Package 'crane'

January 21, 2026

Title Supplements the 'gtsummary' Package for Pharmaceutical Reporting

Version 0.3.1

Description Tables summarizing clinical trial results are often complex and require detailed tailoring prior to submission to a health authority. The 'crane' package supplements the functionality of the 'gtsummary' package for creating these often highly bespoke tables in the pharmaceutical industry.

License Apache License 2.0

URL https://github.com/insightsengineering/crane, https://insightsengineering.github.io/crane/

BugReports https://github.com/insightsengineering/crane/issues

Depends gtsummary (>= 2.5.0), R (>= 4.2)

Imports broom (>= 1.0.8), broom.helpers (>= 1.20.0), cards (>= 0.7.0), cardx (>= 0.3.0), cli (>= 3.6.4), cowplot (>= 1.2.0), dplyr (>= 1.1.4), flextable (>= 0.9.7), ggplot2 (>= 4.0.0), glue (>= 1.8.0), gt (>= 0.11.1), labeling, lifecycle, patchwork, rlang (>= 1.1.5), survival (>= 3.6-4), tidyr (>= 1.3.0)

Suggests ggtext, labelled, magick, parameters, pharmaverseadam, testthat (>= 3.0.0), webshot2, withr (>= 3.0.1)

Config/Needs/check hms

Config/Needs/website rmarkdown, yaml

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation no

Author Daniel D. Sjoberg [aut] (ORCID: <https://orcid.org/0000-0003-0862-2018>, note: Original creator of the package),

Emily de la Rua [aut] (ORCID: <https://orcid.org/0009-0000-8738-5561>),
Davide Garolini [aut] (ORCID: <https://orcid.org/0000-0002-1445-1369>),
Abinaya Yogasekaram [ctb] (ORCID:
<https://orcid.org/0009-0005-2083-1105>),
Joe Zhu [cre] (ORCID: <https://orcid.org/0000-0001-7566-2787>),
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Joe Zhu <joe.zhu@roche.com>

Repository CRAN

Date/Publication 2026-01-21 07:50:22 UTC

Contents

`add_blank_rows` *Add Blank Row*

Description

Add a blank row below each variable group defined by `variables` or below each specified `row_numbers`. A blank row will not be added to the bottom of the table.

NOTE: For HTML flextable output (which includes the RStudio IDE Viewer), the blank rows do not render. But they will appear when the table is rendered to Word.

Usage

```
add_blank_rows(x, variables = NULL, row_numbers = NULL, variable_level = NULL)
```

Arguments

`x`	(`gtsummary`) a 'gtsummary' table. The table must include a column named `'variable'` in `x$table_body`.
`variables`, `row_numbers`, `variable_level`	(`tidy-select` or `integer`) • `variables`: When a table contains variable summaries, use this argument to add blank rows below the specified variable block. • `row_numbers`: Add blank rows after each row number specified. • `variable_level`: A single column name in `x$table_body` and blank rows will be added after each unique level.

Value

updated 'gtsummary' table.

Examples

```
# Example 1 ----------------------------------
# Default to every variable used
trial |>
  tbl_roche_summary(
    by = trt,
    include = c(age, marker, grade),
    nonmissing = "always"
  ) |>
  add_blank_rows(variables = everything())

# Example 2 ----------------------------------
trial |>
  tbl_roche_summary(
    by = trt,
    include = c(age, marker, grade),
    nonmissing = "always"
  ) |>
  add_blank_rows(variables = age)
```

`add_hierarchical_count_row` *Add row with counts*

Description

Typically used to add a row with overall AE counts to a table that primarily displays AE rates.

Usage

```
add_hierarchical_count_row(
  x,
  label = "Overall total number of events",
  .before = NULL,
  .after = NULL,
  data_preprocess = identity
)
```

Arguments

`x`	(`gtsummary`) a gtsummary table
`label`	(`string`) label for the new row
`.before, .after`	(`integer`) Row index where to add the new row. Default is after last row.
`data_preprocess`	(`function` or `formula`) a function that is applied to `x$inputs$data` before the total row counts are tabulated. Default is `identity`. Tidyverse formula shortcut notation for the function is accepted. See `rlang::as_function()` for details.

Value

gtsummary table

Examples

```
# Example 1 ----------------------------------
cards::ADAE |>
  # subset the data for a shorter example table
  dplyr::slice(1:10) |>
  tbl_hierarchical(
    by = "TRTA",
    variables = AEDECOD,
    denominator = cards::ADSL,
    id = "USUBJID",
    overall_row = TRUE
  ) |>
  add_hierarchical_count_row(.after = 1L)
```

annotate_gg_km *Annotate Kaplan-Meier Plot*

Description

These functions provide capabilities to annotate Kaplan-Meier plots (gg_km()) with additional summary tables, including median survival times, numbers at risk, and cox proportional hazards results. The annotations are added using the cowplot package for flexible placement.

Usage

```
annotate_riskdf(
  gg_plt,
  fit_km,
  title = "Patients at Risk:",
  rel_height_plot = 0.75,
  xlab = "Days",
  ...
)

annotate_surv_med(gg_plt, fit_km, ...)

annotate_coxph(gg_plt, coxph_tbl, ...)
```

Arguments

gg_plt	(ggplot2 or cowplot) The primary plot object (either a ggplot2 or cowplot object) of the Kaplan-Meier plot.
fit_km	(survfit) A fitted Kaplan-Meier object of class survfit (from the survival package). This object contains the necessary survival data used to calculate and generate the content displayed in the annotation table.
title	(string) A single logical value indicating whether to include a above the table. Defaults to ""Patients at Risk:"". If NULL, no title is added.
rel_height_plot	(numeric) A single numeric value defining the **relative height** of the main Kaplan-Meier plot area compared to the 'at-risk' table. This value should be between 0 and 1, where a value closer to 1 gives the main plot more vertical space. Defaults to 0.75.
xlab	(character) A single character string for the **x-axis label** on the 'at-risk' table. This typically represents time (e.g., "Time (Days)").

... Additional arguments passed to the control list for the annotation box. These arguments override the default values. Accepted arguments include:

- x (numeric): X-coordinate for the box anchor position (0 to 1). Default is 0.29.
- y (numeric): Y-coordinate for the box anchor position (0 to 1). Default is 0.51.
- w (numeric): Width of the annotation box (0 to 1). Default is 0.4.
- h (numeric): Height of the annotation box (0 to 1). Default is 0.125.

coxph_tbl (data.frame)
A data frame containing the pre-calculated Cox-PH results, typically from a function like get_cox_pairwise_df. This data is used to generate the annotation table content.

Value

The function annotate_riskdf returns a cowplot object combining the KM plot and the 'Numbers at Risk' table.

The function annotate_surv_med returns a cowplot object with the median survival table annotation added, ready for final display or saving.

The function annotate_coxph returns a cowplot object with the Cox-PH table annotation added.

Functions

- annotate_riskdf(): The function annotate_riskdf adds a "Numbers at Risk" table below a Kaplan-Meier plot (gg_km()) using cowplot::plot_grid.
- annotate_surv_med(): The annotate_surv_med function adds a median survival time summary table as an annotation box.
- annotate_coxph(): The function annotate_coxph() adds a Cox Proportional Hazards summary table created by the function get_cox_pairwise_df() as an annotation box.

See Also

gg_km(), process_survfit(), and get_cox_pairwise_df() for related functionalities.

Examples

```
# Preparing the Kaplan-Meier Plot
use_lung <- survival::lung
use_lung$arm <- factor(sample(c("A", "B", "C"), nrow(use_lung), replace = TRUE))
use_lung$status <- use_lung$status - 1 # Convert status to 0/1
use_lung <- na.omit(use_lung)

formula <- survival::Surv(time, status) ~ arm
fit_kmg01 <- survival::survfit(formula, use_lung)
surv_plot_data <- process_survfit(fit_kmg01)

plt_kmg01 <- gg_km(surv_plot_data)
```

```
# Annotate Plot with Numbers at Risk Table
annotate_riskdf(plt_kmg01, fit_kmg01)

# Change order of y-axis (arm)
use_lung2 <- use_lung
use_lung2$arm <- factor(use_lung2$arm, levels = c("C", "B", "A"))
fit_kmg01 <- survival::survfit(formula, use_lung2)
annotate_riskdf(plt_kmg01, fit_kmg01) # rerun gg_km to change legend order

# Annotate Kaplan-Meier Plot with Median Survival Table
annotate_surv_med(plt_kmg01, fit_kmg01)

# Annotate Kaplan-Meier Plot with Cox-PH Table
coxph_tbl <- get_cox_pairwise_df(formula, data = use_lung, arm = "arm", ref_group = "A")
annotate_coxph(plt_kmg01, coxph_tbl)
```

`get_cox_pairwise_df` *Generate Table of Pairwise Cox-PH and Log-Rank Results*

Description

This function performs pairwise comparisons of treatment arms using the **Cox Proportional Hazards model** and calculates the corresponding **log-rank p-value**. Each comparison tests a non-reference group against a specified reference group.

Usage

```
get_cox_pairwise_df(model_formula, data, arm, ref_group = NULL)
```

Arguments

`model_formula`	(`formula`) A `formula` object specifying the survival model, typically in the form `Surv(time, status) ~ arm + covariates`.
`data`	(`data.frame`) A `data.frame` containing the survival data, including time, status, and the arm variable.
`arm`	(`character`) A single character string specifying the name of the column in `data` that contains the grouping/treatment **arm variable**. This column **must be a factor** for correct stratification and comparison.
`ref_group`	(`character` or `NULL`) A single character string specifying the level of the `arm` variable to be used as the **reference group** for all pairwise comparisons. If `NULL` (the default), the **first unique level** of the `arm` column is automatically selected as the reference group.

Details

The function iterates through each unique arm (excluding the reference group). For each iteration, it filters the data to include only the current comparison arm and the reference arm, and then:

- Fits a Cox model using `survival::coxph`.
- Performs a log-rank test using `survival::survdiff`.

The Hazard Ratio and its 95% confidence interval are extracted from the Cox model summary, and the p-value is extracted from the log-rank test.

Value

A `data.frame` with the results of the pairwise comparisons. The columns include:

- `arm`: (rownames of the `data.frame`) The comparison arm (group) being tested against the reference group.
- `hr`: The Hazard Ratio (HR) for the comparison arm vs. the reference arm, formatted to two decimal places.
- `ci`: The 95% confidence interval for the HR, presented as a string in the format "(lower, upper)", with values formatted to two decimal places.
- `pval`: The log-rank p-value for the comparison.

See Also

`annotate_gg_km()`, `gg_km()`, and the `survival` package functions `survival::coxph` and `survival::survdiff`.

Examples

```
# Example data setup (assuming 'time' is event time, 'status' is event indicator (1=event),
# and 'arm' is the treatment group)
library(dplyr) # For better data handling

# Prepare data in a modern dplyr-friendly way
surv_data <- survival::lung |>
  mutate(
    arm = factor(sample(c("A", "B", "C"), n(), replace = TRUE)),
    status = status - 1 # Convert status to 0/1
  ) |>
  filter(if_all(everything(), ~ !is.na(.)))

formula <- survival::Surv(time, status) ~ arm
results_tbl <- get_cox_pairwise_df(
  model_formula = formula,
  data = surv_data,
  arm = "arm",
  ref_group = "A"
)
print(results_tbl)
```

gg_km *Kaplan-Meier Plot*

Description

This set of functions facilitates the creation of Kaplan-Meier survival plots using ggplot2. Use process_survfit() to prepare the survival data from a fitted survfit object, and then gg_km() to generate the Kaplan-Meier plot with various customization options. Additional functions like annot_surv_med(), annot_cox_ph(), and annotate_riskdf() allow for adding summary tables and annotations to the plot.

Usage

```
process_survfit(fit_km, strata_levels = "All", max_time = NULL)

gg_km(
  surv_plot_data,
  lty = NULL,
  lwd = 0.5,
  censor_show = TRUE,
  size = 2,
  max_time = NULL,
  xticks = NULL,
  yval = c("Survival", "Failure"),
  ylim = NULL,
  font_size = 10,
  legend_pos = NULL
)
```

Arguments

fit_km	A fitted Kaplan-Meier object of class survfit.
strata_levels	(string) A single character string used as the strata level if the input fit_km object has no strata (e.g., "All").
max_time	(numeric) A single numeric value defining the **maximum time point** to display on the x-axis.
surv_plot_data	(data.frame) A data frame containing the pre-processed survival data, ready for plotting. This data should be equivalent to the output of process_survfit.
lty	(numeric or NULL) A numeric vector of **line types** (e.g., 1 for solid, 2 for dashed) for the survival curves, or NULL for ggplot2 defaults. The length should match the number of arms/groups.

lwd	(numeric) A single numeric value specifying the **line width** for the survival curves.
censor_show	(logical) A single logical value indicating whether to display **censoring marks** on the plot. Defaults to TRUE.
size	(numeric) A single numeric value specifying the **size** of the censoring marks.
xticks	(numeric or NULL) A numeric vector of explicit **x-axis tick positions**, or a single numeric value representing the **interval** between ticks, or NULL for automatic ggplot2 scaling.
yval	(character) A single character string, either "Survival" or "Failure" to plot the corresponding probability.
ylim	(numeric) A **numeric vector of length 2** defining the lower and upper limits of the y-axis (e.g., c(0, 1)).
font_size	(numeric) A single numeric value specifying the **base font size** for the plot theme elements.
legend_pos	(numeric or NULL) A **numeric vector of length 2** defining the **legend position** as (x, y) coordinates relative to the plot area (ranging from 0 to 1), or NULL for automatic placement.

Details

Data setup assumes "time" is event time, "status" is event indicator (1 represents an event), while "arm" is the treatment group.

Value

The function process_survfit returns a data frame containing the survival curve steps, confidence intervals, and censoring info.

The function gg_km returns a ggplot2 object of the KM plot.

Functions

- process_survfit(): takes a fitted survival::survfit object and processes it into a data frame suitable for plotting a Kaplan-Meier curve with ggplot2. Time zero is also added to the data.
- gg_km(): creates a Kaplan-Meier survival curve, with support for various customizations like censoring marks, Confidence Intervals (CIs), and axis control.

Examples

```
# Data preparation for KM plot
use_lung <- survival::lung
use_lung$arm <- factor(sample(c("A", "B", "C"), nrow(use_lung), replace = TRUE))
use_lung$status <- use_lung$status - 1 # Convert status to 0/1
use_lung <- na.omit(use_lung)
```

```
# Fit Kaplan-Meier model
formula <- survival::Surv(time, status) ~ arm
fit_kmg01 <- survival::survfit(formula, use_lung)

# Process survfit data for plotting
surv_plot_data <- process_survfit(fit_kmg01)
head(surv_plot_data)

# Example of making the KM plot
plt_kmg01 <- gg_km(surv_plot_data)

# Confidence Interval as Ribbon
plt_kmg01 +
  ggplot2::geom_ribbon(alpha = 0.3, lty = 0, na.rm = TRUE)

# Adding Title and Footnotes
plt_kmg01 +
  ggplot2::labs(title = "title", caption = "footnotes")

# Changing xlab and ylab
plt_kmg01 +
  ggplot2::xlab("Another Day") +
  ggplot2::ylab("THE Survival Probability")
```

`g_forest` *Create a Combined gtsummary Table and Forest Plot*

Description

This is the main wrapper function that takes a 'gtsummary' object, converts it to a 'ggplot' table, extracts the necessary data, creates a forest plot, and combines the two plots side-by-side using `+`. This likely relies on the `patchwork` package for plot combination.

Usage

```
g_forest(tbl)
```

Arguments

`tbl`	(`gtsummary`) A 'gtsummary' object (e.g., from `gtsummary::tbl_regression()`).

Value

A combined 'ggplot' object (likely a 'patchwork' object) showing the table on the left and the forest plot on the right.

See Also

extract_plot_data(), gg_forest_plot()

Examples

```
tbl <-
  trial |>
  tbl_roche_subgroups(
    rsp = "response",
    by = "trt",
    subgroups = c("grade", "stage"),
    ~ glm(response ~ trt, data = .x) |>
      gtsummary::tbl_regression(
        show_single_row = trt,
        exponentiate = TRUE
      )
  )
## Not run:
g_forest(tbl)

## End(Not run)
```

---

label_roche *Formatting percent and p-values*

Description

- label_roche_pvalue() returns formatted p-values.
- label_roche_percent() returns formatted percent values. This function only formats percentages between 0 and 1.
- label_roche_ratio() returns formatted ratios with values below and above a threshold being returned as < 0.1 and > 999.9, for example, when digits=1.
- label_roche_number() returns formatted numbers.

Usage

```
style_roche_pvalue(
  x,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  ...
)

label_roche_pvalue(
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
```

```
  ...
)

style_roche_percent(
  x,
  digits = 1,
  prefix = "",
  suffix = "",
  scale = 100,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  ...
)

label_roche_percent(
  digits = 1,
  suffix = "",
  scale = 100,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  ...
)

style_roche_ratio(
  x,
  digits = 2,
  prefix = "",
  suffix = "",
  scale = 1,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  ...
)

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

style_roche_number(
  x,
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
```

```
  decimal.mark = getOption("OutDec"),
  scale = 1,
  prefix = "",
  suffix = "",
  na = "NE",
  inf = "NE",
  nan = "NE",
  ...
)

label_roche_number(
  digits = 0,
  big.mark = ifelse(decimal.mark == ",", " ", ","),
  decimal.mark = getOption("OutDec"),
  scale = 1,
  prefix = "",
  suffix = "",
  na = "NE",
  inf = "NE",
  nan = "NE",
  ...
)
```

Arguments

x	(numeric) Numeric vector
big.mark	(string) Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ",", except when decimal.mark = "," when the default is a space.
decimal.mark	(string) The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
...	Arguments passed on to base::format()
digits	(non-negative integer) Integer or vector of integers specifying the number of decimals to round x. When vector is passed, each integer is mapped 1:1 to the numeric values in x
prefix	(string) Additional text to display before the number.
suffix	(string) Additional text to display after the number.
scale	(scalar numeric) A scaling factor: x will be multiplied by scale before formatting.
na, inf, nan	(NA/string) scalar to replace NA, infinite, and NaN values with. Default is "NE" for arguments na, inf, and nan argument.

Value

A character vector of rounded p-values

Examples

```
# p-value formatting
x <- c(0.0000001, 0.123456)

style_roche_pvalue(x)
label_roche_pvalue()(x)

# percent formatting
x <- c(0.0008, 0.9998)

style_roche_percent(x)
label_roche_percent()(x)

# ratio formatting
x <- c(0.0008, 0.8234, 2.123, 1000)

style_roche_ratio(x)
label_roche_ratio()(x)

# number formatting
x <- c(0.0008, 0.8234, 2.123, 1000, NA, Inf, -Inf)

style_roche_number(x)
label_roche_number()(x)
```

---

modify_header_rm_md *Remove Markdown Syntax from Header*

Description

Remove markdown syntax (e.g. double star for bold, underscore for italic, etc) from the headers and spanning headers of a gtsummary table.

Usage

```
modify_header_rm_md(x, md = "bold", type = "star")
```

Arguments

x	(gtsummary) A gtsummary table
md	(character) Must be one or more of 'bold' and 'italic'. Default is 'bold'.
type	(character) Must be one or more of 'star' and 'underscore'. Default is 'star'.

Value

gtsummary table

Examples

```
tbl_roche_summary(
  data = cards::ADSL,
  include = AGE,
  by = ARM,
  nonmissing = "always"
) |>
  modify_header_rm_md()
```

---

modify_zero_recode *Zero Count Recode*

Description

This function removes the percentage from cells with zero counts. For example,

```
0 (0.0%)      -->  0
0 (0%)        -->  0
0 (NA%)       -->  0
0 / nn (0%)   -->  0 / nn
0/nn (0.0%)   -->  0/nn
0 / 0 (NA%)   -->  0 / 0
```

Usage

```
modify_zero_recode(x)
```

Arguments

x	(gtsummary) a gtsummary table

Details

The function is a wrapper for gtsummary::modify_post_fmt_fun().

```
gtsummary::modify_post_fmt_fun(
  x,
  fmt_fun = \(x) {
    dplyr::case_when(
      # convert "0 (0%)" OR "0 (0.0%)" OR 0 (NA%) to "0"
    str_detect(x, "^0\\s\\((?:0(?:\\.0)?|NA)%\\)$") ~ str_remove(x, pattern = "\\s\\((?:0(?:\\.0)?|NA
    # convert "0 / nn (0%)" OR "0/nn (0.0%)" OR 0/0 (NA%) to "0 / nn" OR "0/nn" OR "0/0"
```

```
      str_detect(x, pattern = "^(0 ?/) ?\\d+[^()]* \\((?:0(?:\\.0)?|NA)%\\)$") ~ str_remove(x, pattern =
        .default = x
      )
    },
    columns = gtsummary::all_stat_cols()
  )
```

Value

a gtsummary table

Examples

```
trial |>
  dplyr::mutate(trt = factor(trt, levels = c("Drug A", "Drug B", "Drug C"))) |>
  tbl_summary(include = trt) |>
  modify_zero_recode()
```

---

tbl_baseline_chg *Change from Baseline*

Description

Typical use is tabulating changes from baseline measurement of an Analysis Variable.

Usage

```
tbl_baseline_chg(
  data,
  baseline_level,
  denominator,
  by = NULL,
  digits = NULL,
  id = "USUBJID",
  visit = "AVISIT",
  visit_number = "AVISITN",
  analysis_variable = "AVAL",
  change_variable = "CHG"
)

## S3 method for class 'tbl_baseline_chg'
add_overall(
  x,
  last = FALSE,
  col_label = "All Participants  \n(N = {style_roche_number(n)})",
  ...
)
```

Arguments

data	(data.frame) A data frame.
baseline_level	(string) String identifying baseline level in the visit variable.
denominator	(string) Data set used to compute the header counts (typically ADSL).
by	(tidy-select) A single column from data. Summary statistics will be stratified by this variable. Default is NULL.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits(). See below for details.
id	(string) String identifying the unique subjects. Default is 'USUBJID'.
visit	(string) String for the visit variable. Default is 'AVISIT'. If there are more than one entry for each visit and subject, only the first row is kept.
visit_number	(string) String identifying the visit or analysis sequence number. Default is 'AVISITN'.
analysis_variable	(string) String identifying the analysis values. Default is 'AVAL'.
change_variable	(string) String identifying the change from baseline values. Default is 'CHG'.
x	(tbl_summary, tbl_svysummary, tbl_continuous, tbl_custom_summary) A stratified 'gtsummary' table
last	(scalar logical) Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.
col_label	(string) String indicating the column label. Default is "**Overall** \nN = {style_number(N)}"
...	These dots are for future extensions and must be empty.

Value

a gtsummary table

Examples

```
theme_gtsummary_roche()

df <- cards::ADLB |>
```

```
  dplyr::mutate(AVISIT = trimws(AVISIT)) |>
  dplyr::filter(
    AVISIT != "End of Treatment",
    PARAMCD == "SODIUM"
  )

tbl_baseline_chg(
  data = df,
  baseline_level = "Baseline",
  by = "TRTA",
  denominator = cards::ADSL
)

tbl_baseline_chg(
  data = df,
  baseline_level = "Baseline",
  by = "TRTA",
  denominator = cards::ADSL
) |>
  add_overall(last = TRUE, col_label = "All Participants")
```

`tbl_hierarchical_rate_and_count` *Hierarchical Rates and Counts*

Description

A mix of adverse event rates (from `gtsummary::tbl_hierarchical()`) and counts (from `gtsummary::tbl_hierarchical_`
The function produces additional summary rows for the higher level nesting variables providing both rates and counts.

When a hierarchical summary is filtered, the summary rows no longer provide useful/consistent information. When creating a filtered summary, use `gtsummary::tbl_hierarchical()` or `gtsummary::tbl_hierarchical_` directly, followed by a call to `gtsummary::filter_hierarchical()`.

Usage

```
tbl_hierarchical_rate_and_count(
  data,
  variables,
  denominator,
  by = NULL,
  id = "USUBJID",
  label = NULL,
  digits = NULL,
  sort = NULL,
 label_overall_rate = "Total number of participants with at least one adverse event",
  label_overall_count = "Overall total number of events",
```

```
  label_rate = "Total number of participants with at least one adverse event",
  label_count = "Total number of events"
)

## S3 method for class 'tbl_hierarchical_rate_and_count'
add_overall(
  x,
  last = FALSE,
  col_label = "All Participants  \n(N = {style_roche_number(N)})",
  ...
)
```

Arguments

data	(data.frame) a data frame.
variables	(tidy-select) Hierarchical variables to summarize. Must be 2 or 3 variables. Typical inputs are c(AEBODSYS, AEDECOD) for an SOC/AE summary or c(AEBODSYS, AEHLT, AEDECOD) for an SOC/HLT/AE summary. Variables must be specified in the nesting order.
denominator	(data.frame, integer) used to define the denominator and enhance the output. The argument is required for tbl_hierarchical() and optional for tbl_hierarchical_count(). The denominator argument must be specified when id is used to calculate event rates.
by	(tidy-select) a single column from data. Summary statistics will be stratified by this variable. Default is NULL.
id	(tidy-select) argument used to subset data to identify rows in data to calculate event rates in tbl_hierarchical().
label	(formula-list-selector) used to override default labels in hierarchical table, e.g. list(AESOC = "System Organ Class"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If a theme is applied, the digits specifications of the theme is applied.
sort	*Optional* arguments passed to gtsummary::sort_hierarchical(sort).
label_overall_rate	(string) String for the overall rate summary. Default is "Total number of participants with at least one adverse event".

label_overall_count
: (string)
String for the overall count summary. Default is "Overall total number of events".

label_rate
: (string)
String for the rate summary. Default is "Overall total number of events". "Total number of participants with at least one adverse event".

label_count
: (string)
String for the overall count summary. Default is "Total number of events".

x
: (tbl_hierarchical_rate_and_count)
a stratified 'tbl_hierarchical_rate_and_count' table

last
: (scalar logical)
Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.

col_label
: (string)
String indicating the column label. Default is "**Overall** \nN = {style_number(N)}"

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

Value

a gtsummary table

Examples

```
# Example 1 ----------------------------------
cards::ADAE[c(1, 2, 3, 8, 16), ] |>
  tbl_hierarchical_rate_and_count(
    variables = c(AEBODSYS, AEDECOD),
    denominator = cards::ADSL,
    by = TRTA
  ) |>
  add_overall(last = TRUE)
```

---

tbl_hierarchical_rate_by_grade

AE Rates by Highest Toxicity Grade

Description

A wrapper function for gtsummary::tbl_hierarchical() to calculate rates of highest toxicity grades with the options to add rows for grade groups and additional summary sections at each variable level.

Only the highest grade level recorded for each subject will be analyzed. Prior to running the function, ensure that the toxicity grade variable (grade) is a factor variable, with factor levels ordered lowest to highest.

Grades will appear in rows in the order of the factor levels given, with each grade group appearing prior to the first level in its group.

Usage

```
tbl_hierarchical_rate_by_grade(
  data,
  variables,
  denominator,
  by = NULL,
  id = "USUBJID",
  include_overall = everything(),
  statistic = everything() ~ "{n} ({p}%)",
  label = NULL,
  digits = NULL,
  sort = "alphanumeric",
  filter = NULL,
  grade_groups = list(),
  grades_exclude = NULL,
  keep_zero_rows = FALSE
)

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

Arguments

data	(data.frame) a data frame.
variables	(tidy-select) A character vector or tidy-selector of 3 columns in data specifying a system organ class variable, an adverse event terms variable, and a toxicity grade level variable, respectively.
denominator	(data.frame, integer) used to define the denominator and enhance the output. The argument is required for tbl_hierarchical() and optional for tbl_hierarchical_count(). The denominator argument must be specified when id is used to calculate event rates.
by	(tidy-select) a single column from data. Summary statistics will be stratified by this variable. Default is NULL.
id	(tidy-select) argument used to subset data to identify rows in data to calculate event rates in tbl_hierarchical().

include_overall
: (tidy-select)
Variables from variables for which an overall section at that hierarchy level should be computed. An overall section at the SOC variable level will have label "- Any adverse events -". An overall section at the AE term variable level will have label "- Overall -". If the grade level variable is included it has no effect. The default is everything().

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

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

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

sort
: (formula-list-selector, string)
a named list, a list of formulas, a single formula where the list element is a named list of functions (or the RHS of a formula), or a string specifying the types of sorting to perform at each hierarchy level. If the sort method for any variable is not specified then the method will default to "descending". If a single unnamed string is supplied it is applied to all hierarchy levels. For each variable, the value specified must be one of:

- "alphanumeric" - at the specified hierarchy level, groups are ordered alphanumerically (i.e. A to Z) by variable_level text.
- "descending" - at the specified hierarchy level, count sums are calculated for each row and rows are sorted in descending order by sum. If sort is "descending" for a given variable and n is included in statistic for the variable then n is used to calculate row sums, otherwise p is used. If neither n nor p are present in x for the variable, an error will occur.

Defaults to everything() ~ "descending".

filter
: (expression)
An expression that is used to filter rows of the table. Filter will be applied to the second variable (adverse event terms) specified via variables. See the Details section below for more information.

grade_groups
: (named list)
A named list of grade groups for which rates should be calculated. Grade groups must be mutually exclusive, i.e. each grade cannot be assigned to more than one grade group. Each grade group must be specified in the list as a character vector of the grades included in the grade group, named with the corresponding name of the grade group, e.g. "Grade 1-2" = c("1", "2").

grades_exclude
: (character)
A vector of grades to omit individual rows for when printing the table. These grades will still be used when computing overall totals and grade group totals.

For example, to avoid duplication, if a grade group is defined as `"Grade 5" = "5"`, the individual rows corresponding to grade 5 can be excluded by setting `grades_exclude = "5"`.

`keep_zero_rows` (`logical`)
Whether rows containing zero rates across all columns should be kept. If `FALSE`, this filter will be applied prior to any filters specified via the `filter` argument which may still remove these rows. Defaults to `FALSE`.

`x` (`tbl_hierarchical_rate_by_grade`)
A gtsummary table of class `'tbl_hierarchical_rate_by_grade'`.

`last` (scalar `logical`)
Logical indicator to display overall column last in table. Default is `FALSE`, which will display overall column first.

`col_label` (`string`)
String indicating the column label. Default is `"**Overall** \nN = {style_number(N)}"`

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

Details

When using the `filter` argument, the filter will be applied to the second variable from `variables`, i.e. the adverse event terms variable. If an AE does not meet the filtering criteria, the AE overall row as well as all grade and grade group rows within an AE section will be excluded from the table. Filtering out AEs does not exclude the records corresponding to these filtered out rows from being included in rate calculations for overall sections. If all AEs for a given SOC have been filtered out, the SOC will be excluded from the table. If all AEs are filtered out and the SOC variable is included in `include_overall` the `- Any adverse events -` section will still be kept.

See `gtsummary::filter_hierarchical()` for more details and examples.

Value

a gtsummary table of class `"tbl_hierarchical_rate_by_grade"`.

Examples

```
theme_gtsummary_roche()
ADSL <- cards::ADSL
ADAE_subset <- cards::ADAE |>
  dplyr::filter(
    AESOC %in% unique(cards::ADAE$AESOC)[1:5],
    AETERM %in% unique(cards::ADAE$AETERM)[1:10]
  )

grade_groups <- list(
  "Grade 1-2" = c("1", "2"),
  "Grade 3-4" = c("3", "4"),
  "Grade 5" = "5"
)

# Example 1 ----------------------------------
tbl_hierarchical_rate_by_grade(
```

```
  ADAE_subset,
  variables = c(AEBODSYS, AEDECOD, AETOXGR),
  denominator = ADSL,
  by = TRTA,
  label = list(
    AEBODSYS = "MedDRA System Organ Class",
    AEDECOD = "MedDRA Preferred Term",
    AETOXGR = "Grade"
  ),
  grade_groups = grade_groups,
  grades_exclude = "5"
)

# Example 2 ---------------------------------
# Filter: Keep AEs with an overall prevalence of greater than 10%
tbl_hierarchical_rate_by_grade(
  ADAE_subset,
  variables = c(AEBODSYS, AEDECOD, AETOXGR),
  denominator = ADSL,
  by = TRTA,
  grade_groups = list("Grades 1-2" = c("1", "2"), "Grades 3-5" = c("3", "4", "5")),
  filter = sum(n) / sum(N) > 0.10
) |>
  add_overall(last = TRUE)
```

`tbl_listing` *Create listings from a data frame*

Description

This function creates a listing from a data frame. Common uses rely on few pre-processing steps, such as ensuring unique values in key columns or split by rows or columns. They are described in the note section.

Usage

```
tbl_listing(
  data,
  split_by_rows = list(),
  split_by_columns = list(),
  add_blank_rows = list()
)

remove_duplicate_keys(x, keys = NULL, value = NA)
```

Arguments

`data`	(`data.frame`) a data frame containing the data to be displayed in the listing.

split_by_rows, split_by_columns, add_blank_rows
: (named list)

  - split_by_rows: Named list of arguments that are passed to gtsummary::tbl_split_by_rows().
  - split_by_columns: Named list of arguments that are passed to gtsummary::tbl_split_by_colum
  - add_blank_rows: Named list of arguments that are passed to crane::add_blank_rows(). add_blank_rows() is applied after table splitting and applied to each table individually.

  Variable names passed in these named lists must be character vectors; tidyselect/unquoted syntax is not accepted.

x
: (tbl_listing or list)
a tbl_listing object or a list of tbl_listing objects.

keys
: (tidy-select)
columns to highlight for duplicate values. If NULL, nothing is done.

value
: (string)
string to use for blank values. Defaults to NA. It should not be changed.

Note

Common pre-processing steps for the data frame that may be common:

- Unique values - this should be enforced in pre-processing by users.
- NA values - they are not printed by default in {gtsummary}. You can make them explicit if they need to be displayed in the listing. See example 3.
- Sorting key columns and moving them to the front. See the examples pre-processing.

Splitting the listing:

- Split by rows - you can split the data frame by rows by using split_by_rows parameter. You can use the same parameters used in gtsummary::tbl_split_by_rows(). See example 4.
- Split by columns - you can split the data frame by columns by using split_by_columns parameter. Use the same parameters from gtsummary::tbl_split_by_rows(). See example 5.

Examples

```
# Load the trial dataset
trial_data <- trial |>
  dplyr::select(trt, age, marker, stage) |>
  dplyr::filter(stage %in% c("T2", "T3")) |>
  dplyr::slice_head(n = 2, by = c(trt, stage)) |> # downsampling
  dplyr::arrange(trt, stage) |> # key columns should be sorted
  dplyr::relocate(trt, stage) # key columns should be first

# Example 1 ---------------------------------
out <- tbl_listing(trial_data)
out
out |> remove_duplicate_keys(keys = "trt")
```

```
# Example 2 --------------------------------
# make NAs explicit
trial_data_na <- trial_data |>
  mutate(across(everything(), ~ tidyr::replace_na(labelled::to_character(.), "-")))
tbl_listing(trial_data_na)

# Example 3 --------------------------------
# Add blank rows for first key column
lst <- tbl_listing(trial_data_na, add_blank_rows = list(variable_level = "trt"))
lst

# Can add them also manually in post-processing
lst |> add_blank_rows(row_numbers = seq(2))

# Example 4 --------------------------------
# Split by rows
list_lst <- tbl_listing(trial_data, split_by_rows = list(row_numbers = c(2, 3, 4)))
list_lst[[2]]

# Example 5 --------------------------------
# Split by columns
show_header_names(lst)
grps <- list(c("trt", "stage", "age"), c("trt", "stage", "marker"))
list_lst <- tbl_listing(trial_data, split_by_columns = list(groups = grps))
list_lst[[2]]

# Example 6 --------------------------------
# Split by rows and columns
list_lst <- tbl_listing(trial_data,
  split_by_rows = list(row_numbers = c(2, 3, 4)), split_by_columns = list(groups = grps)
)
length(list_lst) # 8 tables are flatten out
list_lst[[2]]

# Example 7 --------------------------------
# Hide duplicate columns in post-processing
out <- list_lst |>
  remove_duplicate_keys(keys = c("trt", "stage"))
out[[2]]
```

`tbl_null_report` *Creates null report*

Description

This function creates a null report for tables or listings without any statistics.

Usage

```
tbl_null_report(
  label = "No observations met the reporting criteria for this output."
)
```

Arguments

label	(string) label to display in the header of the null report. It defaults to "No observations met the reporting criteria for this output."

Examples

```
tbl_null_report(label = "No data available for the selected criteria.")
```

---

tbl_roche_subgroups *Subgroup Analyses*

Description

Function adapted from `gtforester::tbl_subgroups()`.

Usage

```
tbl_roche_subgroups(data, rsp, by, subgroups, .tbl_fun)
```

Arguments

data	(data.frame, survey.design) a data frame or survey object
rsp	(tidy-select) Variable to use in responder rate calculations.
by	(tidy-select) Variable to make comparison between groups.
subgroups	(tidy-select) Variables to perform stratified analyses for.
.tbl_fun	(function) A function or formula. If a *function*, it is used as is. If a formula, e.g. `~ .x %>% tbl_summary() %>% add_p()`, it is converted to a function. The stratified data frame is passed to this function.

Value

a 'gtsummary' table

Examples

```
tbl <-
  trial |>
  tbl_roche_subgroups(
    rsp = "response",
    by = "trt",
    subgroups = c("grade", "stage"),
    .tbl_fun =
      ~ glm(response ~ trt, data = .x) |>
        tbl_regression(
          show_single_row = trt,
          exponentiate = TRUE
        )
  )

tbl
```

`tbl_roche_summary` *Roche Summary Table*

Description

This is a thin wrapper of `gtsummary::tbl_summary()` with the following differences:

- Default summary type for continuous variables is `'continuous2'`.
- Number of non-missing observations, when requested, is added for each variable and placed on the row under the variable label/header.
- The `tbl_summary(missing*)` arguments have been renamed to `tbl_roche_summary(nonmissing*)` with updated default values.
- The default footnotes from `tbl_summary()` are removed.
- Cells with `"0 (0.0%)"` are converted to `"0"` with `gtsummary::modify_post_fmt_fun()`.

Usage

```
tbl_roche_summary(
  data,
  by = NULL,
  label = NULL,
  statistic = list(gtsummary::all_continuous() ~ c("{mean} ({sd})", "{median}",
    "{min} - {max}"), gtsummary::all_categorical() ~ "{n} ({p}%)"),
  digits = NULL,
  type = NULL,
  value = NULL,
  nonmissing = c("no", "always", "ifany"),
  nonmissing_text = "n",
  nonmissing_stat = "{N_nonmiss}",
```

```
  sort = gtsummary::all_categorical(FALSE) ~ "alphanumeric",
  percent = c("column", "row", "cell"),
  include = everything()
)
```

Arguments

data	(data.frame) A data frame.
by	(tidy-select) A single column from data. Summary statistics will be stratified by this variable. Default is NULL.
label	(formula-list-selector) Used to override default labels in summary table, e.g. list(age = "Age, years"). The default for each variable is the column label attribute, attr(., 'label'). If no label has been set, the column name is used.
statistic	(formula-list-selector) Specifies summary statistics to display for each variable. The default is list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)"). See below for details.
digits	(formula-list-selector) Specifies how summary statistics are rounded. Values may be either integer(s) or function(s). If not specified, default formatting is assigned via assign_summary_digits(). See below for details.
type	(formula-list-selector) Specifies the summary type. Accepted value are c("continuous", "continuous2", "categorical", "dichotomous"). If not specified, default type is assigned via assign_summary_type(). See below for details.
value	(formula-list-selector) Specifies the level of a variable to display on a single row. The gtsummary type selectors, e.g. all_dichotomous(), cannot be used with this argument. Default is NULL. See below for details.
nonmissing, nonmissing_text, nonmissing_stat	Arguments dictating how and if missing values are presented: • nonmissing: must be one of c("always", "ifany", "no") • nonmissing_text: string indicating text shown on non-missing row. Default is "n" • nonmissing_stat: statistic to show on non-missing row. Default is "{N_nonmiss}". Possible values are N_nonmiss, N_miss, N_obs, p_nonmiss p_miss.
sort	(formula-list-selector) Specifies sorting to perform for categorical variables. Values must be one of c("alphanumeric", "frequency"). Default is all_categorical(FALSE) ~ "alphanumeric".
percent	(string) Indicates the type of percentage to return. Must be one of c("column", "row", "cell"). Default is "column".

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

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

Value

a 'gtsummary' table

Examples

```
# Example 1 ---------------------------------
trial |>
  tbl_roche_summary(
    by = trt,
    include = c(age, grade),
    nonmissing = "always"
  ) |>
  add_overall()
```

`tbl_shift` *Shift Table*

Description

Typical use is tabulating post-baseline measurement stratified by the baseline measurement.

Usage

```
tbl_shift(
  data,
  variable,
  strata = NULL,
  by = NULL,
  data_header = NULL,
  strata_location = c("new_column", "header"),
  strata_label = "{strata}",
  header = "{level}  \nN = {n}",
  label = NULL,
  nonmissing = "always",
  nonmissing_text = "Total",
  ...
)

## S3 method for class 'tbl_shift'
add_overall(
```

```
  x,
  col_label = "All Participants  \n(N = {style_roche_number(n)})",
  last = FALSE,
  ...
)
```

Arguments

data	(data.frame) A data frame.
variable	(tidy-select) Variable to tabulate. Typically the post-baseline grade.
strata	(tidy-select) Stratifying variable. Typically the baseline grade.
by	(tidy-select) Variable to report results by. Typical value is the treatment arm.
data_header	(data.frame) Data frame used to calculate the Ns in the table header. Only include the columns needed to merge with data: these are typically the 'USUBJID' and the treatment arm only, e.g ADSL[c("USUBJID", "ARM")].
strata_location	(string) Specifies the location where the individual stratum levels will be printed. Must be one of c("new_column", "header"). "new_column": stratum labels are placed in a new column to the left of the tabulated results. "header": stratum labels are placed in a header row above the tabulations.
strata_label	(string) A glue-string that inserts stratum level. Default is '{strata}', and {n} is also available to insert.
header	(string) String that is passed to gtsummary::modify_header(all_stat_cols() ~ header).
label	(formula-list-selector) Used to specify the labels for the strata and variable columns. Default is to use the column label attribute.
nonmissing, nonmissing_text, ...	Argument passed to tbl_roche_summary(). See details below for call details to tbl_roche_summary().
x	(tbl_shift) Object of class 'tbl_shift'.
col_label	(string) String indicating the column label. Default is "All Participants \nN = {style_roche_number(n)}"
last	(scalar logical) Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.

Details

Broadly, this function is a wrapper for chunk below with some additional calls to `gtsummary::modify_*()` function to update the table's headers, indentation, column alignment, etc.

```
gtsummary::tbl_strata2(
  data = data,
  strata = strata,
   ~ tbl_roche_summary(.x, include = variable, by = by)
)
```

Value

a 'gtsummary' table

Examples

```
library(dplyr, warn.conflicts = FALSE)

# subsetting ADLB on one PARAM, and the highest grade
adlb <- pharmaverseadam::adlb |>
  select("USUBJID", "TRT01A", "PARAM", "PARAMCD", "ATOXGRH", "BTOXGRH", "VISITNUM") |>
  mutate(TRT01A = factor(TRT01A)) |>
  filter(PARAMCD %in% c("CHOLES", "GLUC")) |>
  slice_max(by = c(USUBJID, PARAMCD), order_by = ATOXGRH, n = 1L, with_ties = FALSE) |>
  labelled::set_variable_labels(
    BTOXGRH = "Baseline  \nNCI-CTCAE Grade",
    ATOXGRH = "Post-baseline  \nNCI-CTCAE Grade"
  )
adsl <- pharmaverseadam::adsl[c("USUBJID", "TRT01A")] |>
  filter(TRT01A != "Screen Failure")

# Example 1 ---------------------------------
# tabulate baseline grade by worst grade
tbl_shift(
  data = filter(adlb, PARAMCD %in% "CHOLES"),
  strata = BTOXGRH,
  variable = ATOXGRH,
  by = TRT01A,
  data_header = adsl
)

# Example 2 ---------------------------------
# same as Ex1, but with the stratifying variable levels in header rows
adlb |>
  filter(PARAMCD %in% "CHOLES") |>
  labelled::set_variable_labels(
    BTOXGRH = "Baseline NCI-CTCAE Grade",
    ATOXGRH = "Post-baseline NCI-CTCAE Grade"
  ) |>
  tbl_shift(
    data = ,
```

```
    strata = BTOXGRH,
    variable = ATOXGRH,
    strata_location = "header",
    by = TRT01A,
    data_header = adsl
  )

# Example 3 ----------------------------------
# same as Ex2, but with two labs
adlb |>
  labelled::set_variable_labels(
    BTOXGRH = "Baseline NCI-CTCAE Grade",
    ATOXGRH = "Post-baseline NCI-CTCAE Grade"
  ) |>
  tbl_strata_nested_stack(
    strata = PARAM,
    ~ .x |>
      tbl_shift(
        strata = BTOXGRH,
        variable = ATOXGRH,
        strata_location = "header",
        by = TRT01A,
        data_header = adsl
      )
  ) |>
  # Update header with Lab header and indentation (the '\U00A0' character adds whitespace)
  modify_header(
    label = "Lab  \n\U00A0\U00A0\U00A0\U00A0
            Baseline NCI-CTCAE Grade  \n\U00A0\U00A0\U00A0\U00A0\U00A0\U00A0\U00A0\U00A0
             Post-baseline NCI-CTCAE Grade"
  )

# Example 4 ----------------------------------
# Include the treatment variable in a new column
filter(adlb, PARAMCD %in% "CHOLES") |>
  right_join(
    pharmaverseadam::adsl[c("USUBJID", "TRT01A")] |>
      filter(TRT01A != "Screen Failure"),
    by = c("USUBJID", "TRT01A")
  ) |>
  tbl_shift(
    strata = TRT01A,
    variable = BTOXGRH,
    by = ATOXGRH,
    header = "{level}",
    strata_label = "{strata}, N={n}",
    label = list(TRT01A = "Actual Treatment"),
    percent = "cell",
    nonmissing = "no"
  ) |>
  modify_spanning_header(all_stat_cols() ~ "Worst Post-baseline NCI-CTCAE Grade")
```

`tbl_survfit_quantiles` *Survival Quantiles*

Description

Create a gtsummary table with Kaplan-Meier estimated survival quantiles. If you must further customize the way these results are presented, see the Details section below for the full details.

Usage

```
tbl_survfit_quantiles(
  data,
  y = "survival::Surv(time = AVAL, event = 1 - CNSR, type = 'right', origin = 0)",
  by = NULL,
  header = "Time to event",
  estimate_fun = label_roche_number(digits = 1, na = "NE"),
  method.args = list(conf.int = 0.95)
)

## S3 method for class 'tbl_survfit_quantiles'
add_overall(
  x,
  last = FALSE,
  col_label = "All Participants  \nN = {style_roche_number(N)}",
  ...
)
```

Arguments

`data`	(`data.frame`) A data frame
`y`	(`string` or `expression`) A string or expression with the survival outcome, e.g. `survival::Surv(time, status)`. The default value is `survival::Surv(time = AVAL, event = 1 - CNSR, type = "right", origin = 0)`.
`by`	(`tidy-select`) A single column from `data`. Summary statistics will be stratified by this variable. Default is `NULL`, which returns results for the unstratified model.
`header`	(`string`) String for the header of the survival quantile chunks. Default is `"Time to event"`.
`estimate_fun`	(`function`) Function used to round and format the estimates in the table. Default is `label_roche_number(digits = 1)`.
`method.args`	(named `list`) Named list of arguments that will be passed to `survival::survfit()`.

Note that this list may contain non-standard evaluation components, and must be handled similarly to tidyselect inputs by using rlang's embrace operator `{{ . }}` or `!!enquo()` when programming with this function.

`x` (`tbl_survfit_quantiles`)
A stratified 'tbl_survfit_quantiles' object.

`last` (scalar `logical`)
Logical indicator to display overall column last in table. Default is `FALSE`, which will display overall column first.

`col_label` (`string`)
String indicating the column label. Default is `"**Overall** \nN = {style_number(N)}"`

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

Value

a gtsummary table

ARD-first

This function is a helper for creating a common summary. But if you need to modify the appearance of this table, you may need to build it from ARDs.

Here's the general outline for creating this table directly from ARDs.

1. Create an ARD of survival quantiles using `cardx::ard_survival_survfit()`.
2. Construct an ARD of the minimum and maximum survival times using `cards::ard_summary()`.
3. Combine the ARDs and build summary table with `gtsummary::tbl_ard_summary()`.

```
# get the survival quantiles with 95% CI
ard_surv_quantiles <-
  cardx::ard_survival_survfit(
    x = cards::ADTTE,
   y = survival::Surv(time = AVAL, event = 1 - CNSR, type = 'right', origin = 0),
    variables = "TRTA",
    probs = c(0.25, 0.50, 0.75)
  ) |>
  # modify the shape of the ARD to look like a
  # 'continuous' result to feed into `tbl_ard_summary()`
  dplyr::mutate(
    stat_name = paste0(.data$stat_name, 100 * unlist(.data$variable_level)),
    variable_level = list(NULL)
  )

# get the min/max followup time
ard_surv_min_max <-
  cards::ard_summary(
    data = cards::ADTTE,
    variables = AVAL,
    by = "TRTA",
```

```
    statistic = everything() ~ cards::continuous_summary_fns(c("min", "max"))
  )

# stack the ARDs and pass them to `tbl_ard_summary()`
cards::bind_ard(
  ard_surv_quantiles,
  ard_surv_min_max
) |>
  tbl_ard_summary(
    by = "TRTA",
    type = list(prob = "continuous2", AVAL = "continuous"),
    statistic = list(
    prob = c("{estimate50}", "({conf.low50}, {conf.high50})", "{estimate25}, {estimate75}"),
      AVAL = "{min} to {max}"
    ),
    label = list(
      prob = "Time to event",
      AVAL = "Range"
    )
  ) |>
  # directly modify the labels in the table to match spec
  modify_table_body(
    ~ .x |>
    dplyr::mutate(
      label = dplyr::case_when(
        .data$label == "Survival Probability" ~ "Median",
        .data$label == "(CI Lower Bound, CI Upper Bound)" ~ "95% CI",
     .data$label == "Survival Probability, Survival Probability" ~ "25% and 75%-ile",
        .default = .data$label
      )
    )
  ) |>
  # update indentation to match spec
  modify_indent(columns = "label", rows = label == "95% CI", indent = 8L) |>
  modify_indent(columns = "label", rows = .data$label == "Range", indent = 4L) |>
  # remove default footnotes
  remove_footnote_header(columns = all_stat_cols())
```

Examples

```
# Example 1 ----------------------------------
tbl_survfit_quantiles(
  data = cards::ADTTE,
  by = "TRTA",
  estimate_fun = label_roche_number(digits = 1, na = "NE")
) |>
  add_overall(last = TRUE, col_label = "**All Participants**  \nN = {n}")

# Example 2: unstratified analysis -----------
```

```
tbl_survfit_quantiles(data = cards::ADTTE)
```

`tbl_survfit_times` *Survival Times*

Description

Create a gtsummary table with Kaplan-Meier estimated survival estimates and specified times.

Usage

```
tbl_survfit_times(
  data,
  times,
  y = "survival::Surv(time = AVAL, event = 1 - CNSR, type = 'right', origin = 0)",
  by = NULL,
  label = "Time {time}",
  statistic = c("{n.risk}", "{estimate}", "({conf.low}, {conf.high})"),
  estimate_fun = label_roche_number(digits = 1, scale = 100),
  method.args = list(conf.int = 0.95)
)

## S3 method for class 'tbl_survfit_times'
add_difference_row(
  x,
  reference,
  statistic = c("{estimate}", "({conf.low}, {conf.high})", "{p.value}"),
  conf.level = 0.95,
  pvalue_fun = label_roche_pvalue(),
  estimate_fun = label_roche_number(digits = 2, scale = 100),
  ...
)

## S3 method for class 'tbl_survfit_times'
add_overall(
  x,
  last = FALSE,
  col_label = "All Participants  \nN = {style_roche_number(N)}",
  ...
)
```

Arguments

`data`	(`data.frame`) A data frame
`times`	(`numeric`) a vector of times for which to return survival probabilities.

y (string or expression)
A string or expression with the survival outcome, e.g. survival::Surv(time, status). The default value is survival::Surv(time = AVAL, event = 1 - CNSR, type = "right", origin = 0).

by (tidy-select)
A single column from data. Summary statistics will be stratified by this variable. Default is NULL, which returns results for the unstratified model.

label (string)
Label to appear in the header row. Default is "Time {time}", where the glue syntax injects the time estimate into the label.

statistic (character)
Character vector of the statistics to report. May use any of the following statistics: c(n.risk, estimate, std.error, conf.low, conf.high), Default is c("{n.risk}", "{estimate}", "({conf.low}, {conf.high})")

Statistics available to include when using add_difference_row() are: "estimate", "std.error", "statistic", "conf.low", "conf.high", "p.value".

estimate_fun (function)
Function used to style/round the c(estimate, conf.low, conf.high) statistics.

method.args (named list)
Named list of arguments that will be passed to survival::survfit().

Note that this list may contain non-standard evaluation components, and must be handled similarly to tidyselect inputs by using rlang's embrace operator {{ . }} or !!enquo() when programming with this function.

x (tbl_survfit_times)
A stratified 'tbl_survfit_times' object

reference (string)
Value of the tbl_survfit_times(by) variable value that is the reference for each of the difference calculations. For factors, use the character level. The reference column will appear as the leftmost column in the table.

conf.level (numeric)
a scalar in the interval (0, 1) indicating the confidence level. Default is 0.95

pvalue_fun (function)
Function to round and format the p.value statistic. Default is label_roche_pvalue(). The function must have a numeric vector input, and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = label_style_pvalue(digits = 3)).

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

last (scalar logical)
Logical indicator to display overall column last in table. Default is FALSE, which will display overall column first.

col_label (string)
String indicating the column label. Default is "**Overall** \nN = {style_number(N)}"

Details

When the `statistic` argument is modified, the statistic labels will likely also need to be updated. To change the label, call the `modify_table_body()` function to directly update the underlying `x$table_body` data frame.

Value

a gtsummary table

Methods (by generic)

- `add_difference_row(tbl_survfit_times)`: Adds survival differences between groups as additional rows to tables created by `tbl_survfit_times()`.

  Difference statistics are calculated using `cardx::ard_survival_survfit_diff()` for all `tbl_survfit_times(times)` variable values, using `survfit` formula:

  ```
  survival::survfit(y ~ by, data = data)
  ```

  where `y`, `by` and `data` are the inputs of the same names to the `tbl_survfit_times()` object `x`.

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

Examples

```
# Example 1 ----------------------------------
tbl_survfit_times(
  data = cards::ADTTE,
  by = "TRTA",
  times = c(30, 60),
  label = "Day {time}"
) |>
  add_overall()
# Example 2 - Survival Differences -----------
tbl_survfit_times(
  data = cards::ADTTE,
  by = "TRTA",
  times = c(30, 60),
  label = "Day {time}"
) |>
  add_difference_row(reference = "Placebo")
```

`theme_gtsummary_roche` *Roche Theme*

Description

A gtsummary theme for Roche tables

- flextable- and gt-printed tables are styled with reduced padding and font size.
- Uses `label_roche_pvalue()` as the default formatting function for all p-values.
- Uses `label_roche_percent()` as the default formatting function for all percent values.
- Font size defaults are 8 points for all the table by the footers that are 7 points.
- Border defaults to `flextable::fp_border_default(width = 0.5)`.
- The `add_overall(col_label)` default value has been updated.
- The results from `gtsummary::tbl_hierarchical()` and `gtsummary::tbl_hierarchical_count()` are now post-processed with `gtsummary::remove_footnote_header()`, `crane::modify_zero_recode()`, and `crane::modify_header_rm_md()`.

Usage

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

Arguments

`font_size`	(scalar `numeric`) Numeric font size for compact theme. Default is 13 for gt tables, and 8 for all other output types
`print_engine`	String indicating the print method. Must be one of `"gt"`, `"kable"`, `"kable_extra"`, `"flextable"`, `"tibble"`
`set_theme`	(scalar `logical`) Logical indicating whether to set the theme. Default is `TRUE`. When `FALSE` the named list of theme elements is returned invisibly

Value

theme list

Examples

```
theme_gtsummary_roche()

tbl_roche_summary(
  trial,
  by = trt,
  include = c(age, grade),
  nonmissing = "always"
)

reset_gtsummary_theme()
```

Index