Package ‘gtsummary’

January 29, 2022

Title  Presentation-Ready Data Summary and Analytic Result Tables

Version  1.5.2

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

License  MIT + file LICENSE


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

Depends  R (>= 3.4)

Imports  broom (>= 0.7.11), broom.helpers (>= 1.6.0), cli (>= 3.1.1), dplyr (>= 1.0.7), forcats (>= 0.5.1), glue (>= 1.6.0), gt (>= 0.3.1), knitr (>= 1.37), lifecycle (>= 1.0.1), purrr (>= 0.3.4), rlang (>= 0.4.12), stringr (>= 1.4.0), tibble (>= 3.1.6), tidyr (>= 1.1.4)

Suggests  broom.mixed (>= 0.2.7), car (>= 3.0-11), covr, effectsize (>= 0.6.0), emmeans (>= 1.7.2), flextable (>= 0.6.10),
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_ci</td>
<td>4</td>
</tr>
<tr>
<td>add_difference</td>
<td>5</td>
</tr>
<tr>
<td>add_glance</td>
<td>7</td>
</tr>
<tr>
<td>add_global_p</td>
<td>9</td>
</tr>
<tr>
<td>add_n.tbl_summary</td>
<td>11</td>
</tr>
<tr>
<td>add_n.tbl_survfit</td>
<td>13</td>
</tr>
<tr>
<td>add_nevent.tbl_survfit</td>
<td>13</td>
</tr>
<tr>
<td>add_nevent_regression</td>
<td>14</td>
</tr>
<tr>
<td>add_n_regression</td>
<td>15</td>
</tr>
<tr>
<td>add_overall</td>
<td>16</td>
</tr>
<tr>
<td>add_p.tbl_continuous</td>
<td>18</td>
</tr>
<tr>
<td>add_p.tbl_cross</td>
<td>20</td>
</tr>
<tr>
<td>add_p.tbl_summary</td>
<td>21</td>
</tr>
<tr>
<td>add_p.tbl_survfit</td>
<td>23</td>
</tr>
<tr>
<td>add_p.tbl_svysummary</td>
<td>24</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>add_q</td>
<td>26</td>
</tr>
<tr>
<td>add_significance_stars</td>
<td>28</td>
</tr>
<tr>
<td>add_stat</td>
<td>29</td>
</tr>
<tr>
<td>add_stat_label</td>
<td>31</td>
</tr>
<tr>
<td>add_vif</td>
<td>33</td>
</tr>
<tr>
<td>as_flex_table</td>
<td>34</td>
</tr>
<tr>
<td>as_gt</td>
<td>36</td>
</tr>
<tr>
<td>as_lux_table</td>
<td>37</td>
</tr>
<tr>
<td>as_kable</td>
<td>38</td>
</tr>
<tr>
<td>as_kable_extra</td>
<td>39</td>
</tr>
<tr>
<td>as_tibble.gtsummary</td>
<td>42</td>
</tr>
<tr>
<td>bold_italicize_labels_levels</td>
<td>43</td>
</tr>
<tr>
<td>bold_p</td>
<td>44</td>
</tr>
<tr>
<td>combine_terms</td>
<td>45</td>
</tr>
<tr>
<td>continuous_summary</td>
<td>46</td>
</tr>
<tr>
<td>custom_tidiers</td>
<td>47</td>
</tr>
<tr>
<td>inline_text.gtsummary</td>
<td>49</td>
</tr>
<tr>
<td>inline_text.tbl_cross</td>
<td>50</td>
</tr>
<tr>
<td>inline_text.tbl_regression</td>
<td>51</td>
</tr>
<tr>
<td>inline_text.tbl_summary</td>
<td>52</td>
</tr>
<tr>
<td>inline_text.tbl_survfit</td>
<td>54</td>
</tr>
<tr>
<td>inline_text.tbl_uvregression</td>
<td>56</td>
</tr>
<tr>
<td>modify</td>
<td>57</td>
</tr>
<tr>
<td>modify.cols_merge</td>
<td>60</td>
</tr>
<tr>
<td>modify.column_alignment</td>
<td>61</td>
</tr>
<tr>
<td>modify.column_hide</td>
<td>62</td>
</tr>
<tr>
<td>modify.fmt.fun</td>
<td>63</td>
</tr>
<tr>
<td>modify.table_body</td>
<td>64</td>
</tr>
<tr>
<td>modify.table_styling</td>
<td>65</td>
</tr>
<tr>
<td>plot</td>
<td>67</td>
</tr>
<tr>
<td>proportion_summary</td>
<td>68</td>
</tr>
<tr>
<td>ratio_summary</td>
<td>70</td>
</tr>
<tr>
<td>remove_row_type</td>
<td>71</td>
</tr>
<tr>
<td>select_helpers</td>
<td>72</td>
</tr>
<tr>
<td>separate_p_footnotes</td>
<td>73</td>
</tr>
<tr>
<td>set_gtsummary_theme</td>
<td>74</td>
</tr>
<tr>
<td>sort_filter_p</td>
<td>75</td>
</tr>
<tr>
<td>style_number</td>
<td>76</td>
</tr>
<tr>
<td>style_percent</td>
<td>77</td>
</tr>
<tr>
<td>style_pvalue</td>
<td>78</td>
</tr>
<tr>
<td>style_ratio</td>
<td>79</td>
</tr>
<tr>
<td>style_sigfig</td>
<td>80</td>
</tr>
<tr>
<td>tbl butcher</td>
<td>81</td>
</tr>
<tr>
<td>tbl_continuous</td>
<td>82</td>
</tr>
<tr>
<td>tbl_cross</td>
<td>83</td>
</tr>
<tr>
<td>tbl_custom_summary</td>
<td>85</td>
</tr>
<tr>
<td>tbl_merge</td>
<td>89</td>
</tr>
<tr>
<td>tbl_regression</td>
<td>91</td>
</tr>
<tr>
<td>tbl_split</td>
<td>93</td>
</tr>
<tr>
<td>tbl_stack</td>
<td>94</td>
</tr>
<tr>
<td>tbl_strata</td>
<td>96</td>
</tr>
<tr>
<td>tbl_summary</td>
<td>99</td>
</tr>
</tbody>
</table>
Add CI Column

Description

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

Usage

```r
add_ci(x, ...)
```

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

Arguments

- `x`: A `tbl_summary` object
- `...`: Not used
- `method`: Confidence interval method. Default is `list(all_categorical() ~ "wilson", all_continuous() ~ "t.test")`. Must be one of `c("wilson", "wilson.no.correct", "exact", "asymptotic")` for categorical variables, and `c("t.test", "wilcox.test")` for continuous variables. See details below.
- `include`: variables to include in the summary table. Default is `everything()`
- `statistic`: Formula indicating how the confidence interval will be displayed. Default is `list(all_categorical() ~ "{conf.low}%,{conf.high}%", all_continuous() ~ "{conf.low},{conf.high}"`) for categorical variables, and `c("t.test", "wilcox.test")` for continuous variables. See details below.
- `conf.level`: Confidence level. Default is `0.95`
- `style_fun`: Function to style upper and lower bound of confidence interval. Default is `list(all_categorical() ~ purrr::partial(style_sigfig,scale = 100),all_continuous() ~ style_sigfig)`.
- `pattern`: string indicating 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.
Value

gtsummary table

method argument

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

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

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

Examples

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

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

add_difference

Add difference between groups

Description

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

Usage

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

Arguments

x "tbl_summary" or "tbl_svysummary" object

test List of formulas specifying statistical tests to perform for each variable, e.g. list(all_continuous() ~ \"t.test\") Common tests include "t.test" or "ancova" for continuous data, and "prop.test" for dichotomous variables. See tests for details and more tests.

group Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is NULL. See tests for methods that utilize the group= argument.

adj.vars Variables to include in mean difference adjustment (e.g. in ANCOVA models)

test.args List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests(\"t.test\") ~ list(var.equal = TRUE)

conf.level Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

include Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().

pvalue_fun Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the roundedformatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x, digits = 2) or equivalently, purrr::partial(style_pvalue, digits = 2)).

estimate_fun List of formulas specifying the formatting functions to round and format differences. Default is list(all_continuous() ~ style_sigfig, all_categorical() ~ function(x) paste0(style_sigfig(x * 100), \"\%\\)) Function to round and format difference. Default is style_sigfig()

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary
add_glance

Examples

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

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

---

**add_glance**

Add Model Statistics

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| Add model statistics returned from `broom::glance()`.
| **Usage** |
| **add_glance_table**(x, include = everything(), label = NULL, fmt_fun = NULL, glance_fun = broom::glance)

add_glance source note

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| Add model statistics returned from `broom::glance()`.
| **Usage** |
| **add_glance_source_note**(x, include = everything())


\[
\begin{align*}
\text{label} &= \text{NULL}, \\
\text{fmt_fun} &= \text{NULL}, \\
\text{glance_fun} &= \text{broom::glance}, \\
\text{text_interpret} &= \text{c("md", "html"),} \\
\text{sep1} &= \" = \", \\
\text{sep2} &= \"; \"
\end{align*}
\]

Arguments

\[x\] 'tbl_regression' object

\[\text{include}\] list of statistics to include in output. Must be column names of the tibble returned by \text{broom::glance()}\). The include argument can also be used to specify the order the statistics appear in the table.

\[\text{label}\] List of formulas specifying statistic labels, e.g. \text{list(r.squared \sim \"R2\", p.value \sim \"P\")}

\[\text{fmt_fun}\] List of formulas where the LHS is a statistic and the RHS is a function to format/round the statistics. The default is to round the number of observations and degrees of freedom to the nearest integer, p-values are styled with \text{style_pvalue()} and the remaining statistics are styled with \text{style_sigfig(x,digits = 3)}

\[\text{glance_fun}\] function that returns model statistics. Default is \text{broom::glance()}. Custom functions must return a single row tibble.

\[\text{text_interpret}\] String indicates whether source note text will be interpreted with \text{gt::md()} or \text{gt::html()}. Must be "md" (default) or "html".

\[\text{sep1}\] Separator between statistic name and statistic. Default is " = ", e.g. "R2 = 0.456"

\[\text{sep2}\] Separator between statistics. Default is " ; "

Value
gtsummary table

Default Labels

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

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Default Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>r.squared</td>
<td>R²</td>
</tr>
<tr>
<td>adj.r.squared</td>
<td>Adjusted R²</td>
</tr>
<tr>
<td>p.value</td>
<td>p-value</td>
</tr>
<tr>
<td>logLik</td>
<td>Log-likelihood</td>
</tr>
<tr>
<td>statistic</td>
<td>Statistic</td>
</tr>
<tr>
<td>df.residual</td>
<td>Residual df</td>
</tr>
<tr>
<td>null.deviance</td>
<td>Null deviance</td>
</tr>
<tr>
<td>df.null</td>
<td>Null df</td>
</tr>
<tr>
<td>nevent</td>
<td>N events</td>
</tr>
<tr>
<td>concordance</td>
<td>c-index</td>
</tr>
<tr>
<td>std.error.concordance</td>
<td>c-index SE</td>
</tr>
<tr>
<td>nob</td>
<td>No. Obs.</td>
</tr>
</tbody>
</table>
add_global_p

Deviance

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:

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

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Examples

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

# Example 1 ----------------------------------
add_glance_ex1 <-
  mod %>%
  add_glance_table(
    label = list(sigma ~ "\U03C3"),
    include = c(r.squared, AIC, sigma)
  )

# Example 2 ----------------------------------
add_glance_ex2 <-
  mod %>%
  add_glance_source_note(
    label = list(sigma ~ "\U03C3"),
    include = c(r.squared, AIC, sigma)
  )
```

add_global_p

Add the global p-values

Description

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

add_global_p(x, ...)

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

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

Arguments

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

Example Output

Author(s)

Daniel D. Sjoberg

See Also

- Review list, formula, and selector syntax used throughout gtsummary
- Other `tbl_uvregression` tools: `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_split()`, `tbl_strata()`, `tbl_uvregression()`
- Other `tbl_regression` tools: `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_split()`, `tbl_stack()`, `tbl_strata()`
Examples

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

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

---

**add_n.tbl_summary**  
**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

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

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

- **x**: Object with class tbl_summary from the tbl_summary function or with class tbl_svysummary from the tbl_svysummary function.
- **statistic**: String indicating the statistic to report. Default is the number of non-missing observation for each variable, statistic = "{n}". Other statistics available to report include:
  - "{N}" total number of observations,
  - "{n}" number of non-missing observations,
  - "{n_miss}" number of missing observations,
  - "{p}" percent non-missing data,
  - "{p_miss}" percent missing data

- **col_label**: String indicating the column label. Default is "**N**".
- **footnote**: Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is FALSE.
- **last**: Logical indicator to include N column last in table. Default is FALSE, which will display N column first.
- **missing**: DEPRECATED. Logical argument indicating whether to print N (missing = FALSE), or N missing (missing = TRUE). Default is FALSE.

... Not used

Value

A tbl_summary or tbl_svysummary object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Review list, formula, and selector syntax used throughout gtsummary

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

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

Examples

# Example 1 ----------------------------------
tbl_n_ex <-
trial[cl("trt", "age", "grade", "response")]
add_n(tbl_summary(by = trt))
Description

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

Usage

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

Arguments

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

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

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

Examples

```r
library(survival)
fic1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fic2 <- survfit(Surv(ttdeath, death) ~ trt, trial)

# Example 1 ----------------------------------
add_n.tbl_survfit.ex1 <-
  list(fic1, fic2) %>%
  tbl_survfit(times = c(12, 24)) %>%
  add_n()
```

Description

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

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Other tbl_survfit tools: add_nevent.tbl_survfit(), add_p.tbl_survfit(), modify.tbl_merge(), tbl_split(), tbl_stack(), tbl_strata(), tbl_survfit()
### Usage

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

### Arguments

- `x`: object of class `tbl_survfit`
- `...`: Not used

### Example Output

See Also

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

### Examples

```r
library(survival)
fit1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ trt, trial)
# Example 1 ----------------------------------
add_nevent.tbl_survfit_ex1 <-
list(fit1, fit2) %>%
tbl_survfit(times = c(12, 24)) %>%
add_n() %>%
add_nevent()
```

---

### Description

**Add event N to regression table**

Add event N to regression table

### Usage

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

### Arguments

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

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

Description

Add N to regression table

Usage

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

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

Arguments

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

Example Output
Examples

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

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

add_overall

Add column with overall summary statistics

Description

Add 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 = NULL,
  statistic = NULL,
  digits = NULL,
  ...
)

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

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

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

### Arguments

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

### Value

A tbl_* of same class as x

### Example Output

### Author(s)

Daniel D. Sjoberg

### See Also

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

Other tbl_continuous tools: add_p.tbl_continuous(), tbl_continuous()

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

Examples

# Example 1 ----------------------------------
tbl_overall_ex1 <-
  trial %>%
  tbl_summary(include = c(age, grade), by = trt) %>%
  add_overall()

# Example 2 ----------------------------------
tbl_overall_ex2 <-
  trial %>%
  tbl_summary(
    include = grade,
    by = trt,
    percent = "row",
    statistic = ~ "(p)%",
    digits = ~ 1
  ) %>%
  add_overall(
    last = TRUE,
    statistic = ~ "(p)% (n=n)",
    digits = ~ c(1, 0)
  )

# Example 3 ----------------------------------
tbl_overall_ex3 <-
  trial %>%
  tbl_continuous(
    variable = age,
    by = trt,
    include = grade
  ) %>%
  add_overall(last = TRUE)

---

add_p.tbl_continuous  P-values for tbl_continuous

Description

P-values for tbl_continuous

Usage

## S3 method for class 'tbl_continuous'
add_p(
Arguments

x Object with class tbl_summary from the tbl_summary function
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 to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
include Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().
test.args List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests("t.test") ~ list(var.equal = TRUE)
group Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is NULL. See tests for methods that utilize the group= argument.
...

See Also

Other tbl_continuous tools: add_overall(), tbl_continuous()

Examples

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

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

Usage

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

Arguments

- `x`: Object with class `tbl_cross` from the `tbl_cross` function
- `test`: 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 to round and format p-value. Default is `style_pvalue`, except when `source_note = TRUE` when the default is `style_pvalue(x,prepend_p = TRUE)`
- `source_note`: Logical value indicating whether to show p-value in the {gt} table source notes rather than a column.
- `test.args`: Named list containing additional arguments to pass to the test (if it accepts additional arguments). For example, add an argument for a chi-squared test with `test.args = list(correct = TRUE)`
- `...`: Not used

Example Output

Author(s)

Karissa Whiting

See Also

Other `tbl_cross` tools: `inline_text.tbl_cross()`, `tbl_cross()`
## add_p.tbl_summary

### Examples

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

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

### Description

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

### Usage

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

### Arguments

- **x**: Object with class `tbl_summary` from the `tbl_summary` function
- **test**: List of formulas specifying statistical tests to perform for each variable, e.g. `list(all_continuous() ~ "t.test", all_categorical() ~ "fisher.test")`. Common tests include "t.test", "aov", "wilcox.test", "kruskal.test", "chisq.test", "fisher.test", and "lm" (for clustered data). See `tests` for details, more tests, and instruction for implementing a custom test.
- **pvalue_fun**: Function to round and format p-values. Default is `style_pvalue`. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = function(x) style_pvalue(x, digits = 2)` or equivalently, `purrr::partial(style_pvalue, digits = 2)`).
add_p.tbl_summary

**group**
Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data. Default is NULL. See tests for methods that utilize the group= argument.

**include**
Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().

**test.args**
List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use test.args = all_tests("t.test") ~ list(var.equal = TRUE)

**exclude**
DEPRECATED.

... Not used

**Value**
A tbl_summary object

**Example Output**

**Author(s)**
Daniel D. Sjoberg, Emily C. Zabor

**See Also**
See tbl_summary vignette for detailed examples

Review list, formula, and selector syntax used throughout gtsummary

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

**Examples**

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

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


add_p.tbl_survfit

Description

[Maturing] Calculate and add a p-value

Usage

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

Arguments

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

Test argument

The most common way to specify `test=` is by using a single string indicating the test name. However, if you need to specify different tests within the same table, the input 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.

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

See Also

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

Examples

```r
library(survival)

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

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

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

---

**Description**

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

**Usage**

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

x Object with class tbl_svysummary from the tbl_svysummary function

test List of formulas specifying statistical tests to perform, e.g. list(all_continuous() ~ "svy.t.test", all_categorical() ~ "svy.wald.test"). Options include
- "svy.t.test" for a t-test adapted to complex survey samples (cf. survey::svytest),
- "svy.wilcox.test" for a Wilcoxon rank-sum test for complex survey samples (cf. survey::svyranktest),
- "svy.kruskal.test" for a Kruskal-Wallis rank-sum test for complex survey samples (cf. survey::svyranktest),
- "svy.vanderwaerden.test" for a van der Waerden’s normal-scores test for complex survey samples (cf. survey::svyranktest),
- "svy.median.test" for a Mood’s test for the median for complex survey samples (cf. survey::svyranktest),
- "svy.chisq.test" for a Chi-squared test with Rao & Scott’s second-order correction (cf. survey::svychisq),
- "svy.adj.chisq.test" for a Chi-squared test adjusted by a design effect estimate (cf. survey::svychisq),
- "svy.wald.test" for a Wald test of independence for complex survey samples (cf. survey::svychisq),
- "svy.adj.wald.test" for an adjusted Wald test of independence for complex survey samples (cf. survey::svychisq),
- "svy.lincom.test" for a test of independence using the exact asymptotic distribution for complex survey samples (cf. survey::svychisq),
- "svy.saddlepoint.test" for a test of independence using a saddlepoint approximation for complex survey samples (cf. survey::svychisq).

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

pvalue_fun Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = function(x) style_pvalue(x, digits = 2)` or equivalently, `purrr::partial(style_pvalue, digits = 2)`).

include Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is `everything()`.

test.args List of formulas containing additional arguments to pass to tests that accept arguments. For example, add an argument for all t-tests, use `test.args = all_tests("t.test") ~ list(var.equal = TRUE)`.

... Not used

Value

A tbl_svysummary object

Example Output

Author(s)

Joseph Larmarange
add_q

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

Description

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

Usage

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

Arguments

x a gtsummary object

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

quiet Logical indicating whether to print messages in console. Default is FALSE

Example Output

Author(s)

Esther Drill, Daniel D. Sjoberg

See Also

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

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

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

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

Examples

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

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

Add significance stars

Description

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

Usage

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

Arguments

x a 'tbl_regression' or 'tbl_uvregression' object

pattern glue-syntax string indicating what to display in formatted column. Default is "{estimate}{stars}". Other common patterns are "{estimate}{stars} ({conf.low},{conf.high})" and "{estimate} ({conf.low} to {conf.high}){stars}"

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

hide_ci logical whether to hide confidence interval. Default is TRUE

hide_p logical whether to hide p-value. Default is TRUE

hide_se logical whether to hide standard error. Default is FALSE

Future Updates

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

Example Output

Examples

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

# Example 1 ----------------------------------
add_significance_stars_ex1 <-
add_stat

Description

[Maturing] 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

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

Arguments

- `x`: tbl_summary, tbl_svysummary, or tbl_continuous object
- `fns`: list of formulas indicating the functions that create the statistic. See details below.
- `location`: list of formulas indicating the location the new statistics are placed. The RHS of the formula 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.
- `...`: DEPRECATED
Details

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

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

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

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

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

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

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Examples

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

add_stat_ex1 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(fns = everything() ~ my_ttest) %>%
  modify_header(
```
# Example 2 ----------------------------------
# fn returns t-test test statistic and pvalue
my_ttest2 <- function(data, variable, by, ...) {
  t.test(data[[variable]] ~ as.factor(data[[by]])) %>%
  broom::tidy() %>%
  mutate(
    stat = str_glue("t={style_sigfig(statistic)}, {style_pvalue(p.value, prepend_p = TRUE)}")
  ) %>%
  pull(stat)
}

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

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

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

**Description**

Adds labels describing the summary statistics presented for each variable in the `tbl_summary` / `tbl_svysummary` table.

**Usage**

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

**Arguments**

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

**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

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

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

**Example Output**

**Author(s)**

Daniel D. Sjoberg

**See Also**

- Review list, formula, and selector syntax used throughout gtsummary
- Other `tbl_summary` tools: `add_ci()`, `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`
add_vif

Add Variance Inflation Factor

Description

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

Usage

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

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 style_sigfig().

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Examples

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

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

as_flex_table

Convert gtsummary object to a flextable object

Description

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

Usage

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

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

Value

A flextable object

Details

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

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

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

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

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: as_gt(), as_hux_table(), as_kable_extra(), as_kable(), as_tibble.gtsummary()
Examples

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

Description

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

Review the `tbl_summary vignette` or `tbl_regression vignette` for detailed examples in the 'Advanced Customization' section.

Usage

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

Arguments

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

Value

A `gt_tbl` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

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

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

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

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

Arguments

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

Value

A huxtable object

Details

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

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

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

Author(s)
David Hugh-Jones

See Also
Other gtsummary output types: as_flex_table(), as_gt(), as_kable_extra(), as_kable(), as_tibble.gtsummary()

Examples

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

---

as_kable

Convert gtsummary object to a kable object

Description

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

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

Usage

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

Convert gtsummary object to a kableExtra object

Function converts a gtsummary object to a knitr_kable + kableExtra object. This allows the customized formatting available via knitr:kable and kableExtra: as_kable_extra() supports arguments in knitr:kable(). as_kable_extra() output via gtsummary supports bold and italic cells for table bodies. Users creating pdf output should specify as_kable_extra(format = "latex").
Usage

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

Arguments

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

Value

A `kableExtra` object

PDF via LaTeX Tips

This section discusses options intended for use with

- **output: pdf_document in yaml of .Rmd.**
- **as_kable_extra(format = "latex")**

Custom column names:

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

Additional table styling:

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

Example Output
Author(s)

Daniel D. Sjoberg

See Also

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

Examples

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

# Example 2 (PDF via LaTeX) ------------------------------------------
custom_names <- c(
  "\textbf{Characteristic}\n  \textbf{Drug A}\n  \textit{N = 98}\n  \textbf{Drug B}\n  \textit{N = 102}\n"
)
as_kable_extra_ex2_pdf <-
  trial %>%
  select(trt, age, stage) %>%
  tbl_summary(by = trt,
    statistic = list(all_categorical() ~ "{n} ({p}\%)"
  ) %>%
  bold_labels() %>%
  modify_footnote(
    update = all_stat_cols() ~ "Median (IQR); n (%)"
  ) %>%
  as_kable_extra(
    format = "latex",
    col.names = kableExtra::linebreak(custom_names, align = "c"),
    escape = FALSE
  )

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

Convert gtsummary object to a tibble

Description

Function converts a gtsummary object to a tibble.

Usage

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

Arguments

x Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
col_labels Logical argument adding column labels to output tibble. Default is TRUE.
return_calls Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
extclude DEPRECATED
fmt_missing Logical argument adding the missing value formats.
...

Value

a tibble

Author(s)

Daniel D. Sjoberg

See Also

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

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

  as_tibble(tbl)

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

---

**bold_italicize_labels_levels**

*Bold or Italicize labels or levels in gtsummary tables*

Description

Bold or Italicize labels or levels in gtsummary tables

Usage

- `bold_labels(x)`
- `bold_levels(x)`
- `italicize_labels(x)`
- `italicize_levels(x)`

Arguments

- `x` Object created using gtsummary functions

Value

Functions return the same class of gtsummary object supplied

Functions

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

Example Output

Author(s)

Daniel D. Sjoberg
bold_p

See Also

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

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

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

Examples

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

Description

Bold significant p-values or q-values below a chosen threshold (e.g. <0.05) in a gtsummary table.

Usage

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

Arguments

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

Example Output

Author(s)

Daniel D. Sjoberg, Esther Drill
**Examples**

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

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

**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**

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

**Arguments**

- `x` a `tbl_regression` object
- `formula_update` 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` help file for proper syntax. function’s formula.= argument
- `label` Option string argument labeling the combined rows
- `quiet` Logical indicating whether to print messages in console. Default is FALSE
- `...` Additional arguments passed to `stats::anova`

**Value**

`tbl_regression` object

**Example Output**

**Author(s)**

Daniel D. Sjoberg
See Also

Review list, formula, and selector syntax used throughout gtsummary

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

Examples

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

continuous_summary

Summarize a continuous variable

Description

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

Usage

```r
continuous_summary(variable)
```

Arguments

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

Details

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

Example Output
Author(s)

Joseph Larmarange

See Also

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

Examples

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

description

**Collection of custom tidiers**

### Description

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

### Usage

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

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

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

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

Arguments

x a regression model object

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

conf.level The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

... arguments passed to method:
  • pool_and_tidy_mice(): mice::tidy(x,...)
  • tidy_standardize(): effectsize::standardize_parameters(x,...)
  • tidy_bootstrap(): parameters::bootstrap_parameters(x,...)
  • tidy_robust(): parameters::model_parameters(x,...)

quiet Logical indicating whether to print messages in console. Default is FALSE

vcov_estimation, vcov_type, vcov_args arguments passed to parameters::model_parameters()

pool.args named list of arguments passed to mice::pool() in pool_and_tidy_mice(). Default is NULL

Details

• tidy_standardize() tidier to report standardized coefficients. The effectsize package includes a wonderful function to estimate standardized coefficients. The tidier uses the output from effectsize::standardize_parameters(), and merely takes the result and puts it in broom::tidy() format.

• tidy_bootstrap() tidier to report bootstrapped coefficients. The parameters package includes a wonderful function to estimate bootstrapped coefficients. The tidier uses the output from parameters::bootstrap_parameters(test = "p"), and merely takes the result and puts it in broom::tidy() format.
• tidy_robust() tidier to report robust standard errors, confidence intervals, and p-values. The parameters package includes a wonderful function to calculate robust standard errors, confidence intervals, and p-values. The tidier uses the output from parameters::model_parameters(), and merely takes the result and puts it in broom::tidy() format. To use this function with tbl_regression(), pass a function with the arguments for tidy_robust() populated. This is easily done using purrr::partial() e.g. tbl_regression(tidy_fun = partial(tidy_robust, vcov_estimation = "CL"))

• pool_and_tidy_mice() tidier to report models resulting from multiply imputed data using the mice package. Pass the mice model object before the model results have been pooled. See example.

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

Example Output

Examples

# Example 1 ----------------------------------
mod <- lm(age ~ marker + grade, trial)
tbl_stnd <- tbl_regression(mod, tidy_fun = tidy_standardize)
tbl <- tbl_regression(mod)
tidy_standardize_ex1 <- tbl_merge(
  list(tbl_stnd, tbl),
  tab_spanner = c("**Standardized Model**", "**Original Model**")
)

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

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

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

Arguments

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

**column + pattern**

Some gtsummary tables report multiple statistics in a single cell, e.g. 
```r
"{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. 
```r
inline_text(x, column = "stat_1", pattern = "{mean}")
```
reports just the mean from a `tbl_summary()`.

Description

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

Usage

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

Arguments

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

A string reporting results from a gtsummary table

See Also

Other tbl_cross tools: `add_p.tbl_cross()`, `tbl_cross()`

Examples

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

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

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

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

Arguments

- `x`: Object created from `tbl_regression`
- `variable`: Variable name of statistics to present
- `level`: Level of the variable to display for categorical variables. Default is NULL, returning the top row in the table for the variable.
- `pattern`: String indicating the statistics to return. Uses `glue::glue` formatting. Default is "{estimate} {{conf.level*100}% 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.
**inline_text.tbl_summary**

`estimate_fun` function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is `x$inputs$estimate_fun`.

`pvalue_fun` function to style p-values and/or q-values. Default is `function(x) style_pvalue(x, prepend_p = TRUE)`.

... Not used

**Value**

A string reporting results from a `gtsummary` table

**pattern argument**

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

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

**Author(s)**

Daniel D. Sjoberg

**See Also**

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

**Examples**

```r
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

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

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

Arguments

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

Value

A string reporting results from a `gtsummary` table

Author(s)

Daniel D. Sjoberg
See Also

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

Examples

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

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

inline_text.tbl_survfit
Report statistics from survfit tables inline

Description

[Maturing] Extracts and returns statistics from a tbl_survfit object for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

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

Arguments

- `x`: Object created from tbl_survfit
- `variable`: Variable name of statistic to present.
- `level`: Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is NULL
- `pattern`: String indicating the statistics to return.
- `time`: time for which to return survival probabilities.
- `prob`: probability with values in (0,1)
column column to print from $table_body$. Columns may be selected with time= or prob= as well.

estimate_fun Function to round and format estimate and confidence limits. Default is the same function used in tbl_survfit()

pvalue_fun Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).

Value
A string reporting results from a gtsummary table

Author(s)
Daniel D. Sjoberg

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

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

# 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, column = p.value)
inline_text(tbl2, prob = 0.5)
inline_text.tbl_uvregression

Report statistics from regression summary tables inline

Description

Extracts and returns statistics from a table created by the tbl_uvregression function for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

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

Arguments

x Object created from tbl_uvregression
variable Variable name of statistics to present
level Level of the variable to display for categorical variables. Default is NULL, returning the top row in the table for the variable.
pattern String indicating the statistics to return. Uses glue::glue formatting. Default is "{estimate} \{conf.level\% CI \{conf.low\}, \{conf.high\}; \{p.value\}". All columns from x$table_body are available to print as well as the confidence level (conf.level). See below for details.
estimate_fun function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is x$inputs$estimate_fun
pvalue_fun function to style p-values and/or q-values. Default is function(x) style_pvalue(x, prepend_p = TRUE)
...

Value

A string reporting results from a gtsummary table

pattern argument

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

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

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_uvregression tools: add_global_p(). add_q(). bold_italicize_labels_levels. modify(). tbl_merge(). tbl_split(). tbl_stack(). tbl_strata(). tbl_uvregression()

Examples

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

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

modify

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

Description

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

Usage

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

modify_footnote(
  x,
modify

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

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

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

Arguments

x
  a gtsummary object

update, ...
  use these arguments to assign updates to headers, spanning headers, and footnotes. See examples below.
  • update expects a list of assignments, with the variable name or selector on the LHS of the formula, and the updated string on the RHS. Also accepts a named list.
  • ... pass individual updates outside of a list, e.g. modify_header(p.value = "**P**", all_stat_cols() ~ **{level}**)

  Use the show_header_names() to see the column names that can be modified.

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

quiet
  Logical indicating whether to print messages in console. Default is FALSE

stat_by
  DEPRECATED, use update = all_stat_cols() ~ "<label>" instead.

abbreviation
  Logical indicating if an abbreviation is being updated.

caption
  a string of the table caption/title

include_example
  logical whether to include print of modify_header() example

Value

Updated gtsummary object

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

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

When there is a stratifying by= argument present, the following fields are additionally available to stratifying columns: \{level\}, \{n\}, and \{p\}(\{n_unweighted\} and \{p_unweighted\} for tbl_svysummary objects)
Syntax follows `glue::glue()`, e.g. `all_stat_cols() ~ "**{level}**, N = {n}"`.

**tbl_regression()**

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

**Captions**

Captions are assigned based on output type.
- `gt::gt(caption=)`
- `flextable::set_caption(caption=)`
- `huxtable::set_caption(value=)`
- `knitr::kable(caption=)`

**Example Output**

**Author(s)**

Daniel D. Sjoberg

**See Also**

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

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

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

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

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

**Examples**

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

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

# Example 1 ----------------------------------
# updating column headers, footnote, and table caption
```
modify_cols_merge

Modify Column Merging

Description

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

Usage

modify_cols_merge(x, pattern, rows = NULL)

Arguments

x gtsummary object

pattern glue syntax string indicating how to merge columns in x$table_body. For example, to construct a confidence interval use "(conf.low),(conf.high)".

rows 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.

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

Example Output

See Also

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

Examples

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

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

modify_column_alignment

Modify Column Alignment

Description

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

Usage

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

Arguments

x  gtsummary object

columns  vector or selector of columns in x$table_body

align  string indicating alignment of column, must be one of c("left", "right", "center")

See Also

Review list, formula, and selector syntax used throughout gtsummary

Other Advanced modifiers: modify_cols_merge(), modify_column_hide(), modify_fmt_fun(), modify_table_body(), modify_table_styling()

Examples

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

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

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

tbl <-
  lm(age ~ marker + grade, trial) %>%
  tbl_regression() %>%
  modify_column_hide(columns = c("Marker", "Grade"))

Description

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

Usage

modify_column_hide(x, columns)

modify_column_unhide(x, columns)

Arguments

x  gtsummary object

columns  vector or selector of columns in x$table_body

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Other Advanced modifiers: modify_cols_merge(), modify_column_alignment(), modify_fmt_fun(), modify_table_body(), modify_table_styling()
modify_fmt_fun

Examples

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

modify_fmt_fun

Modify Formatting Functions

Description

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

Usage

modify_fmt_fun(x, update, rows = NULL)

Arguments

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

Example Output

rows argument

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

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

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

Description

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

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

Usage

modify_table_body(x, fun, ...)

Arguments

x  gtsummary object

fun  A function or formula. If a 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 mapped function

Example Output

See Also

modify_table_styling()

See gtsummary internals vignette

Other Advanced modifiers: modify_cols_merge(), modify_column_alignment(), modify_column_hide(), modify_fmt_fun(), modify_table_styling()
Examples

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

modify_table_styling
Modify Table Styling

Description

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

Usage

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


Arguments

- **x**: gtsummary object
- **columns**: vector or selector of columns in x$table_body
- **rows**: predicate expression to select rows in x$table_body. Can be used to style footnote, formatting functions, missing symbols, and text formatting. Default is NULL. See details below.
- **label**: string of column label(s)
- **spanning_header**: string with text for spanning header
- **hide**: logical indicating whether to hide column from output
- **footnote**: string with text for footnote
- **footnote_abbrev**: string with abbreviation definition, e.g. "CI = Confidence Interval"
- **align**: string indicating alignment of column, must be one of c("left","right","center")
- **missing_symbol**: string indicating how missing values are formatted.
- **fmt_fun**: function that formats the statistics in the columns/rows in columns= and rows=
- **text_format**: string indicated which type of text formatting to apply to the rows and columns. Must be one of c("bold","italic","indent","indent2"). Do not assign both "indent" and "indent2" to the same cell.
- **undo_text_format**: rarely used. Logical that undoes the indent, bold, and italic styling when TRUE
- **text_interpret**: string, must be one of "md" or "html"
- **cols_merge_pattern**: glue-syntax string indicating how to merge columns in x$table_body. For example, to construct a confidence interval use "{conf.low},{conf.high}". The first column listed in the pattern string must match the single column name passed in columns=.

Details

Review the gtsummary definition vignette for information on .$table_styling objects.

**Rows argument**

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

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

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

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

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

See Also

modify_table_body()

See gtsummary internals vignette

Review list, formula, and selector syntax used throughout gtsummary

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

---

plot

\textit{Plot Regression Coefficients}

Description

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

Usage

\begin{verbatim}
## 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, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \ 'tbl_regression' or 'tbl_uvregression' object
  \item \texttt{remove_header_rows} logical indicating whether to remove header rows for categorical variables. Default is TRUE
  \item \texttt{remove_reference_rows} logical indicating whether to remove reference rows for categorical variables. Default is FALSE.
  \item ... arguments passed to GGally::ggcoef_plot(...)
\end{itemize}
proportion_summary

Details

[Experimental]

Value

a ggplot

Examples

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

proportion_summary  Summarize a proportion

Description

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

Usage

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

Arguments

variable  String indicating the name of the variable from which the proportion will be computed.

value  Value (or list of values) of variable to be taken into account in the numerator.

weights  Optional string indicating the name of a weighting variable. If NULL, all observations will be assumed to have a weight equal to 1.

na.rm  Should missing values be removed before computing the proportion? (default is TRUE)

conf.level  Confidence level for the returned confidence interval. Must be strictly greater than 0 and less than 1. Default to 0.95, which corresponds to a 95 percent confidence interval.

method  Confidence interval method. Must be one of c("wilson","wilson.no.correct","exact","asymptotic"). See details below.
**Details**

Computed statistics:

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

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

**Example Output**

**Author(s)**

Joseph Larmarange

**See Also**

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

**Examples**

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

Summarize the ratio of two variables

Description

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

Usage

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

Arguments

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

Details

Computed statistics:

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

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

Example Output

Author(s)

Joseph Larmarange

See Also

Other tbl_custom_summary tools: add_overall(), continuous_summary(), proportion_summary(), tbl_custom_summary()
Examples

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

---

**remove_row_type**  
*Remove rows by type*

**Description**

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

**Usage**

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

**Arguments**

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

**Example Output**

**See Also**

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

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

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 = NULL)
all_stat_cols(stat_0 = TRUE)
all_interaction()
all_intercepts()
all_contrasts(contrasts_type = NULL)
```
**Arguments**

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

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

- **tests**
  String indicating the test type of the variables to select, e.g. select all variables being compared with "t.test"

- **stat_0**
  When `FALSE`, will not select the "stat_0" column. Default is `TRUE`

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

**Value**

A character vector of column names selected

**Example Output**

**See Also**

Review list, formula, and selector syntax used throughout gtsummary

**Examples**

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

---

**separate_p_footnotes**

*Create footnotes for individual p-values*

**Description**

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

**Usage**

```r
separate_p_footnotes(x)
```

**Arguments**

- **x**
  Object with class "tbl_summary" or "tbl_svysummary"

**Example Output**
**set_gtsummary_theme**

Set a gtsummary theme

**Description**

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

**Usage**

set_gtsummary_theme(x)

reset_gtsummary_theme()

**Arguments**

- `x` A gtsummary theme function, e.g. theme_gtsummary_journal(), or a named list defining a gtsummary theme. See details below.

**Example Output**

**See Also**

Themes vignette

Available gtsummary themes
Examples

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

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

# reset gtsummary theme
reset_gtsummary_theme()

Description

Sort and filter variables in table by p-values

Usage

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

Arguments

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

Example Output

Author(s)

Karissa Whiting, Daniel D. Sjoberg

Examples

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

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

---

**style_number**

<table>
<thead>
<tr>
<th><strong>Style numbers</strong></th>
</tr>
</thead>
</table>

**Description**
Style numbers

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

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

**Value**
formatted character vector

**See Also**
Other style tools: `style_percent()`, `style_pvalue()`, `style_ratio()`, `style_sigfig()`

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

**Style percentages**

**Description**

Style percentages

**Usage**

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

**Arguments**

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

**Value**

A character vector of styled percentages

**Author(s)**

Daniel D. Sjoberg

**See Also**

See Table Gallery vignette for example

Other style tools: `style_number()`, `style_pvalue()`, `style_ratio()`, `style_sigfig()`

**Examples**

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

Description
Style p-values

Usage

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

Arguments

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

Value
A character vector of styled p-values

Author(s)
Daniel D. Sjoberg

See Also
See `tbl_summary` vignette for examples
Other style tools: `style_number()`, `style_percent()`, `style_ratio()`, `style_sigfig()`

Examples

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

*Style significant figure-like rounding for ratios*

**Description**

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

**Usage**

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

**Arguments**

- `x` Numeric vector
- `digits` 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` Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ",", except when `decimal.mark = "",` when the default is a space.
- `decimal.mark` The character to be used to indicate the numeric decimal point. Default is "." or `getOption("OutDec")`
- `...` Other arguments passed on to `base::format()`

**Value**

A character vector of styled ratios

**Author(s)**

Daniel D. Sjoberg

**See Also**

Other style tools: `style_number()`, `style_percent()`, `style_pvalue()`, `style_sigfig()`

**Examples**

```r
x <- 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

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

Arguments

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

Details

If 2 sig figs are input, the number is rounded to 2 decimal places when `abs(x) < 1`, 1 decimal place when `abs(x) >= 1 & abs(x) < 10`, and to the nearest integer when `abs(x) >= 10`.

Value

A character vector of styled numbers

Author(s)

Daniel D. Sjoberg

See Also

Other style tools: `style_number()`, `style_percent()`, `style_pvalue()`, `style_ratio()`
Examples

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

---

tbl_butcher Reduce size of gtsummary objects

Description

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

Usage

tbl_butcher(x)

Arguments

x a gtsummary object

Value

a gtsummary object

Examples

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

tbl_butchered <-
  tbl_large %>%
  tbl_butcher()

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

Summarize a continuous variable

Description

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

Usage

\[
tbl_continuous(
  data, \\
  variable, \\
  include = everything(), \\
  digits = NULL, \\
  by = NULL, \\
  statistic = NULL, \\
  label = NULL
)
\]

Arguments

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

Value

A `gtsummary` table

Example Output

See Also

Review list, formula, and selector syntax used throughout gtsummary

Other `tbl_continuous` tools: `add_overall()`, `add_p.tbl_continuous()`
Examples

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

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

Description

The function creates a cross table of two categorical variables.

Usage

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

Arguments

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

A string with the statistic name in curly brackets to be replaced with the numeric statistic (see \texttt{glue::glue}). The default is \{n\}. If percent argument is "column", "row", or "cell", default is \{n\} \{p\}%. 

digits

Specifies the number of decimal places to round the summary statistics. By default integers are shown to the zero decimal places, and percentages are formatted with \texttt{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 = \texttt{style_sfig}

percent

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

Indicates which margins to add to the table. Default is c("row","column"). Use margin = \texttt{NULL} to suppress both row and column margins.

missing

Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".

missing_text

String to display for count of missing observations. Default is "Unknown".

margin_text

Text to display for margin totals. Default is "Total"

Value

A tbl_cross object

Example Output

Author(s)

Karissa Whiting, Daniel D. Sjoberg

See Also

Review list, formula, and selector syntax used throughout gtsummary

Other tbl_cross tools: \texttt{add_p.tbl_cross()}, \texttt{inline_text.tbl_cross()}

Examples

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

tbl_cross_ex1 <-
  trial %>%
  tbl_cross(row = trt, col = response)
  
# Example 2 ----------------------------------

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

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

Arguments

data A data frame

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

label List of formulas specifying variables labels, e.g. list(age ~ "Age", stage ~ "Path T Stage"). If a variable’s label is not specified here, the label attribute (attr(data$age,"label")) is used. If attribute label is NULL, the variable name will be used.

stat_fns Formula or list of formulas specifying the function to be used to compute the statistics (see below for details and examples). You can also use dedicated helpers such as continuous_summary(), ratio_summary() or proportion_summary().

statistic List of formulas specifying the glue::glue() pattern to display the statistics for each variable. The statistics should be returned by the functions specified in stat_fns (see below for details and examples).

digits List of formulas specifying the number of decimal places to round summary statistics. If not specified, tbl_summary guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is "(mean) ((sd))" and you want the mean rounded to 1 decimal
place, and the SD to 2 use digits = list(age ~ c(1,2)). User may also pass
a styling function: digits = age ~ style_sigfig
type
List of formulas specifying variable types. Accepted values are c("continuous","continuous2","categorical","dichotomous"), e.g. type = list(age ~ "continuous", female ~ "dichotomous"). If type not
specified for a variable, the function will default to an appropriate summary type. See below for details.
value
List of formulas specifying the value to display for dichotomous variables. See below for details.
missing
Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text
String to display for count of missing observations. Default is "Unknown".
include
variables to include in the summary table. Default is everything().
overall_row
Logical indicator to display an overall row. Default is FALSE. Use add_overall() to add an overall column.
overall_row_last
Logical indicator to display overall row last in table. Default is FALSE, which will display overall row first.
overall_row_label
String indicating the overall row label. Default is "Overall".

Value
A tbl_custom_summary and tbl_summary object

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

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

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

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

The user-defined does not need to utilize each of these inputs. It’s encouraged the user-defined function accept ... as each of the arguments will be passed to the function, even if not all inputs are utilized by the user’s function, e.g. foo(data, ...) (see examples).
The user-defined function should return a one row dplyr::tibble() with one column per summary statistics (see examples).
**statistic argument**

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

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

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

- `(N_obs)` total number of observations
- `(N_miss)` number of missing observations
- `(N_nonmiss)` number of non-missing observations
- `(p_miss)` percentage of observations missing
- `(p_nonmiss)` percentage of observations not missing

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

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

**Caution**

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

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

**Example Output**

**Author(s)**

Joseph Larmarange

**See Also**

Review `list`, `formula`, and selector syntax used throughout `gtsummary`

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

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

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

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

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

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

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

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

==tbl_merge==

**Merge two or more gtsummary objects**

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

**Usage**

```r
tbl_merge(tblk, tab_spanner = NULL)
```

**Arguments**

tbls | List of gtsummary objects to merge

tab_spanner | Character vector specifying the spanning headers. Must be the same length as tbls. The strings are interpreted with `gt::md`. Must be same length as tbls argument. Default is NULL, and places a default spanning header. If FALSE, no header will be placed.

**Value**
A tbl_merge object

**Example Output**
Author(s)

Daniel D. Sjoberg

See Also

- tbl_stack
- Other tbl_regression tools: add_global_p(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify.tbl_regression(), tbl_split(), tbl_stack(), tbl_strata()
- Other tbl_uvregression tools: add_global_p(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify.tbl_split(), tbl_stack(), tbl_strata(), tbl_uvregression()
- Other tbl_summary tools: add_ci(), add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(), modify.separate_p_footnotes(), tbl_custom_summary(), tbl_split(), tbl_stack(), tbl_strata(), tbl_summary()
- Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), add_p.tbl_survfit(), modify.tbl_split(), tbl_stack(), tbl_strata(), tbl_survfit()
- Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_q(), add_stat_label(), modify, separate_p_footnotes(), tbl_split(), tbl_stack(), tbl_strata(), tbl_svysummary()

Examples

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

# Example 2 ----------------------------------
# Descriptive statistics alongside univariate regression, with no spanning header
t3 <-
  trial[c("age", "grade", "response")]
  tbl_summary(missing = "no") %>%
  add_n() %>%
  modify_header(stat_0 ~ "**Summary Statistics**")
t4 <-
  tbl_uvregression(
    trial[c("ttdeath", "death", "age", "grade", "response")],
    method = coxph,
    y = Surv(ttdeath, death),
    exponentiate = TRUE,
    hide_n = TRUE
  )
This function takes a regression model object and returns a formatted table that is publication-ready. The function is highly customizable allowing the user to obtain a bespoke summary table of the regression model results. Review the `tbl_regression` vignette for detailed examples.

### Usage

```r
tbl_regression(x, ...)  
```

### Arguments

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

A tbl_regression object

Methods

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

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

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

Example Output

Author(s)

Daniel D. Sjoberg
**See Also**

See tbl_regression vignette for detailed examples

Review list, formula, and selector syntax used throughout gtsummary

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

**Examples**

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

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

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

---

**Description**

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

**Usage**

```r
tbl_split(x, ...)
```

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

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

**Arguments**

- `x`: gtsummary table
- `...`: not used
- `variables`: variables at which to split the gtsummary table rows (tables will be separated after each of these variables)
Value

tbl_split object

See Also

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

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

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

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

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

Examples

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

tbl_stack

Stacks two or more gtsummary objects

Description

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

Usage

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

Arguments

tbls List of gtsummary objects

group_header Character vector with table headers where length matches the length of tbls=

quiet Logical indicating whether to print messages in console. Default is FALSE

Value

A tbl_stack object
### Example Output

**Author(s)**

Daniel D. Sjoberg

**See Also**

- `tbl_merge`

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

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

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

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

**Examples**

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

# stacking two `tbl_regression` objects

t1 <-
  glm(response ~ trt, trial, family = binomial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)"
  )
)

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

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

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

# stacking two `tbl_merge` objects

calling `survival`

t3 <-
  coxph(Surv(ttdeath, death) ~ trt, trial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)"
  )
t4 <-
coxph(Surv(ttdeath, death) ~ trt + grade + stage + marker, trial) %>%
tbl_regression(
  include = "trt",
  exponentiate = TRUE,
  label = list(trt = "Treatment (adjusted)"
)
)

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

---

**tbl_strata**

*Stratified gtsummary tables*

**Description**

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

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

**Usage**

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

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

data  a data frame or survey object
strata  character vector or tidy-selector of columns in data to stratify results by
.tbl_fun  A function or formula. If a 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  when more than one stratifying variable is passed, this string is used to separate the levels in the spanning header. Default is "," 
.combine_with  One of c("tbl_merge","tbl_stack"). Names the function used to combine the stratified tables.
.combine_args  named list of arguments that are passed to function specified in .combine_with=
.header  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=)
.stack_group_header  DEPRECATED.
.quiet  Logical indicating whether to print messages in console. Default is FALSE

Tips

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

Example Output

Author(s)

Daniel D. Sjoberg
See Also

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

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

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

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

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

Examples

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

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

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

Usage

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

Arguments

data A data frame

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

label List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`. If a variable's label is not specified here, the label attribute (attr(data$age,"label")) is used. If attribute label is `NULL`, the variable name will be used.

statistic List of formulas specifying types of 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 List of formulas specifying the number of decimal places to round summary statistics. If not specified, `tbl_summary` guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is "(mean) ((sd))" and you want the mean rounded to 1 decimal place, and the SD to 2 use `digits = list(age ~ c(1,2))`. User may also pass a styling function: `digits = age ~ style_sigfig`

type List of formulas specifying variable types. Accepted values are `c("continuous", "continuous2", "categorical", "dichotomous")`, e.g `type = list(age ~ "continuous", female ~ "dichotomous")`. If `type` is not specified for a variable, the function will default to an appropriate summary type. See below for details.
value List of formulas specifying the value to display for dichotomous variables. See below for details.

missing Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".

missing_text String to display for count of missing observations. Default is "Unknown".

sort List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. sort = list(everything() ~ "frequency")

percent Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".

include variables to include in the summary table. Default is everything()

Value
A tbl_summary object

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

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

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

Read more on the syntax used through the package.

type argument
The tbl_summary() function has four summary types:

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

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

For categorical variables the following statistics are available to display.

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

For continuous variables the following statistics are available to display.

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

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

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

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

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

Example Output

Author(s)

Daniel D. Sjoberg
See Also

See `tbl_summary` vignette for detailed tutorial

See table gallery for additional examples

Review list, formula, and selector syntax used throughout gtsummary

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

Examples

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

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

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

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

Creates table of survival probabilities

Description

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

Usage

```
tbl_survfit(x, ...)
```

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

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

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

Arguments

- **x**: a survfit object, list of survfit objects, or a data frame. If a data frame is passed, a list of survfit objects is constructed using each variable as a stratifying variable.
- **...**: For `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.
- **times**: numeric vector of times for which to return survival probabilities.
- **probs**: numeric vector of probabilities with values in (0,1) specifying the survival quantiles to return.
- **statistic**: string defining the statistics to present in the table. Default is "{estimate} ({conf.low},{conf.high})".
- **label**: List of formulas specifying variables labels, e.g. `list(age ~ "Age,yrs", stage ~ "Path T Stage")`, or a string for a single variable table.
label_header  string specifying column labels above statistics. Default is "{prob} Percentile" for survival percentiles, and "Time {time}" for n-year survival estimates

estimate_fun  function to format the Kaplan-Meier estimates. Default is style_percent() for survival probabilities and style_sigfig for survival times

missing  text to fill when estimate is not estimable. Default is "--"

conf.level  Confidence level for confidence intervals. Default is 0.95

reverse  Flip the probability reported, i.e. 1 -estimate. Default is FALSE. Does not apply to survival quantile requests

quiet  Logical indicating whether to print messages in console. Default is FALSE

y  outcome call, e.g. y = Surv(ttdeath, death)

include  Variable to include as stratifying variables.

Example Output

Author(s)
Daniel D. Sjoberg

See Also
Review list, formula, and selector syntax used throughout gtsummary

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

Examples

library(survival)

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

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

# Example 3 ------------------------------
# Pass a list of survfit() objects
tbl_survfit_ex3 <- list
  survfit(Surv(ttdeath, death) ~ 1, trial),
  survfit(Surv(ttdeath, death) ~ trt, trial),
  survfit(Surv(ttdeath, death) ~ grade, trial)
)
survfit(Surv(ttdeath, death) ~ trt, trial) %>%
tbl_survfit(times = c(12, 24))

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

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

---

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

**Description**

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

**Usage**

```r
tbl_svysummary(
  data,  # A survey object created with created with survey::svydesign()
  by = NULL,  # A column name (quoted or unquoted) in data. Summary statistics will be
calculated separately for each level of the by variable (e.g. by = trt). If NULL,
summary statistics are calculated using all observations. To stratify a table by
two or more variables, use tbl_strata()
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  sort = NULL,
  percent = NULL,
  include = everything()
)
```

**Arguments**

- `data`  
- `by`
**label**

List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`. If a variable's label is not specified here, the label attribute (`attr(data$age, "label")`) is used. If attribute label is NULL, the variable name will be used.

**statistic**

List of formulas specifying types of 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**

List of formulas specifying the number of decimal places to round summary statistics. If not specified, `tbl_summary` guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is "{mean} ({sd})" and you want the mean rounded to 1 decimal place, and the SD to 2 use `digits = list(age ~ c(1,2))`. User may also pass a styling function: `digits = age ~ style_sigfig`.

**type**

List of formulas specifying variable types. Accepted values are c("continuous", "continuous2", "categorical", "dichotomous"), e.g. `type = list(age ~ "continuous", female ~ "dichotomous")`. If type not specified for a variable, the function will default to an appropriate summary type. See below for details.

**value**

List of formulas specifying the value to display for dichotomous variables. See below for details.

**missing**

Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".

**missing_text**

String to display for count of missing observations. Default is "Unknown".

**sort**

List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. `sort = list(everything() ~ "frequency")`.

**percent**

Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".

**include**

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

---

**Value**

A `tbl_ssvsummary` object

---

**statistic argument**

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

For categorical variables the following statistics are available to display.

- `(n)` frequency
- `(N)` denominator, or cohort size
- `(p)` formatted percentage
- `(n_unweighted)` unweighted frequency
For continuous variables the following statistics are available to display.

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

- \( \{ \text{N_obs} \} \) total number of observations
- \( \{ \text{N_miss} \} \) number of missing observations
- \( \{ \text{N_nonmiss} \} \) number of non-missing observations
- \( \{ \text{p_miss} \} \) percentage of observations missing
- \( \{ \text{p_nonmiss} \} \) percentage of observations not missing
- \( \{ \text{N_obs_unweighted} \} \) unweighted total number of observations
- \( \{ \text{N_miss_unweighted} \} \) unweighted number of missing observations
- \( \{ \text{N_nonmiss_unweighted} \} \) unweighted number of non-missing observations
- \( \{ \text{p_miss_unweighted} \} \) unweighted percentage of observations missing
- \( \{ \text{p_nonmiss_unweighted} \} \) unweighted percentage of observations not missing

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

**Example Output**

**type argument**

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

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

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

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

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

Read more on the syntax used through the package.

**Author(s)**

Joseph Larmarange

**See Also**

Review list, formula, and selector syntax used throughout gtsummary

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

**Examples**

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

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

---

**tbl_uvregression**

Display univariate regression model results in table

**Description**

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

For models holding outcome constant, the function takes as arguments a data frame, the type of regression model, and the outcome variable y=. Each column in the data frame is regressed on the specified outcome. The `tbl_uvregression` function arguments are similar to the `tbl_regression` arguments. Review the `tbl_uvregression vignette` for detailed examples.
You may alternatively hold a single covariate constant. For this, pass a data frame, the type of regression model, and a single covariate in the x= argument. Each column of the data frame will serve as the outcome in a univariate regression model. Take care using the x argument that each of the columns in the data frame are appropriate for the same type of model, e.g. they are all continuous variables appropriate for lm, or dichotomous variables appropriate for logistic regression with glm.

Usage

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

Arguments

data  Data frame to be used in univariate regression modeling. Data frame includes the outcome variable(s) and the independent variables. Survey design objects are also accepted.

method  Regression method (e.g. lm, glm, survival::coxph, survey::svyglm, and more).

y  Model outcome (e.g. y = recurrence or y = Surv(time, recur)). All other column in data will be regressed on y. Specify one and only one of y or x

x  Model covariate (e.g. x = trt). All other columns in data will serve as the outcome in a regression model with x as a covariate. Output table is best when x is a continuous or dichotomous variable displayed on a single row. Specify one and only one of y or x

method.args  List of additional arguments passed on to the regression function defined by method.

exponentiate  Logical indicating whether to exponentiate the coefficient estimates. Default is FALSE.

date  Logical indicating whether to exponentiate the coefficient estimates. Default is FALSE.

label  List of formulas specifying variables labels, e.g. list(age ~ "Age", stage ~ "Path T Stage")

include  Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().
tidy_fun  Option to specify a particular tidier function for the model. Default is to use broom::tidy, but if an error occurs then tidying of the model is attempted with parameters::model_parameters(). if installed.

hide_n  Hide N column. Default is FALSE.

show_single_row  By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here—quoted and unquoted variable name accepted.

conf.level  Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

estimate_fun  Function to round and format coefficient estimates. Default is style_sigfig when the coefficients are not transformed, and style_ratio when the coefficients have been exponentiated.

pvalue_fun  Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).

formula  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  add a reference value. Default is FALSE.

conf.int  Logical indicating whether or not to include a confidence interval in the output. Defaults to TRUE.

...  Not used

Value

A tbl_uvregression object

Example Output

Methods

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

- "parsnip/workflows": If the model was prepared using parsnip/workflows, the original model fit is extracted and the original x= argument is replaced with the model fit. This will typically go unnoticed; however, if you’ve provided a custom tidier in tidy_fun= the tidier will be applied to the model fit object and not the parsnip/workflows object.

- "survreg": The scale parameter is removed, broom::tidy(x) %>% dplyr::filter(term != "Log(scale)")

- "multinom": This multinominal outcome is complex, with one line per covariate per outcome (less the reference group)
### Description

**[Maturing]** The following themes are available to use within the gtsummary package. Print theme elements with `theme_gtsummary_journ(set_theme = FALSE) %>% print()`. Review the [themes vignette](http://example.com) for details.

<table>
<thead>
<tr>
<th>theme_gtsummary</th>
<th>Available gtsummary themes</th>
</tr>
</thead>
</table>

- **“gam”**: Uses the internal tidier `tidy_gam()` to print both parametric and smooth terms.
- **“lmerMod”, “glmerMod”, “glmmTMB”, “glmmadmb”, “stanreg”, “brmsfit”**: These mixed effects models use `broom.mixed::tidy(x, effects = “fixed”). Specify tidy_fun = broom.mixed::tidy` to print the random components.

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

**Author(s)**

Daniel D. Sjoberg

**See Also**

See `tbl_regression` vignette for detailed examples

Review [list, formula, and selector syntax](http://example.com) used throughout gtsummary

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

**Examples**

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

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

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(
    decimal.mark = NULL,
    big.mark = NULL,
    iqr.sep = NULL,
    ci.sep = NULL,
    set_theme = TRUE
)

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

theme_gtsummary_mean_sd(set_theme = TRUE)

theme_gtsummary_eda(set_theme = TRUE)

Arguments

journal String indicating the journal theme to follow. One of c("jama", "lancet", "nejm", "qjecon").
Details below.

set_theme Logical indicating whether to set the theme. Default is TRUE. When FALSE the
named list of theme elements is returned invisibly.

font_size Numeric font size for compact theme. Default is 13 for gt tables, and 8 for all
other output types.

print_engine String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"

language String indicating language. Must be one of "de" (German), "en" (English),
"es" (Spanish), "fr" (French), "gu" (Gujarati), "hi" (Hindi), "is" (Icelandic), "ja"
(Japanese), "kr" (Korean), "mr" (Marathi), "pt" (Portuguese), "se" (Swedish),
"zh-c", "zh-tw" (Chinese Simplified)
If a language is missing a translation for a word or phrase, please feel free to
reach out on GitHub with the translated text!

decimal.mark The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
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 "1l 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 "1l 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 "1l to ul".
    * tbl_summary() Doesn’t show percent symbol; use em-dash to separate IQR
  - "qjecon" *The Quarterly Journal of Economics Under Development*
    * tbl_summary() all percentages rounded to one decimal place
    * tbl_regression()/tbl_uvregression() add significance stars with add_significance_stars(); hides CI and p-value from output
- theme_gtsummary_compact()
  - tables printed with gt, flextable, kableExtra, or huxtable will be compact with smaller font size and reduced cell padding
- theme_gtsummary_printer(print_engine=)
  - Use this theme to permanently change the default printer.
- theme_gtsummary_continuous2()
  - Set all continuous variables to summary type "continuous2" by default
- theme_gtsummary_mean_sd()
  - Set default summary statistics to mean and standard deviation in tbl_summary()
  - Set default continuous tests in add_p() to t-test and ANOVA
- theme_gtsummary_eda()
  - Set all continuous variables to summary type "continuous2" by default
  - In tbl_summary() show the median, mean, IQR, SD, and Range by default

Use reset_gtsummary_theme() to restore the default settings

Review the themes vignette to create your own themes.
Example Output

See Also

Themes vignette
set_gtsummary_theme(), reset_gtsummary_theme()

Examples

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

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

# reset gtsummary themes
reset_gtsummary_theme()

---

trial

Results from a simulated study of two chemotherapy agents

Description

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

Usage

trial

Format

A data frame with 200 rows–one row per patient

trt  Chemotherapy Treatment
age  Age
marker  Marker Level (ng/mL)
stage  T Stage
grade  Grade
response  Tumor Response
death  Patient Died
ttdeath  Months to Death/Censor
Index

* Advanced modifiers
  modify_cols_merge, 60
  modify_column_alignment, 61
  modify_column_hide, 62
  modify_fmt_fun, 63
  modify_table_body, 64
  modify_table_styling, 65
* datasets
  trial, 114
* gtsummary output types
  as_flex_table, 34
  as_gt, 36
  as_hux_table, 37
  as_kable, 38
  as_kable_extra, 39
  as_tibble.gtsummary, 42
* style tools
  style_number, 76
  style_percent, 77
  style_pvalue, 78
  style_ratio, 79
  style_siffig, 80
* tbl_continuous tools
  add_overall, 16
  add_p.tbl_continuous, 18
  tbl_continuous, 82
* tbl_cross tools
  add_p.tbl_cross, 20
  inline_text.tbl_cross, 50
  tbl_cross, 83
* tbl_custom_summary tools
  add_overall, 16
  continuous_summary, 46
  proportion_summary, 68
  ratio_summary, 70
  tbl_custom_summary, 85
* tbl_regression tools
  add_global_p, 9
  add_q, 26
  bold_italicize_labels_levels, 43
  combine_terms, 45
  inline_text.tbl_regression, 51
  modify, 57
  tbl_merge, 89
  tbl_regression, 91
  tbl_split, 93
  tbl_stack, 94
  tbl_strata, 96
* tbl_summary tools
  add_c1, 4
  add_n.tbl_summary, 11
  add_overall, 16
  add_p.tbl_summary, 21
  add_q, 26
  add_stat_label, 31
  bold_italicize_labels_levels, 43
  inline_text.tbl_summary, 52
  inline_text.tbl_survfit, 54
  modify, 57
  separate_p_footnotes, 73
  tbl_custom_summary, 85
  tbl_merge, 89
  tbl_split, 93
  tbl_stack, 94
  tbl_strata, 96
  tbl_summary, 99
* tbl_survfit tools
  add_n.tbl_survfit, 13
  add_nevent.tbl_survfit, 13
  add_p.tbl_survfit, 23
  modify, 57
  tbl_merge, 89
  tbl_split, 93
  tbl_stack, 94
  tbl_strata, 96
  tbl_survfit, 103
* tbl_svysummary tools
  add_n.tbl_summary, 11
  add_overall, 16
  add_p.tbl_svysummary, 24
  add_q, 26
  add_stat_label, 31
  modify, 57
  separate_p_footnotes, 73
  tbl_merge, 89
  tbl_split, 93
tbl_stack, 94
tbl_strata, 96
tbl_svysummary, 105

* tbl_uvregression tools
  add_global_p, 9
  add_q, 26
  bold_italicize_labels_levels, 43
  inline_text.tbl_uvregression, 56
  modify, 57
  tbl_merge, 89
  tbl_split, 93
  tbl_stack, 94
  tbl_strata, 96
  tbl_uvregression, 108

  add_ci, 4, 12, 17, 22, 27, 32, 44, 54, 55, 59, 74, 87, 90, 94, 95, 98, 102
  add_difference, 5
  add_glance, 7
  add_glance_source_note (add_glance), 7
  add_glance_table (add_glance), 7
  add_global_p, 9, 27, 44, 46, 52, 57, 59, 90, 93–95, 98, 111
  add_n.tbl_regression
    (add_n_regression), 15
  add_n.tbl_summary, 5, 11, 17, 18, 22, 26, 27, 32, 33, 44, 54, 55, 59, 74, 87, 90, 94, 95, 98, 102, 108
  add_n.tbl_survfit, 13, 14, 24, 59, 90, 94, 95, 98, 104
  add_n.tbl_svysummary
    (add_n_regression), 11
  add_n.tbl_uvregression
    (add_n_regression), 15
  add_n_regression, 15
  add_nevent.tbl_regression
    (add_nevent_regression), 14
  add_nevent.tbl_survfit, 13, 13, 24, 59, 90, 94, 95, 98, 104
  add_nevent.tbl_uvregression
    (add_nevent_regression), 14
  add_nevent_regression, 14
  add_overall, 5, 12, 16, 19, 22, 26, 27, 32, 33, 44, 47, 54, 55, 59, 69, 70, 74, 82, 87, 90, 94, 95, 98, 102, 108
  add_overall(), 86, 87
  add_p(), 87
  add_p.tbl_continuous, 18, 18, 82
  add_p.tbl_cross, 20, 51, 84
  add_p.tbl_summary, 5, 12, 17, 21, 27, 32, 44, 54, 55, 59, 74, 87, 90, 94, 95, 98, 102
  add_p.tbl_survfit, 13, 14, 23, 59, 90, 94, 95, 98, 104
  add_p.tbl_svysummary, 12, 18, 24, 27, 33, 59, 74, 90, 94, 95, 98, 108
  add_q, 5, 10, 12, 17, 18, 22, 26, 32, 33, 44, 46, 52, 54, 55, 57, 59, 74, 87, 90, 93–95, 98, 102, 108, 111
  add_significance_stars, 28
  add_stat, 29
  add_stat_label, 5, 12, 17, 18, 22, 26, 27, 31, 44, 54, 55, 59, 74, 87, 90, 94, 95, 98, 102, 108
  add_vif, 33
  all_categorical (select_helpers), 72
  all_continuous (select_helpers), 72
  all_continuous2 (select_helpers), 72
  all_contrasts (select_helpers), 72
  all_dichotomous (select_helpers), 72
  all_interaction (select_helpers), 72
  all_intercepts (select_helpers), 72
  all_stat_cols (select_helpers), 72
  all_tests (select_helpers), 72
  as_flex_table, 34, 36, 38, 39, 41, 42
  as_gt, 35, 36, 38, 39, 41, 42
  as_hux_table, 35, 36, 37, 39, 41, 42
  as_kable, 35, 36, 38, 39, 41, 42
  as_kable_extra, 35, 36, 38, 39, 41, 42
  as_table.gtsummary, 35, 36, 38, 39, 41, 42

  bold_italicize_labels_levels, 5, 10, 12, 17, 22, 27, 32, 44, 54, 55, 59, 74, 87, 90, 93–95, 98, 102, 111
  bold_labels
    (bold_italicize_labels_levels), 43
  bold_labels(), 39, 87
  bold_levels
    (bold_italicize_labels_levels), 43
  bold_p, 44

  combine_terms, 10, 27, 44, 45, 52, 59, 90, 93–95, 98
  continuous_summary, 18, 46, 69, 70, 87
  continuous_summary(), 85
  custom_tidiers, 47

  dplyr::tibble(), 86

  filter_p (sort_filter_p), 75

  glm, 109
  glue::glue, 12, 50, 51, 53, 56, 101, 106, 110
  glue::glue(), 59, 85, 87
  gt::gt, 36
gt::html(), 8, 58

gt::md(), 8, 58

gtsummary themes, 74

Hmisc::binconf(), 69

inline_text.gtsummary, 49

inline_text.tbl_regression, 10, 27, 44, 46, 51, 59, 90, 93–95, 98

inline_text.tbl_summary, 5, 12, 17, 22, 27, 32, 44, 52, 55, 59, 74, 87, 90, 94, 95, 98, 102

inline_text.tbl_survfit, 5, 12, 17, 22, 27, 32, 44, 54, 59, 74, 87, 90, 94, 95, 98, 102

inline_text.tbl_svysummary
(inline_text.tbl_summary), 52

italicize_labels
(bold_italicize_labels_levels), 43

italicize_levels
(bold_italicize_labels_levels), 43

italicize_levels(), 39

knitr::kable, 38–40

list, formula, and selector syntax, 5, 6, 9, 10, 12, 13, 22, 30, 32, 34, 46, 62, 64, 67, 71, 73, 74, 82, 84, 87, 93, 102, 104, 108, 111

lm, 109

modifications, 92, 110

modify, 5, 10, 12–14, 17, 18, 22, 24, 26, 27, 33, 44, 46, 52, 54, 55, 57, 74, 87, 90, 93–95, 98, 102, 104, 108, 111

modify_caption (modify), 57

modify_cols_merge, 60, 62, 64, 67

modify_column_alignment, 61, 61, 62, 64, 67

modify_column_hide, 61, 62, 62, 64, 67

modify_column_unhide
(modify_column_hide), 62

modify_fmt_fun, 61, 62, 63, 64, 67

modify_footnote (modify), 57

modify_footnote(), 87

modify_header (modify), 57

modify_spanning_header (modify), 57

modify_table_body, 61, 62, 64, 64, 67

modify_table_styling, 61, 62, 64, 65

plot, 67

pool_and_tidy_mice (custom_tidiers), 47

print.tbl_split (tbl_split), 93

proportion_summary, 18, 47, 68, 70, 87

proportion_summary(), 85

ratio_summary, 18, 47, 69, 70, 87

ratio_summary(), 85

remove_row_type, 71

reset_gtsummary_theme
(set_gtsummary_theme), 74

reset_gtsummary_theme(), 114

select_helpers, 72

separate_p_footnotes, 5, 12, 17, 18, 22, 26, 27, 33, 44, 54, 55, 59, 73, 87, 90, 94, 95, 98, 102, 108

set_gtsummary_theme, 74

set_gtsummary_theme(). 114

show_header_names (modify), 57

sort_filter_p, 75

sort_p (sort_filter_p), 75

stats::anova, 45

stats::anova(). 45

stats::p.adjust, 26

stats::poisson.test, 70

stats::prop.test, 69

stats::update, 45

style_number, 76, 77–80

style_percent, 76, 77, 78–80

style_percent(), 104

style_pvalue, 6, 19–21, 23, 25, 27, 50, 53, 55, 76, 77, 78, 79, 80, 92, 110

style_ratio, 76–78, 79, 80, 92, 110

style_siginig, 76–79, 80, 92, 104, 110

style_siginig(), 6, 34

survival::coxph, 109

syntax, 100, 108

tbl_butcher, 81

tbl_continuous, 17–19, 82

tbl_cross, 20, 51, 83

tbl_custom_summary, 5, 12, 17, 18, 22, 27, 33, 44, 47, 54, 55, 59, 69, 70, 74, 85, 90, 94, 95, 98, 102

tbl_custom_summary(), 46, 68, 70

tbl_merge, 5, 10, 12–14, 17, 18, 22, 24, 26, 27, 33, 44, 46, 52, 54, 55, 57, 59, 74, 87, 89, 93–95, 98, 102, 104, 108, 111

tbl_regression, 10, 27, 35–37, 39, 40, 42, 44, 46, 51, 52, 59, 90, 91, 94, 95, 98, 108
tbl_split. 5, 10, 12–14, 17, 18, 22, 24, 26, 27, 33, 44, 46, 52, 54, 55, 57, 59, 74, 87, 90, 93, 95, 98, 102, 104, 108, 111

tbl_stack. 5, 10, 12–14, 17, 18, 22, 24, 26, 27, 33, 44, 46, 52, 54, 55, 57, 59, 74, 87, 90, 93, 94, 98, 102, 104, 108, 111

tbl_strata. 5, 10, 12–14, 17, 18, 22, 24, 26, 27, 33, 44, 46, 52, 54, 55, 57, 59, 74, 87, 90, 93–95, 96, 102, 104, 108, 111

tbl_strata2(tbl_strata), 96

tbl_summary. 5, 12, 17, 19, 21, 22, 27, 32, 33, 35–37, 39, 40, 42, 44, 53–55, 59, 74, 87, 90, 94, 98, 99

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

tbl_survfit, 13, 14, 24, 54, 59, 90, 94, 95, 98, 103

tbl_survfit.data.frame(), 103

tbl_survfit.list(), 103

tbl_survfit.survfit(), 103

tbl_svysummary, 12, 17, 18, 25–27, 32, 33, 59, 74, 90, 94, 95, 98, 105

tbl_uvregression, 10, 27, 44, 56, 57, 59, 90, 94, 95, 98, 108

tests, 6, 19, 21, 22

theme_gtsummary, 111

theme_gtsummary_compact (theme_gtsummary), 111

theme_gtsummary_continuous2 (theme_gtsummary), 111

theme_gtsummary_eda (theme_gtsummary), 111

theme_gtsummary_journal (theme_gtsummary), 111

theme_gtsummary_language (theme_gtsummary), 111

theme_gtsummary_mean_sd (theme_gtsummary), 111

theme_gtsummary_printer (theme_gtsummary), 111

tibble, 42

tidy_bootstrap (custom_tidiers), 47

tidy_gam (custom_tidiers), 47

tidy_robust (custom_tidiers), 47

tidy_standardize (custom_tidiers), 47

trial, 114