ard_summary()` function tables descriptive statistics for continuous, categorical, and dichotomous variables. The functions accepts an ARD object.

Usage

```
tbl_ard_summary(
  cards,
  by = NULL,
 statistic = list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~
    "{n} ({p}%)"),
  type = NULL,
  label = NULL,
  missing = c("no", "ifany", "always"),
  missing_text = "Unknown",
  missing_stat = "{N_miss}",
  include = everything(),
  overall = FALSE
)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.
`by`	(`tidy-select`) A single column from `data`. Summary statistics will be stratified by this variable. Default is `NULL`
`statistic`	(`formula-list-selector`) Used to specify the summary statistics for each variable. Each of the statistics must be present in `card` as no new statistics are calculated in this function. The default is `list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)")`.
`type`	(`formula-list-selector`) Specifies the summary type. Accepted value are `c("continuous", "continuous2", "categorical", "dichotomous")`. Continuous summaries may be assigned `c("continuous", "continuous2")`, while categorical and dichotomous cannot be modified.
`label`	(`formula-list-selector`) Used to override default labels in summary table, e.g. `list(age = "Age, years")`. The default for each variable is the column label attribute, `attr(., 'label')`. If no label has been set, the column name is used.

`missing`, `missing_text`, `missing_stat`
: Arguments dictating how and if missing values are presented:
  - `missing`: must be one of `c("no", "ifany", "always")`
  - `missing_text`: string indicating text shown on missing row. Default is `"Unknown"`
  - `missing_stat`: statistic to show on missing row. Default is `"{N_miss}"`. Possible values are `N_miss`, `N_obs`, `N_nonmiss`, `p_miss`, `p_nonmiss`

`include`
: (tidy-select)
Variables to include in the summary table. Default is `everything()`

`overall`
: (scalar `logical`)
When `TRUE`, the `cards` input is parsed into two parts to run `tbl_ard_summary(cards_by) |> add_overall(cards_overall)`. Can only by used when `by` argument is specified. Default is `FALSE`.

Details

There are three types of additional data that can be included in the ARD to improve the default appearance of the table.

1. **Attributes**: When attributes are included, the default labels will be the variable labels, when available. Attributes can be included in an ARD with `cards::ard_attributes()` or `ard_stack(.attributes = TRUE)`.
2. **Missing**: When missing results are included, users can include missing counts or rates for variables with `tbl_ard_summary(missing = c("ifany", "always"))`. The missing statistics can be included in an ARD with `cards::ard_missing()` or `ard_stack(.missing = TRUE)`.
3. **Total N**: The total N is saved internally when available, and it can be calculated with `cards::ard_total_n()` or `ard_stack(.total_n = TRUE)`.

Value

a gtsummary table of class `"tbl_ard_summary"`

Examples

```
library(cards)

ard_stack(
  data = ADSL,
  ard_tabulate(variables = "AGEGR1"),
  ard_summary(variables = "AGE"),
  .attributes = TRUE,
  .missing = TRUE,
  .total_n = TRUE
) |>
  tbl_ard_summary()

ard_stack(
  data = ADSL,
```

```
  .by = ARM,
  ard_tabulate(variables = "AGEGR1"),
  ard_summary(variables = "AGE"),
  .attributes = TRUE,
  .missing = TRUE,
  .total_n = TRUE
) |>
  tbl_ard_summary(by = ARM)

ard_stack(
  data = ADSL,
  .by = ARM,
  ard_tabulate(variables = "AGEGR1"),
  ard_summary(variables = "AGE"),
  .attributes = TRUE,
  .missing = TRUE,
  .total_n = TRUE,
  .overall = TRUE
) |>
  tbl_ard_summary(by = ARM, overall = TRUE)
```

`tbl_ard_wide_summary` *Wide ARD summary table*

Description

[Experimental]
This function is similar to `tbl_ard_summary()`, but places summary statistics wide, in separate columns. All included variables must be of the same summary type, e.g. all continuous summaries or all categorical summaries (which encompasses dichotomous variables).

Usage

```
tbl_ard_wide_summary(
  cards,
  statistic = switch(type[[1]], continuous = c("{median}", "{p25}, {p75}"), c("{n}",
    "{p}%")),
  type = NULL,
  label = NULL,
  value = NULL,
  include = everything()
)
```

Arguments

`cards`	(`card`) An ARD object of class `"card"` typically created with `cards::ard_*()` functions.

`statistic` (character)
character vector of the statistics to present. Each element of the vector will result in a column in the summary table. Default is `c("{median}", "{p25}, {p75}")` for continuous summaries, and `c("{n}", "{p}%")` for categorical/dichotomous summaries

`type` (formula-list-selector)
Specifies the summary type. Accepted value are `c("continuous", "continuous2", "categorical", "dichotomous")`. If not specified, default type is assigned via `assign_summary_type()`. See below for details.

`label` (formula-list-selector)
Used to override default labels in summary table, e.g. `list(age = "Age, years")`. The default for each variable is the column label attribute, `attr(., 'label')`. If no label has been set, the column name is used.

`value` (formula-list-selector)
Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. `all_dichotomous()`, cannot be used with this argument. Default is `NULL`. See below for details.

`include` (tidy-select)
Variables to include in the summary table. Default is `everything()`.

Value

a gtsummary table of class 'tbl_wide_summary'

Examples

```
library(cards)

ard_stack(
  trial,
  ard_summary(variables = age),
  .missing = TRUE,
  .attributes = TRUE,
  .total_n = TRUE
) |>
  tbl_ard_wide_summary()

ard_stack(
  trial,
  ard_tabulate_value(variables = response),
  ard_tabulate(variables = grade),
  .missing = TRUE,
  .attributes = TRUE,
  .total_n = TRUE
) |>
  tbl_ard_wide_summary()
```

`tbl_butcher` *Butcher table*

Description

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

Usage

```
tbl_butcher(x, include = c("table_body", "table_styling"))
```

Arguments

`x`	(`gtsummary`) a gtsummary object
`include`	(`character`) names of additional elements to retain in the gtsummary object. `c("table_body", "table_styling")` will always be retained.

Value

a gtsummary object

Examples

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

tbl_butchered <-
  tbl_large |>
  tbl_butcher()

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

`tbl_continuous` *Summarize continuous variable*

Description

Summarize a continuous variable by one or more categorical variables

Usage

```
tbl_continuous(
  data,
  variable,
  include = everything(),
  digits = NULL,
  by = NULL,
  statistic = everything() ~ "{median} ({p25}, {p75})",
  label = NULL,
  value = NULL
)
```

Arguments

`data`	(`data.frame`) A data frame.
`variable`	(`tidy-select`) A single column from `data`. Variable name of the continuous column to be summarized.
`include`	(`tidy-select`) Variables to include in the summary table. Default is `everything()`.
`digits`	(`formula-list-selector`) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via `assign_summary_digits()`. See below for details.
`by`	(`tidy-select`) A single column from `data`. Summary statistics will be stratified by this variable. Default is `NULL`.
`statistic`	(`formula-list-selector`) Specifies summary statistics to display for each variable. The default is `everything() ~ "{median} ({p25}, {p75})"`.
`label`	(`formula-list-selector`) Used to override default labels in summary table, e.g. `list(age = "Age, years")`. The default for each variable is the column label attribute, `attr(., 'label')`. If no label has been set, the column name is used.

value (formula-list-selector)
Supply a value to display a variable on a single row, printing the results for the variable associated with the value (similar to a 'dichotomous' display in tbl_summary()).

Value

a gtsummary table

Examples

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

# Example 2 ----------------------------------
trial |>
  dplyr::mutate(all_subjects = 1) |>
  tbl_continuous(
    variable = age,
    statistic = ~"{mean} ({sd})",
    by = trt,
    include = c(all_subjects, stage, grade),
    value = all_subjects ~ 1,
    label = list(all_subjects = "All Subjects")
  )
```

---

tbl_cross *Cross table*

Description

The function creates a cross table of categorical variables.

Usage

```
tbl_cross(
  data,
  row = 1L,
  col = 2L,
  label = NULL,
  statistic = ifelse(percent == "none", "{n}", "{n} ({p}%)"),
  digits = NULL,
  percent = c("none", "column", "row", "cell"),
```

```
  margin = c("column", "row"),
  missing = c("ifany", "always", "no"),
  missing_text = "Unknown",
  margin_text = "Total"
)
```

Arguments

data	(data.frame) A data frame.
row	(tidy-select) Column name in data to be used for the rows of cross table. Default is the first column in data.
col	(tidy-select) Column name in data to be used for the columns of cross table. Default is the second column in data.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
statistic	(string) A string with the statistic name in curly brackets to be replaced with the numeric statistic (see glue::glue). The default is {n}. If percent argument is "column", "row", or "cell", default is "{n} ({p}%)".
digits	(numeric/list/function) Specifies the number of decimal places to round the summary statistics. This argument is passed to tbl_summary(digits = ~digits). By default integers are shown to the zero decimal places, and percentages are formatted with style_percent(). If you would like to modify either of these, pass a vector of integers indicating the number of decimal places to round the statistics. For example, if the statistic being calculated is "{n} ({p}%)" and you want the percent rounded to 2 decimal places use digits = c(0, 2). User may also pass a styling function: digits = style_sigfig
percent	(string) Indicates the type of percentage to return. Must be one of "none", "column", "row", or "cell". Default is "cell" when {N} or {p} is used in statistic.
margin	(character) Indicates which margins to add to the table. Default is c("row", "column"). Use margin = NULL to suppress both row and column margins.
missing	(string) Must be one of c("ifany", "no", "always").
missing_text	(string) String indicating text shown on missing row. Default is "Unknown"
margin_text	(string) Text to display for margin totals. Default is "Total"

Value

A tbl_cross object

Author(s)

Karissa Whiting, Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_cross(row = trt, col = response) |>
  bold_labels()

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

---

tbl_custom_summary *Create a table of summary statistics using a custom summary function*

Description

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

Usage

```
tbl_custom_summary(
  data,
  by = NULL,
  label = NULL,
  stat_fns,
  statistic,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = c("ifany", "no", "always"),
  missing_text = "Unknown",
  missing_stat = "{N_miss}",
  include = everything(),
  overall_row = FALSE,
  overall_row_last = FALSE,
  overall_row_label = "Overall"
)
```

Arguments

data	(data.frame) A data frame.
by	(tidy-select) A single column from data. Summary statistics will be stratified by this variable. Default is NULL.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
stat_fns	(formula-list-selector) Specifies the function to be used to compute the statistics (see below for details and examples). You can also use dedicated helpers such as ratio_summary() or proportion_summary().
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)"). See below for details.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits(). See below for details.
type	(formula-list-selector) Specifies the summary type. Accepted value are c("continuous", "continuous2", "categorical", "dichotomous"). If not specified, default type is assigned via assign_summary_type(). See below for details.
value	(formula-list-selector) Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. all_dichotomous(), cannot be used with this argument. Default is NULL. See below for details.
missing, missing_text, missing_stat	Arguments dictating how and if missing values are presented: • missing: must be one of c("ifany", "no", "always"). • missing_text: string indicating text shown on missing row. Default is "Unknown". • missing_stat: statistic to show on missing row. Default is "{N_miss}". Possible values are N_miss, N_obs, N_nonmiss, p_miss, p_nonmiss.
include	(tidy-select) Variables to include in the summary table. Default is everything().
overall_row	(scalar logical) Logical indicator to display an overall row. Default is FALSE. Use add_overall() to add an overall column.
overall_row_last	(scalar logical) Logical indicator to display overall row last in table. Default is FALSE, which will display overall row first.

`overall_row_label`
(string)
String indicating the overall row label. Default is `"Overall"`.

Value

A `tbl_custom_summary` object

Similarities with `tbl_summary()`

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

`stat_fns` argument

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

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

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

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

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

statistic argument

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

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

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

- `{N_obs}` total number of observations

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

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

It is recommended to use modify_footnote_header() to properly describe the displayed statistics (see examples).

Caution

The returned table is compatible with all gtsummary features applicable to a tbl_summary object, like add_overall(), modify_footnote_header() or bold_labels().

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

Author(s)

Joseph Larmarange

Examples

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

my_stats(trial)

trial |>
  tbl_custom_summary(
    include = c("stage", "grade"),
    by = "trt",
    stat_fns = everything() ~ my_stats,
    statistic = everything() ~ "A: {mean_age} - S: {marker_sum}",
    digits = everything() ~ c(1, 0),
    overall_row = TRUE,
    overall_row_label = "All stages & grades"
  ) |>
  add_overall(last = TRUE) |>
  modify_footnote_header(
```

```
    footnote = "A: mean age - S: sum of marker",
    columns = all_stat_cols()
  ) |>
  bold_labels()

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

trial |>
  tbl_custom_summary(
    include = c("marker", "ttdeath"),
    by = "trt",
    stat_fns = ~ mean_ci,
    statistic = ~ "{mean} [{conf.low}; {conf.high}]"
  ) |>
  add_overall(last = TRUE) |>
  modify_footnote_header(
    footnote = "mean [95% CI]",
    columns = all_stat_cols()
  )

# Example 3 ----------------------------------
# Use `full_data` to access the full datasets
# Returned statistic can also be a character
diff_to_great_mean <- function(data, full_data, ...) {
  mean <- mean(data$marker, na.rm = TRUE)
  great_mean <- mean(full_data$marker, na.rm = TRUE)
  diff <- mean - great_mean
  dplyr::tibble(
    mean = mean,
    great_mean = great_mean,
    diff = diff,
    level = ifelse(diff > 0, "high", "low")
  )
}

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

`tbl_hierarchical` *Hierarchical Table*

Description

[Experimental]
Use these functions to generate hierarchical tables.

- `tbl_hierarchical()`: Calculates *rates* of events (e.g. adverse events) utilizing the `denominator` and `id` arguments to identify the rows in `data` to include in each rate calculation. If `variables` contains more than one variable and the last variable in `variables` is an ordered factor, then rates of events by highest level will be calculated.
- `tbl_hierarchical_count()`: Calculates *counts* of events utilizing all rows for each tabulation.

Usage

```
tbl_hierarchical(
  data,
  variables,
  id,
  denominator,
  by = NULL,
  include = everything(),
  statistic = everything() ~ "{n} ({p}%)",
  overall_row = FALSE,
  label = NULL,
  digits = NULL
)

tbl_hierarchical_count(
  data,
  variables,
  denominator = NULL,
  by = NULL,
  include = everything(),
  overall_row = FALSE,
  statistic = everything() ~ "{n}",
  label = NULL,
  digits = NULL
)
```

Arguments

`data`	(`data.frame`) a data frame.

variables (tidy-select)
character vector or tidy-selector of columns in data used to create a hierarchy. Hierarchy will be built with variables in the order given.

id (tidy-select)
argument used to subset data to identify rows in data to calculate event rates in tbl_hierarchical().

denominator (data.frame, integer)
used to define the denominator and enhance the output. The argument is required for tbl_hierarchical() and optional for tbl_hierarchical_count(). The denominator argument must be specified when id is used to calculate event rates.

by (tidy-select)
a single column from data. Summary statistics will be stratified by this variable. Default is NULL.

include (tidy-select)
columns from the variables argument for which summary statistics should be returned (on the variable label rows). Including the last element of variables has no effect since each level has its own row for this variable. The default is everything().

statistic (formula-list-selector)
used to specify the summary statistics to display for all variables in tbl_hierarchical(). The default is everything() ~ "{n} ({p})".

overall_row (scalar logical)
whether an overall summary row should be included at the top of the table. The default is FALSE.

label (formula-list-selector)
used to override default labels in hierarchical table, e.g. list(AESOC = "System Organ Class"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.

digits (formula-list-selector)
specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via label_style_number() for statistics n and N, and label_style_percent(digits=1) for statistic p.

Value

a gtsummary table of class "tbl_hierarchical" (for tbl_hierarchical()) or "tbl_hierarchical_count" (for tbl_hierarchical_count()).

Overall Row

An overall row can be added to the table as the first row by specifying overall_row = TRUE. Assuming that each row in data corresponds to one event record, this row will count the overall number of events recorded when used in tbl_hierarchical_count(), or the overall number of patients recorded with any event when used in tbl_hierarchical().

A label for this overall row can be specified by passing an `'..ard_hierarchical_overall..'` element in `label`. Similarly, the rounding for statistics in the overall row can be modified using the `digits` argument, again referencing the `'..ard_hierarchical_overall..'` name.

Examples

```
ADAE_subset <- cards::ADAE |>
  dplyr::filter(
    AESOC %in% unique(cards::ADAE$AESOC)[1:5],
    AETERM %in% unique(cards::ADAE$AETERM)[1:5]
  )

# Example 1 - Event Rates --------------------
tbl_hierarchical(
  data = ADAE_subset,
  variables = c(AESOC, AETERM),
  by = TRTA,
  denominator = cards::ADSL,
  id = USUBJID,
  digits = everything() ~ list(p = 1),
  overall_row = TRUE,
  label = list(..ard_hierarchical_overall.. = "Any Adverse Event")
)

# Example 2 - Rates by Highest Severity ------
tbl_hierarchical(
  data = ADAE_subset |> mutate(AESEV = factor(AESEV, ordered = TRUE)),
  variables = c(AESOC, AESEV),
  by = TRTA,
  id = USUBJID,
  denominator = cards::ADSL,
  include = AESEV,
  label = list(AESEV = "Highest Severity")
)

# Example 3 - Event Counts -------------------
tbl_hierarchical_count(
  data = ADAE_subset,
  variables = c(AESOC, AETERM, AESEV),
  by = TRTA,
  overall_row = TRUE,
  label = list(..ard_hierarchical_overall.. = "Total Number of AEs")
)
```

---

tbl_likert *Likert Summary*

Description

Create a table of ordered categorical variables in a wide format.

Usage

```
tbl_likert(
  data,
  statistic = ~"{n} ({p}%)",
  label = NULL,
  digits = NULL,
  include = everything(),
  sort = c("ascending", "descending")
)
```

Arguments

data	(data.frame) A data frame.
statistic	(formula-list-selector) Used to specify the summary statistics for each variable. The default is everything() ~ "{n} ({p}%)".
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits().
include	(tidy-select) Variables to include in the summary table. Default is everything().
sort	(string) indicates whether levels of variables should be placed in ascending order (the default) or descending.

Value

a 'tbl_likert' gtsummary table

Examples

```
levels <- c("Strongly Disagree", "Disagree", "Agree", "Strongly Agree")
df_likert <- data.frame(
  recommend_friend = sample(levels, size = 20, replace = TRUE) |> factor(levels = levels),
  regret_purchase = sample(levels, size = 20, replace = TRUE) |> factor(levels = levels)
)

# Example 1 ----------------------------------
tbl_likert_ex1 <-
  df_likert |>
  tbl_likert(include = c(recommend_friend, regret_purchase)) |>
  add_n()
tbl_likert_ex1
```

```
# Example 2 ---------------------------------
# Add continuous summary of the likert scores
list(
  tbl_likert_ex1,
  tbl_wide_summary(
    df_likert |> dplyr::mutate(dplyr::across(everything(), as.numeric)),
    statistic = c("{mean}", "{sd}"),
    type = ~"continuous",
    include = c(recommend_friend, regret_purchase)
  )
) |>
  tbl_merge(tab_spanner = FALSE)
```

`tbl_merge` *Merge tables*

Description

Merge gtsummary tables, e.g. `tbl_regression`, `tbl_uvregression`, `tbl_stack`, `tbl_summary`, `tbl_svysummary`, etc.

This function merges **like tables**. Generally, this means each of the tables being merged should have the same structure. When merging tables with different structures, rows may appear out of order. The ordering of rows can be updated with `modify_table_body(~dplyr::arrange(.x, ...))`.

Usage

```
tbl_merge(
  tbls,
  tab_spanner = NULL,
  merge_vars = NULL,
  tbl_ids = NULL,
  quiet = FALSE
)
```

Arguments

`tbls`	(`list`) List of gtsummary objects to merge
`tab_spanner`	(`character`) Character vector specifying the spanning headers. Must be the same length as `tbls`. The strings are interpreted with `gt::md`. Must be same length as `tbls` argument. Default is `NULL`, and places a default spanning header. If `FALSE`, no header will be placed.
`merge_vars`	(`character`) Column names that are used as the merge IDs. The default is `NULL`, which merges on `c(any_of(c("variable", "row_type", "var_label", "label"), cards::all_ard_grou`

Any column name included here that does not appear in all tables, will be removed.

`tbl_ids` (`character`)
Optional character vector of IDs that will be assigned to the input tables. The ID is assigned by assigning a name to the `tbls` list, which is returned in `x$tbls`.

`quiet` (scalar `logical`)
When `FALSE`, a message is printed when unlike tables are merged warning users of potential row ordering issues.

Value

A `'tbl_merge'` object

Author(s)

Daniel D. Sjoberg

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(
  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(tbls = list(t3, t4)) %>%
  modify_spanning_header(everything() ~ NA_character_)
```

`tbl_regression` *Regression model summary*

Description

This function takes a regression model object and returns a formatted table that is publication-ready. The function is customizable allowing the user to create bespoke regression model summary tables. 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 = 0.95,
  intercept = FALSE,
 estimate_fun = ifelse(exponentiate, label_style_ratio(), label_style_sigfig()),
  pvalue_fun = label_style_pvalue(digits = 1),
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
  add_estimate_to_reference_rows = FALSE,
  conf.int = TRUE,
  ...
)
```

Arguments

`x`	(regression model) Regression model object
`...`	Additional arguments passed to `broom.helpers::tidy_plus_plus()`.
`label`	(`formula-list-selector`) Used to change variables labels, e.g. `list(age = "Age", stage = "Path T Stage")`
`exponentiate`	(scalar `logical`) Logical indicating whether to exponentiate the coefficient estimates. Default is `FALSE`.
`include`	(`tidy-select`) Variables to include in output. Default is `everything()`.

show_single_row
: (tidy-select)
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.

conf.level
: (scalar real)
Confidence level for confidence interval/credible interval. Defaults to 0.95.

intercept
: (scalar logical)
Indicates whether to include the intercept in the output. Default is FALSE

estimate_fun
: (function)
Function to round and format coefficient estimates. Default is label_style_sigfig() when the coefficients are not transformed, and label_style_ratio() when the coefficients have been exponentiated.

pvalue_fun
: (function)
Function to round and format p-values. Default is label_style_pvalue().

tidy_fun
: (function)
Tidier function for the model. Default is to use broom::tidy(). If an error occurs, the tidying of the model is attempted with parameters::model_parameters(), if installed.

add_estimate_to_reference_rows
: (scalar logical)
Add a reference value. Default is FALSE.

conf.int
: (scalar logical)
Logical indicating whether or not to include a confidence interval in the output. Default is TRUE.

Value

A tbl_regression object

Methods

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

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

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
glm(response ~ age + grade, trial, family = binomial()) |>
  tbl_regression(exponentiate = TRUE)
```

---

tbl_split_by *Split gtsummary table by rows and/or columns*

Description

[Experimental]
The `tbl_split_by_rows()` and `tbl_split_by_columns()` functions split a single gtsummary table into multiple tables. Both column-wise splitting (that is, splits by columns in `x$table_body`) and row-wise splitting is possible.

Usage

```
tbl_split_by_rows(
  x,
  variables = NULL,
  row_numbers = NULL,
  variable_level = NULL,
  footnotes = c("all", "first", "last"),
  caption = c("all", "first", "last")
)

tbl_split_by_columns(
  x,
  keys,
  groups,
  footnotes = c("all", "first", "last"),
  caption = c("all", "first", "last")
)

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

Arguments

`x`	(`gtsummary` or `list`) gtsummary table.

`variables`, `row_numbers`, `variable_level`
: (`tidy-select` or `integer`)
Specifies where the table will be split.

- `variables`: Tables will be separated after each of the variables specified. The `x$table_body` data frame must contains a `'variable'` column to use this argument.
- `row_numbers`: Row numbers after which the table will be split.
- `variable_level`: A single column name in `x$table_body`. When specified, the table will be split at each unique level of the variable.

`footnotes`, `caption`
: (`string`) **[Experimental]**
can be either `"first"`, `"all"`, or `"last"`, to locate global footnotes or caption only on the first, in each, or in the last table, respectively. It defaults to `"all"`. Reference footnotes are always present wherever they appear.

`keys`
: (`tidy-select`)
columns to be repeated in each table split. It defaults to the first column if missing (usually label column).

`groups`
: (list of `character` vectors)
list of column names that appear in `x$table_body`. Each group of column names represent a different table in the output list.

`...`
: These dots are for future extensions and must be empty.

Details

Run `show_header_names()` to print all column names to split by.

Footnotes and caption handling are experimental and may change in the future.

`row_numbers` indicates the row numbers at which to split the table. It means that the table will be split after each of these row numbers. If the last row is selected, the split will not happen as it is supposed to happen after the last row.

Value

`tbl_split` object. If multiple splits are performed (e.g., both by row and columns), the output is returned a single level list.

Examples

```
# Example 1 ----------------------------------
# Split by rows
trial |>
  tbl_summary(by = trt) |>
  tbl_split_by_rows(variables = c(marker, grade)) |>
  dplyr::last() # Print only last table for simplicity

# Example 2 ----------------------------------
# Split by rows with row numbers
trial |>
  tbl_summary(by = trt) |>
```

```
  tbl_split_by_rows(row_numbers = c(5, 7)) |>
  dplyr::last() # Print only last table for simplicity

# Example 3 ----------------------------------
# Split by columns
trial |>
  tbl_summary(by = trt, include = c(death, ttdeath)) |>
  tbl_split_by_columns(groups = list("stat_1", "stat_2")) |>
  dplyr::last() # Print only last table for simplicity

# Example 4 ----------------------------------
# Both row and column splitting
trial |>
  tbl_summary(by = trt) |>
  tbl_split_by_rows(variables = c(marker, grade)) |>
  tbl_split_by_columns(groups = list("stat_1", "stat_2")) |>
  dplyr::last() # Print only last table for simplicity

# Example 5 ----------------------------------
# Split by rows with footnotes and caption
trial |>
  tbl_summary(by = trt, missing = "no") |>
  modify_footnote_header(
    footnote = "All but four subjects received both treatments in a crossover design",
    columns = all_stat_cols(),
    replace = FALSE
  ) |>
  modify_footnote_body(
    footnote = "Tumor grade was assessed _before_ treatment began",
    columns = "label",
    rows = variable == "grade" & row_type == "label"
  ) |>
  modify_spanning_header(
    c(stat_1, stat_2) ~ "**TRT**"
  ) |>
  modify_abbreviation("I = 1, II = 2, III = 3") |>
  modify_caption("_Some caption_") |>
  modify_footnote_spanning_header(
    footnote = "Treatment",
    columns = c(stat_1)
  ) |>
  modify_source_note("Some source note!") |>
 tbl_split_by_rows(variables = c(marker, stage, grade), footnotes = "last", caption = "first") |>
  dplyr::nth(n = 2) # Print only one but not last table for simplicity
```

---

tbl_stack *Stack tables*

Description

Assists in patching together more complex tables. `tbl_stack()` appends two or more gtsummary tables.

Usage

```
tbl_stack(
  tbls,
  group_header = NULL,
  quiet = FALSE,
  attr_order = seq_along(tbls),
  tbl_ids = NULL,
  tbl_id_lbls = NULL
)
```

Arguments

`tbls`	(`list`) List of gtsummary objects
`group_header`	(`character`) Character vector with table headers where length matches the length of `tbls`
`quiet`	(scalar `logical`) Logical indicating whether to suppress additional messaging. Default is `FALSE`.
`attr_order`	(`integer`) Set the order table attributes are set. Tables are stacked in the order they are passed in the `tbls` argument: use `attr_order` to specify the order the table attributes take precedent. For example, to use the header from the second table specify `attr_order=2`. Default is to set precedent in the order tables are passed.
`tbl_ids`	(`character`) Optional character vector of IDs that will be assigned to the input tables. The ID is assigned by assigning a name to the `tbls` list, which is returned in `x$tbls`.
`tbl_id_lbls`	(`vector`) Optional vector of the same length `tbls`. When specified a new, hidden column is added to the returned `.$table_body` with these labels. *The most common use case of this argument is for the development of other functions.*

Value

A `tbl_stack` object

Author(s)

Daniel D. Sjoberg

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(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(list(row1, row2), group_header = c("Unadjusted Analysis", "Adjusted Analysis"))
```

`tbl_strata` *Stratified gtsummary tables*

Description

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

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

Tables are created *independently* within each stratum. When merging, keep in mind that merging works best with **like tables**. See `tbl_merge()` for details.

Usage

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

tbl_strata2(
  data,
  strata,
  .tbl_fun,
  ...,
  .sep = ", ",
  .combine_with = c("tbl_merge", "tbl_stack"),
  .combine_args = NULL,
  .header = ifelse(.combine_with == "tbl_merge", "**{strata}**", "{strata}"),
  .quiet = TRUE
)
```

Arguments

`data`	(`data.frame`, `survey.design`) a data frame or survey object
`strata`	(`tidy-select`) character vector or tidy-selector of columns in data to stratify results by. Only *observed* combinations are shown in results.
`.tbl_fun`	(`function`) A function or formula. If a *function*, it is used as is. If a formula, e.g. `~ .x %>% tbl_summary() %>% add_p()`, it is converted to a function. The stratified data frame is passed to this function.

...	Additional arguments passed on to the .tbl_fun function.
.sep	(string) when more than one stratifying variable is passed, this string is used to separate the levels in the spanning header. Default is ", "
.combine_with	(string) One of c("tbl_merge", "tbl_stack"). Names the function used to combine the stratified tables.
.combine_args	(named list) named list of arguments that are passed to function specified in .combine_with
.header	(string) String indicating the headers that will be placed. Default is "**{strata}**" when .combine_with = "tbl_merge" and "{strata}" when .combine_with = "tbl_stack". Items placed in curly brackets will be evaluated according to glue::glue() syntax. - `strata` stratum levels - `n` N within stratum - `N` Overall N The evaluated value of .header is also available within tbl_strata2(.tbl_fun)
.quiet	**[Deprecated]**

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.
  - The summary type for variables (e.g. continuous vs categorical vs dichotomous) are determined separately within stratum. Use the tbl_summary(type) argument to assign a summary type consistent across all tables being combined.
  - By default, a "missing" row appears when there are missing values only. Use the tbl_summary(missing) argument to ensure there is always/never a missing row for the combining of the tables.

Author(s)

Daniel D. Sjoberg

Examples

```
# Example 1 ----------------------------------
trial |>
  select(age, grade, stage, trt) |>
  mutate(grade = paste("Grade", grade)) |>
```

```
  tbl_strata(
    strata = grade,
    .tbl_fun =
      ~ .x |>
        tbl_summary(by = trt, missing = "no") |>
        add_n(),
    .header = "**{strata}**, N = {n}"
  )

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

`tbl_strata_nested_stack` *Stratified Nested Stacking*

Description

This function stratifies your data frame, builds gtsummary tables, and stacks the resulting tables in a nested style. The underlying functionality is similar to `tbl_strata()`, except the resulting tables are nested or indented within each group.

NOTE: The header from the first table is used for the final table. Oftentimes, this header will include incorrect Ns and *must be updated.*

Usage

```
tbl_strata_nested_stack(
  data,
  strata,
  .tbl_fun,
```

```
  ...,
  row_header = "{strata}",
  quiet = FALSE
)
```

Arguments

data	(data.frame) a data frame
strata	(tidy-select) character vector or tidy-selector of columns in data to stratify results by. Only *observed* combinations are shown in results.
.tbl_fun	(function) A function or formula. If a *function*, it is used as is. If a formula, e.g. ~ .x %>% tbl_summary() %>% add_p(), it is converted to a function. The stratified data frame is passed to this function.
...	Additional arguments passed on to the .tbl_fun function.
row_header	(string) string indicating the row headers that appear in the table. The argument uses glue::glue() syntax to insert values into the row headers. Elements available to insert are strata, n, N and p. The strata element is the variable level of the strata variables. Default is '{strata}'.
quiet	(scalar logical) Logical indicating whether to suppress additional messaging. Default is FALSE.

Value

a stacked 'gtsummary' table

Examples

```
# Example 1 ----------------------------------
tbl_strata_nested_stack(
  trial,
  strata = trt,
  .tbl_fun = ~ .x |>
    tbl_summary(include = c(age, grade), missing = "no") |>
    modify_header(all_stat_cols() ~ "**Summary Statistics**")
)

# Example 2 ----------------------------------
tbl_strata_nested_stack(
  trial,
  strata = trt,
  .tbl_fun = ~ .x |>
    tbl_summary(include = c(age, grade), missing = "no") |>
    modify_header(all_stat_cols() ~ "**Summary Statistics**"),
  row_header = "{strata}, n={n}"
) |>
  # bold the row headers; print `x$table_body` to see hidden columns
  modify_bold(columns = "label", rows = tbl_indent_id1 > 0)
```

tbl_summary *Summary table*

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 = list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~
    "{n} ({p}%)"),
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = c("ifany", "no", "always"),
  missing_text = "Unknown",
  missing_stat = "{N_miss}",
  sort = all_categorical(FALSE) ~ "alphanumeric",
  percent = c("column", "row", "cell"),
  include = everything()
)
```

Arguments

data	(data.frame) A data frame.
by	(tidy-select) A single column from data. Summary statistics will be stratified by this variable. Default is NULL.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)"). See below for details.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits(). See below for details.

`type` (formula-list-selector)
Specifies the summary type. Accepted value are `c("continuous", "continuous2", "categorical", "dichotomous")`. If not specified, default type is assigned via `assign_summary_type()`. See below for details.

`value` (formula-list-selector)
Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. `all_dichotomous()`, cannot be used with this argument. Default is `NULL`. See below for details.

`missing`, `missing_text`, `missing_stat`
Arguments dictating how and if missing values are presented:

- `missing`: must be one of `c("ifany", "no", "always")`.
- `missing_text`: string indicating text shown on missing row. Default is `"Unknown"`.
- `missing_stat`: statistic to show on missing row. Default is `"{N_miss}"`. Possible values are `N_miss`, `N_obs`, `N_nonmiss`, `p_miss`, `p_nonmiss`.

`sort` (formula-list-selector)
Specifies sorting to perform for categorical variables. Values must be one of `c("alphanumeric", "frequency")`. Default is `all_categorical(FALSE) ~ "alphanumeric"`.

`percent` (string)
Indicates the type of percentage to return. Must be one of `c("column", "row", "cell")`. Default is `"column"`.

In rarer cases, you may need to define/override the typical denominators. In these cases, pass an integer or a data frame. Refer to the `?cards::ard_tabulate(denominator)` help file for details. When a data frame is passed, this data frame is used to calculate header counts.

`include` (tidy-select)
Variables to include in the summary table. Default is `everything()`.

Value

a gtsummary table of class `"tbl_summary"`

A table of class `c('tbl_summary', 'gtsummary')`

statistic argument

The statistic argument specifies the statistics presented in the table. The input dictates the summary statistics presented in the table. 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.

The values are interpreted using `glue::glue()` syntax: a name that appears between curly brackets will be interpreted as a function name and the formatted result of that function will be placed in the table.

For categorical variables, the following statistics are available to display: `{n}` (frequency), `{N}` (denominator), `{p}` (percent).

For continuous variables, **any univariate function may be used**. The most commonly used functions are {median}, {mean}, {sd}, {min}, and {max}. Additionally, {p##} is available for percentiles, where ## is an integer from 0 to 100. For example, p25: quantile(probs=0.25, type=2).

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

digits argument

The digits argument specifies the the number of digits (or formatting function) statistics are rounded to.

The values passed can either be a single integer, a vector of integers, a function, or a list of functions. If a single integer or function is passed, it is recycled to the length of the number of statistics presented. For example, if the statistic is "{mean} ({sd})", it is equivalent to pass 1, c(1, 1), label_style_number(digits=1), and list(label_style_number(digits=1), label_style_number(digits=1)).

Named lists are also accepted to change the default formatting for a single statistic, e.g. list(sd = label_style_number(digits=1)).

type and value arguments

There are four summary types. Use the type argument to change the default 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")

Author(s)

Daniel D. Sjoberg

See Also

See tbl_summary vignette for detailed tutorial

See table gallery for additional examples

Review list, formula, and selector syntax used throughout gtsummary

Examples

```
# Example 1 ----------------------------------
trial |>
  select(age, grade, response) |>
  tbl_summary()

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

# Example 3 ----------------------------------
trial |>
  tbl_summary(
    include = c(age, marker),
    type = all_continuous() ~ "continuous2",
    statistic = all_continuous() ~ c("{median} ({p25}, {p75})", "{min}, {max}"),
    missing = "no"
  )
```

---

tbl_survfit *Survival table*

Description

Function takes a `survfit` object as an argument, and provides a formatted summary table of the results.

No more than one stratifying variable is allowed in each model. If you're experiencing unexpected errors using `tbl_survfit()`, please review ?tbl_survfit_errors for a possible explanation.

Usage

```
tbl_survfit(x, ...)

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

```
## S3 method for class 'data.frame'
tbl_survfit(x, y, include = everything(), conf.level = 0.95, ...)

## S3 method for class 'list'
tbl_survfit(
  x,
  times = NULL,
  probs = NULL,
  statistic = "{estimate} ({conf.low}, {conf.high})",
  label = NULL,
  label_header = ifelse(!is.null(times), "**Time {time}**",
    "**{style_sigfig(prob, scale=100)}% Percentile**"),
  estimate_fun = ifelse(!is.null(times), label_style_percent(suffix = "%"),
    label_style_sigfig()),
  missing = "--",
  type = NULL,
  reverse = FALSE,
  quiet = TRUE,
  ...
)
```

Arguments

x	(survfit, list, data.frame) 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 tbl_survfit.data.frame() and tbl_survfit.survfit() the arguments are passed to tbl_survfit.list(). They are not used when tbl_survfit.list() is called directly.
y	outcome call, e.g. y = Surv(ttdeath, death)
include	Variable to include as stratifying variables.
conf.level	(scalar numeric)] Confidence level for confidence intervals. Default is 0.95
times	(numeric) a vector of times for which to return survival probabilities.
probs	(numeric) a vector of probabilities with values in (0,1) specifying the survival quantiles to return.
statistic	(string) string defining the statistics to present in the table. Default is "{estimate} ({conf.low}, {conf.high})"
label	(formula-list-selector) List of formulas specifying variables labels, e.g. list(age = "Age, yrs", stage = "Path T Stage"), or a string for a single variable table.

`label_header` (string)
string specifying column labels above statistics. Default is `"{prob} Percentile"` for survival percentiles, and `"Time {time}"` for n-year survival estimates

`estimate_fun` (function)
function to format the Kaplan-Meier estimates. Default is `label_style_percent()` for survival probabilities and `label_style_sigfig()` for survival times

`missing` (string)
text to fill when estimate is not estimable. Default is `"--"`

`type` (string or NULL)
type of statistic to report. Available for Kaplan-Meier time estimates only, otherwise `type` is ignored. Default is `NULL`. Must be one of the following:

type	transformation
`"survival"`	`x`
`"risk"`	`1 - x`
`"cumhaz"`	`-log(x)`

`reverse` **[Deprecated]**

`quiet` **[Deprecated]**

Formula Specification

When passing a `survival::survfit()` object to `tbl_survfit()`, the `survfit()` call must use an evaluated formula and not a stored formula. Including a proper formula in the call allows the function to accurately identify all variables included in the estimation. See below for examples:

```
library(gtsummary)
library(survival)

# include formula in `survfit()` call
survfit(Surv(time, status) ~ sex, lung) |> tbl_survfit(times = 500)

# you can also pass a data frame to `tbl_survfit()` as well.
lung |>
  tbl_survfit(y = Surv(time, status), include = "sex", times = 500)
```

You **cannot**, however, pass a stored formula, e.g. `survfit(my_formula, lung)`, but you can use stored formulas with `rlang::inject(survfit(!!my_formula, lung))`.

Author(s)

Daniel D. Sjoberg

Examples

```
library(survival)

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

# Example 2 ----------------------------------
# Pass a data frame
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
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 |>
  dplyr::mutate(
    death_cr =
      dplyr::case_when(
        death == 0 ~ "censor",
        runif(n()) < 0.5 ~ "death from cancer",
        TRUE ~ "death other causes"
      ) |>
      factor()
  )

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

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

Description

The tbl_svysummary() function calculates descriptive statistics for continuous, categorical, and dichotomous variables taking into account survey weights and design.

Usage

```
tbl_svysummary(
  data,
  by = NULL,
  label = NULL,
 statistic = list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~
    "{n} ({p}%)"),
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = c("ifany", "no", "always"),
  missing_text = "Unknown",
  missing_stat = "{N_miss}",
  sort = all_categorical(FALSE) ~ "alphanumeric",
  percent = c("column", "row", "cell"),
  include = everything()
)
```

Arguments

data	(survey.design) A survey object created with created with survey::svydesign()
by	(tidy-select) A single column from data. Summary statistics will be stratified by this variable. Default is NULL.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)"). See below for details.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits(). See below for details.
type	(formula-list-selector) Specifies the summary type. Accepted value are c("continuous", "continuous2", "categorical", "dichotomous"). If not specified, default type is assigned via assign_summary_type(). See below for details.

`value` (formula-list-selector)
Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. `all_dichotomous()`, cannot be used with this argument. Default is `NULL`. See below for details.

`missing`, `missing_text`, `missing_stat`
Arguments dictating how and if missing values are presented:

- `missing`: must be one of `c("ifany", "no", "always")`.
- `missing_text`: string indicating text shown on missing row. Default is `"Unknown"`.
- `missing_stat`: statistic to show on missing row. Default is `"{N_miss}"`. Possible values are `N_miss`, `N_obs`, `N_nonmiss`, `p_miss`, `p_nonmiss`.

`sort` (formula-list-selector)
Specifies sorting to perform for categorical variables. Values must be one of `c("alphanumeric", "frequency")`. Default is `all_categorical(FALSE) ~ "alphanumeric"`.

`percent` (string)
Indicates the type of percentage to return. Must be one of `c("column", "row", "cell")`. Default is `"column"`.

`include` (tidy-select)
Variables to include in the summary table. Default is `everything()`.

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}` proportion
- `{p.std.error}` standard error of the sample proportion (on the 0 to 1 scale) computed with `survey::svymean()`
- `{deff}` design effect of the sample proportion computed with `survey::svymean()`
- `{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
- {mean.std.error} standard error of the sample mean computed with survey::svymean()
- {deff} design effect of the sample mean computed with survey::svymean()
- {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.

type and value arguments

There are four summary types. Use the type argument to change the default 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")

Author(s)

Joseph Larmarange

Examples

```
# Example 1 ----------------------------------
survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) |>
  tbl_svysummary(by = Survived, percent = "row", include = c(Class, Age))

# Example 2 ----------------------------------
# A dataset with a complex design
data(api, package = "survey")
survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc) |>
  tbl_svysummary(by = "both", include = c(api00, stype)) |>
  modify_spanning_header(all_stat_cols() ~ "**Survived**")
```

---

tbl_uvregression *Univariable regression model summary*

Description

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

Usage

```
tbl_uvregression(data, ...)

## S3 method for class 'data.frame'
tbl_uvregression(
  data,
  y = NULL,
  x = NULL,
  method,
  method.args = list(),
  exponentiate = FALSE,
  label = NULL,
  include = everything(),
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
  hide_n = FALSE,
  show_single_row = NULL,
  conf.level = 0.95,
 estimate_fun = ifelse(exponentiate, label_style_ratio(), label_style_sigfig()),
  pvalue_fun = label_style_pvalue(digits = 1),
  formula = "{y} ~ {x}",
  add_estimate_to_reference_rows = FALSE,
  conf.int = TRUE,
```

```
  ...
)

## S3 method for class 'survey.design'
tbl_uvregression(
  data,
  y = NULL,
  x = NULL,
  method,
  method.args = list(),
  exponentiate = FALSE,
  label = NULL,
  include = everything(),
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
  hide_n = FALSE,
  show_single_row = NULL,
  conf.level = 0.95,
 estimate_fun = ifelse(exponentiate, label_style_ratio(), label_style_sigfig()),
  pvalue_fun = label_style_pvalue(digits = 1),
  formula = "{y} ~ {x}",
  add_estimate_to_reference_rows = FALSE,
  conf.int = TRUE,
  ...
)
```

Arguments

`data`	(`data.frame`, `survey.design`) A data frame or a survey design object.
`...`	Additional arguments passed to `broom.helpers::tidy_plus_plus()`.
`y`, `x`	(`expression`, `string`) Model outcome (e.g. `y=recurrence` or `y=Surv(time, recur)`) or covariate (e.g. `x=trt`. All other column specified in `include` will be regressed against the constant `y` or `x`. Specify one and only one of `y` or `x`.
`method`	(`string/function`) Regression method or function, e.g. lm, glm, survival::coxph, `survey::svyglm`, etc. Methods may be passed as functions (`method=lm`) or as strings (`method='lm'`).
`method.args`	(named `list`) Named list of arguments passed to `method`.
`exponentiate`	(scalar `logical`) Logical indicating whether to exponentiate the coefficient estimates. Default is `FALSE`.
`label`	(`formula-list-selector`) Used to change variables labels, e.g. `list(age = "Age", stage = "Path T Stage")`
`include`	(`tidy-select`) Variables to include in output. Default is `everything()`.

`tidy_fun` (function)
Tidier function for the model. Default is to use `broom::tidy()`. If an error occurs, the tidying of the model is attempted with `parameters::model_parameters()`, if installed.

`hide_n` (scalar `logical`)
Hide N column. Default is `FALSE`

`show_single_row` (`tidy-select`)
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.

`conf.level` (scalar `real`)
Confidence level for confidence interval/credible interval. Defaults to `0.95`.

`estimate_fun` (function)
Function to round and format coefficient estimates. Default is `label_style_sigfig()` when the coefficients are not transformed, and `label_style_ratio()` when the coefficients have been exponentiated.

`pvalue_fun` (function)
Function to round and format p-values. Default is `label_style_pvalue()`.

`formula` (string)
String of the model formula. Uses `glue::glue()` syntax. Default is `"{y} ~ {x}"`, where `{y}` is the dependent variable, and `{x}` represents a single covariate. For a random intercept model, the formula may be `formula = "{y} ~ {x} + (1 | gear)"`.

`add_estimate_to_reference_rows` (scalar `logical`)
Add a reference value. Default is `FALSE`.

`conf.int` (scalar `logical`)
Logical indicating whether or not to include a confidence interval in the output. Default is `TRUE`.

Value

A `tbl_uvregression` object

`x` and `y` arguments

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.

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.

Author(s)

Daniel D. Sjoberg

See Also

See tbl_regression vignette for detailed examples

Examples

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

# Example 2 ----------------------------------
# rounding pvalues to 2 decimal places
library(survival)

tbl_uvregression(
  trial,
  method = coxph,
  y = Surv(ttdeath, death),
  exponentiate = TRUE,
  include = c("age", "grade", "response"),
  pvalue_fun = label_style_pvalue(digits = 2)
)
```

`tbl_wide_summary` *Wide summary table*

Description

This function is similar to `tbl_summary()`, but places summary statistics wide, in separate columns. All included variables must be of the same summary type, e.g. all continuous summaries or all categorical summaries (which encompasses dichotomous variables).

Usage

```
tbl_wide_summary(
  data,
  label = NULL,
 statistic = switch(type[[1]], continuous = c("{median}", "{p25}, {p75}"), c("{n}",
    "{p}%")),
  digits = NULL,
  type = NULL,
  value = NULL,
  sort = all_categorical(FALSE) ~ "alphanumeric",
  include = everything()
)
```

Arguments

`data`	(`data.frame`) A data frame.
`label`	(formula-list-selector) Used to override default labels in summary table, e.g. `list(age = "Age, years")`. The default for each variable is the column label attribute, `attr(., 'label')`. If no label has been set, the column name is used.
`statistic`	(`character`) character vector of the statistics to present. Each element of the vector will result in a column in the summary table. Default is `c("{median}", "{p25}, {p75}")` for continuous summaries, and `c("{n}", "{p}%")` for categorical/dichotomous summaries
`digits`	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via `assign_summary_digits()`. See below for details.
`type`	(formula-list-selector) Specifies the summary type. Accepted value are `c("continuous", "continuous2", "categorical", "dichotomous")`. If not specified, default type is assigned via `assign_summary_type()`. See below for details.

value (formula-list-selector)
Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. `all_dichotomous()`, cannot be used with this argument. Default is `NULL`. See below for details.

sort (formula-list-selector)
Specifies sorting to perform for categorical variables. Values must be one of `c("alphanumeric", "frequency")`. Default is `all_categorical(FALSE) ~ "alphanumeric"`.

include (tidy-select)
Variables to include in the summary table. Default is `everything()`.

Value

a gtsummary table of class 'tbl_wide_summary'

Examples

```
# Example 1 ----------------------------------
trial |>
  tbl_wide_summary(include = c(response, grade))

# Example 2 ----------------------------------
trial |>
  tbl_strata(
    strata = trt,
    ~tbl_wide_summary(.x, include = c(age, marker))
  )
```

---

theme_gtsummary *Available gtsummary themes*

Description

The following themes are available to use within the gtsummary package. Print theme elements with `theme_gtsummary_journal(set_theme = FALSE) |> print()`. Review the themes vignette for details.

Usage

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

theme_gtsummary_compact(set_theme = TRUE, font_size = NULL)

theme_gtsummary_printer(
```

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

theme_gtsummary_language(
  language = c("de", "en", "es", "fr", "gu", "hi", "is", "ja", "kr", "mr", "nl", "no",
    "pt", "se", "zh-cn", "zh-tw"),
  decimal.mark = NULL,
  big.mark = NULL,
  iqr.sep = NULL,
  ci.sep = NULL,
  set_theme = TRUE
)

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

theme_gtsummary_mean_sd(set_theme = TRUE)

theme_gtsummary_eda(set_theme = TRUE)
```

Arguments

journal	String indicating the journal theme to follow. One of c("jama", "lancet", "nejm", "qjecon"). Details below.
set_theme	(scalar logical) Logical indicating whether to set the theme. Default is TRUE. When FALSE the named list of theme elements is returned invisibly
font_size	(scalar numeric) Numeric font size for compact theme. Default is 13 for gt tables, and 8 for all other output types
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"
language	(string) 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), "nl" (Dutch), "mr" (Marathi), "no" (Norwegian), "pt" (Portuguese), "se" (Swedish), "zh-cn" (Chinese Simplified), "zh-tw" (Chinese Traditional) If a language is missing a translation for a word or phrase, please feel free to reach out on GitHub with the translated text.
decimal.mark	(string) The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")

`big.mark`	(`string`) 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`) 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`) 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*
    - `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
      - For flextable and huxtable output, the coefficients' standard error is placed below. For gt, it is placed to the right.
- `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.

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()

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