Package: sumExtras (via r-universe)

June 10, 2026

Title Extra Functions for 'gtsummary' Table Styling

Version 1.0.0

Description Provides additional convenience functions for 'gtsummary' (Sjoberg et al. (2021) <doi:10.32614/RJ-2021-053>) & 'gt' tables, including automatic variable labeling from dictionaries, standardized missing value display, and consistent formatting helpers for streamlined table styling workflows.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Suggests broom (>= 1.0.5), broom.helpers (>= 1.20.0), knitr, labelled, survey, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Imports dplyr, gt (>= 0.9.0), gtsummary (>= 1.7.0), rlang

URL https://github.com/kyleGrealis/sumExtras, https://www.kyleGrealis.com/sumExtras/

BugReports https://github.com/kyleGrealis/sumExtras/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev libnode-dev

Repository https://kylegrealis.r-universe.dev

Date/Publication 2026-06-10 01:43:55 UTC

RemoteUrl https://github.com/kylegrealis/sumextras

RemoteRef HEAD

RemoteSha 11f8afa6fc88ab08dd69e5eb8b41c91e20ab59d5

Contents

`add_auto_labels` *Add automatic labels from dictionary to a gtsummary table*

Description

Applies variable labels from a dictionary or label attributes to `tbl_summary`, `tbl_svysummary`, or `tbl_regression` objects. Preserves manual label overrides set in the original table call. The dictionary can be passed explicitly or will be searched for in the calling environment. If no dictionary is found, the function reads label attributes from the underlying data.

Usage

```
add_auto_labels(tbl, dictionary)
```

Arguments

`tbl`	A gtsummary table object created by `tbl_summary()`, `tbl_svysummary()`, or `tbl_regression()`.
`dictionary`	A data frame or tibble with columns named `variable` and `description` (column name matching is case-insensitive). If not provided (missing), the function will search for a `dictionary` object in the environment. If no dictionary is found, the function will attempt to read label attributes from the data. Set to `NULL` explicitly to skip dictionary search and only use attributes.

Details

Label Priority Hierarchy:

The function applies labels according to this priority (highest to lowest):

1. **Manual labels** – Labels set via `label = list(...)` in `tbl_summary()` etc. are always preserved
2. **Attribute labels** – Labels from `attr(data$var, "label")`
3. **Dictionary labels** – Labels from the dictionary data frame
4. **Default** – If no label source is available, uses variable name

Set `options(sumExtras.prefer_dictionary = TRUE)` to swap priorities 2 and 3 so that dictionary labels take precedence over attribute labels. See `vignette("options")` for details.

Dictionary Format:

The dictionary must be a data frame with columns (column names are case-insensitive):

- `variable`: Character column with exact variable names from datasets
- `description`: Character column with human-readable labels

Label Attributes:

The function reads label attributes from data using `attr(data$var, "label")`, following the same convention used by haven, Hmisc, and ggplot2 4.0+. If your data already has labels (from imported files, other packages, or manual assignment), this function picks them up automatically.

Implementation Note:

This function relies on internal gtsummary structures (`tbl$call_list`, `tbl$inputs`, `tbl$table_body`) to detect manually set labels. Major updates to gtsummary may require corresponding updates to sumExtras. Requires gtsummary >= 1.7.0.

Value

A gtsummary table object with labels applied. Manual labels set via `label = list(...)` in the original table call are always preserved.

See Also

- `gtsummary::modify_table_body()` for advanced table customization

Examples

```
# Create a dictionary
my_dict <- tibble::tribble(
  ~variable, ~description,
  "age", "Age at Enrollment",
  "trt", "Treatment Group",
  "grade", "Tumor Grade"
)

# Strip built-in labels so dictionary labels are visible
trial_data <- gtsummary::trial
for (col in names(trial_data)) attr(trial_data[[col]], "label") <- NULL

# Pass dictionary explicitly
trial_data |>
  gtsummary::tbl_summary(by = trt, include = c(age, grade)) |>
  add_auto_labels(dictionary = my_dict)

# Automatic dictionary search (dictionary in environment)
dictionary <- my_dict
trial_data |>
  gtsummary::tbl_summary(by = trt, include = c(age, grade)) |>
  add_auto_labels() # Finds dictionary automatically
```

```
# Working with pre-labeled data (no dictionary needed)
labeled_data <- gtsummary::trial
attr(labeled_data$age, "label") <- "Patient Age (years)"
attr(labeled_data$marker, "label") <- "Marker Level (ng/mL)"

labeled_data |>
  gtsummary::tbl_summary(include = c(age, marker)) |>
  add_auto_labels() # Reads from label attributes

# Manual overrides always win
trial_data |>
  gtsummary::tbl_summary(
    by = trt,
    include = c(age, grade),
    label = list(age ~ "Custom Age Label") # Manual override
  ) |>
  add_auto_labels(dictionary = my_dict) # grade: dict, age: manual
```

`add_group_colors` *Add background colors to group headers with automatic gt conversion*

Description

Convenience function that adds background colors to variable group headers and converts the table to gt. This is a terminal operation that combines `get_group_rows()`, `gtsummary::as_gt()`, and `gt::tab_style()` into a single pipeable function.

For text formatting (bold/italic), use `add_group_styling()` before calling this function.

Usage

```
add_group_colors(tbl, color = "#E8E8E8")
```

Arguments

`tbl`	A gtsummary table object with variable group headers created by `gtsummary::add_variable_group_he`
`color`	Background color(s) for group headers. Default `"#E8E8E8"` (light gray). Accepts a single color (applied to all groups) or a vector of colors (one per group). Can be any valid CSS color (hex code, color name, rgb(), etc.).

Details

This function:

1. Identifies group header rows with `get_group_rows()`
2. Converts the table to gt with `gtsummary::as_gt()`

3. Applies background color using `gt::tab_style()`

Since this function converts to gt, it should be used as the final styling step in your pipeline. Apply all gtsummary functions (like `modify_caption()`, `modify_footnote()`, etc.) and text formatting with `add_group_styling()` before calling `add_group_colors()`.

Value

A gt table object with colored group headers. **Note:** This is a terminal operation that converts to gt. You cannot pipe to additional gtsummary functions after calling this function.

See Also

- `add_group_styling()` for text formatting only (stays gtsummary)
- `get_group_rows()` for identifying group header rows
- `gtsummary::add_variable_group_header()` for creating variable groups
- `gt::tab_style()` for additional gt-specific styling

Examples

```
# Basic usage - text formatting then color
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras() |>
  gtsummary::add_variable_group_header(
    header = "Patient Characteristics",
    variables = age:stage
  ) |>
  add_group_styling() |>
  add_group_colors()

# Custom color - light blue
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras() |>
  gtsummary::add_variable_group_header(
    header = "Baseline Characteristics",
    variables = age:marker
  ) |>
  add_group_styling() |>
  add_group_colors(color = "#E3F2FD")

# Bold only formatting with custom color
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras() |>
  gtsummary::add_variable_group_header(
    header = "Clinical Measures",
    variables = marker:stage
  ) |>
  add_group_styling(format = "bold") |>
```

```
  add_group_colors(color = "#FFF9E6")

# Multiple group headers with same color
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras() |>
  gtsummary::add_variable_group_header(
    header = "Demographics",
    variables = age
  ) |>
  gtsummary::add_variable_group_header(
    header = "Disease Measures",
    variables = marker:response
  ) |>
  add_group_styling() |>
  add_group_colors(color = "#E8E8E8")

# Different colors per group
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras() |>
  gtsummary::add_variable_group_header(
    header = "Demographics",
    variables = age
  ) |>
  gtsummary::add_variable_group_header(
    header = "Disease Measures",
    variables = marker:response
  ) |>
  add_group_styling() |>
  add_group_colors(color = c("#E3F2FD", "#FFF9E6"))
```

`add_group_styling` *Apply styling to variable group headers in gtsummary tables*

Description

Adds customizable formatting to variable group headers in gtsummary tables. Variable groups are created using `gtsummary::add_variable_group_header()` to organize variables into sections. This function makes group headers stand out from individual variable labels.

Usage

```
add_group_styling(tbl, format = c("bold", "italic"), indent_labels = 0L)
```

Arguments

tbl	A gtsummary table object (e.g., from tbl_summary(), tbl_regression())
format	Character vector specifying text formatting. Options include "bold", "italic", or both. Default is c("bold", "italic").
indent_labels	Integer specifying indentation level (in spaces) for variable labels under group headers. Default is 0L (left-aligned). Set to 4L to preserve gtsummary's default group indentation, or use any non-negative integer for custom spacing.

Details

The function targets rows where row_type == 'variable_group' and applies the specified text formatting to the label column.

By default, variable labels are left-aligned (indent_labels = 0L) to distinguish them from categorical levels and statistics. Use indent_labels = 4L to preserve the default gtsummary behavior where grouped variables are indented under their group headers.

Value

A gtsummary table object with specified formatting applied to variable group headers

See Also

- gtsummary::modify_table_styling() for general table styling options
- gtsummary::add_variable_group_header() for creating variable group headers

Examples

```
# Default formatting (bold and italic)
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt, include = c(age, marker, grade)) |>
  gtsummary::add_variable_group_header(
    header = "Patient Characteristics",
    variables = age:grade
  ) |>
  add_group_styling()

# Bold only
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt, include = c(age, marker)) |>
  gtsummary::add_variable_group_header(
    header = "Demographics",
    variables = age:marker
  ) |>
  add_group_styling(format = "bold")

# Multiple group headers
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  gtsummary::add_variable_group_header(
```

```
    header = "Demographics",
    variables = age
  ) |>
  gtsummary::add_variable_group_header(
    header = "Clinical Measures",
    variables = marker:response
  ) |>
  add_group_styling()

# Custom indentation for grouped variables
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt, include = c(age, marker)) |>
  gtsummary::add_variable_group_header(
    header = "Patient Measures",
    variables = age:marker
  ) |>
  add_group_styling(indent_labels = 4L) # gtsummary indentation restored
```

`clean_table` *Standardize missing value display across all gtsummary table types*

Description

Replaces various missing value representations with a consistent symbol (default `"---"`) so it is easier to tell actual data from missing or undefined values.

Works with all gtsummary table types, including stacked tables (`tbl_strata`) and survey-weighted summaries (`tbl_svysummary`). Handles tables with or without the standard `var_type` column.

Usage

```
clean_table(tbl, symbol = "---")
```

Arguments

`tbl`	A gtsummary table object (e.g., from `tbl_summary()`, `tbl_svysummary()`, `tbl_regression()`, or `tbl_strata()`)
`symbol`	Character string to replace missing values with. Default is `"---"` (em-dash style). Common alternatives: `"\u2014"` (em-dash), `"\u2013"` (en-dash), `"--"`, or `"N/A"`.

Details

The function uses `gtsummary::modify_table_body()` to transform character columns and replace missing, undefined, and zero-valued patterns with a consistent symbol. Matched patterns include:

- Literal `NA` and `Inf` / `-Inf` values

- Count/percent pairs: `"0 (0%)"`, `"0 (NA%)"`, `"0 (NA)"`, `"NA (0)"`, `"NA (NA)"`
- Decimal variants: `"0.00 (0.00)"`, `"0.00% (0.00)"`, `"0% (0.000)"`
- Paired values: `"NA, NA"`
- Confidence intervals: `"NA (NA, NA)"`, `"0% (0.000) (0%, 0%)"`, `"0.00 (0.00) (0.00, 0.00)"`, and similar zero-CI patterns

Replacing these patterns with a single symbol keeps the table easier to read.

Note: The function checks for the presence of `var_type` column before applying `modify_missing_symbol()`. This allows it to work with `tbl_strata` objects which use `var_type_1`, `var_type_2`, etc. instead of `var_type`.

Value

A gtsummary table object with standardized missing value display

See Also

- `gtsummary::modify_table_body()` for general table body modifications
- `extras()` which includes `clean_table()` in its styling pipeline

Examples

```
# Basic usage - clean missing values in summary table
demo_trial <- gtsummary::trial |>
  dplyr::mutate(
    age = dplyr::if_else(trt == "Drug B", 0, age),
    marker = dplyr::if_else(trt == "Drug A", NA, marker)
  ) |>
  dplyr::select(trt, age, marker)

demo_trial |>
  gtsummary::tbl_summary(by = trt) |>
  clean_table()

# Used inside extras() automatically
demo_trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras()

# Custom missing symbol
demo_trial |>
  gtsummary::tbl_summary(by = trt) |>
  clean_table(symbol = "???")
```

`extras` *Add standard styling and formatting to gtsummary tables*

Description

Applies a consistent set of formatting options to gtsummary tables including overall column, bold labels, clean headers, and optional p-values. Wraps the common workflow of adding multiple formatting functions into one call. Always succeeds by applying what works and warning about the rest.

Usage

```
extras(
  tbl,
  pval = TRUE,
  overall = TRUE,
  last = FALSE,
  header = "",
  symbol = "---",
  .args = NULL,
  .add_p_args = NULL
)
```

Arguments

`tbl`	A gtsummary table object (e.g., from `tbl_summary()`, `tbl_regression()`)
`pval`	Logical indicating whether to add p-values. Default is `TRUE`. When `TRUE`, uses gtsummary's default statistical tests (Kruskal-Wallis for continuous variables with 3+ groups, chi-square for categorical variables).
`overall`	Logical indicating whether to add overall column
`last`	Logical indicating if Overall column should be last. Aligns with default from `gtsummary::add_overall()`.
`header`	Character string for the label column header. Default is `""` (blank). Use `"Characteristic"` or any custom text.
`symbol`	Character string for missing value replacement in `clean_table()`. Default is `"---"`. Passed directly to `clean_table(symbol = ...)`.
`.args`	Optional list of arguments to use instead of individual parameters. When provided, overrides `pval`, `overall`, `last`, `header`, and `symbol` arguments.
`.add_p_args`	Optional named list of arguments to pass to `gtsummary::add_p()`. Allows customization of statistical tests and p-value formatting. User-provided arguments override the default arguments (`pvalue_fun` and `test.args`). See `gtsummary::add_p()` documentation for available arguments.

Details

The function applies the following modifications (in order):

1. Bolds variable labels for emphasis (all table types)
2. Removes the "Characteristic" header label (all table types)
3. Adds an "Overall" column (only stratified summary tables)
4. Optionally adds p-values with bold significance (only stratified summary tables)
5. Applies automatic labels if options are set (see Options section)
6. Applies `clean_table()` styling (all table types)

The function automatically detects whether the input table is stratified (has a by argument) and what type of table it is (tbl_summary, tbl_regression, tbl_strata, etc.).

For tables that don't support overall columns or p-values (non-stratified tables, regression tables, or stacked tables), the function warns and applies only basic formatting (bold_labels and modify_header).

For merged tables (`tbl_merge`), call `extras()` on each sub-table before merging. All formatting carries through.

If any individual step fails (e.g., due to unexpected table structure), the function warns and continues without that feature.

Value

A gtsummary table object with standard formatting applied

Options

Set `options(sumExtras.auto_labels = TRUE)` for automatic labeling. See `vignette("options")` for details.

Pipeline Ordering

Call `extras()` before `add_variable_group_header()` and `add_group_colors()` last. See `vignette("sumExtras-intro"`

Table Type Support

Full features (overall, p-values, bold p-values) require a stratified `tbl_summary` or `tbl_svysummary`. Regression tables get bold labels, bold model p-values, header cleaning, and `clean_table()`. Stacked (`tbl_strata`) and merged (`tbl_merge`) tables get bold labels, header cleaning, and `clean_table()`. Warnings only fire when the user explicitly requests unsupported features (e.g., `overall = TRUE` on a non-stratified table).

See Also

- `gtsummary::add_overall()` for adding overall columns
- `gtsummary::add_p()` for adding p-values
- `clean_table()` for additional table styling

Examples

```
# With p-values (default)
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras()

# Using .args list
extra_args <- list(pval = TRUE, overall = TRUE, last = FALSE)
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras(.args = extra_args)

# Without p-values
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras(pval = FALSE)

# Custom header text
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras(header = "Variable")

# Customize add_p() behavior
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt) |>
  extras(.add_p_args = list(
    test = list(age ~ "t.test", marker ~ "t.test"),
    pvalue_fun = ~ gtsummary::style_pvalue(.x, digits = 2)
  ))
```

---

get_group_rows *Get row numbers of variable group headers for gt styling*

Description

Extracts the row indices of variable group headers from a gtsummary table. This is useful for applying background colors or other gt-specific styling after converting a gtsummary table to gt with `as_gt()`.

Usage

```
get_group_rows(tbl)
```

Arguments

`tbl`	A gtsummary table object with variable group headers created by `gtsummary::add_variable_group_he`

Details

Variable group headers are identified by `row_type == 'variable_group'` in the table body. The returned row numbers can be used with `gt::tab_style()` to apply styling like background colors after converting to a gt table.

This function should be called BEFORE converting the table with `as_gt()`, as the row type information is only available in gtsummary table objects.

Value

An integer vector of row numbers where variable_group headers are located

See Also

- `add_group_styling()` for applying text formatting to group headers
- `gtsummary::add_variable_group_header()` for creating variable groups
- `gt::tab_style()` for applying gt-specific styling

Examples

```
# Create table with variable groups
my_tbl <- gtsummary::trial |>
  gtsummary::tbl_summary(by = trt, include = c(age, marker, grade, stage)) |>
  gtsummary::add_variable_group_header(
    header = "Demographics",
    variables = age
  ) |>
  gtsummary::add_variable_group_header(
    header = "Clinical",
    variables = marker:stage
  ) |>
  add_group_styling()

# Get group row numbers before conversion
group_rows <- get_group_rows(my_tbl)

# Convert to gt and apply gray background
my_tbl |>
  gtsummary::as_gt() |>
  gt::tab_style(
    style = gt::cell_fill(color = "#E8E8E8"),
    locations = gt::cells_body(rows = group_rows)
  )
```

`theme_gt_compact` *Apply compact JAMA-style theme to gt tables*

Description

Applies a compact table theme to gt tables that matches the 'jama' theme from gtsummary, so gtsummary and plain gt tables look the same in one document. Reduces padding, adjusts font sizes, and applies JAMA journal styling.

Usage

```
theme_gt_compact(tbl)
```

Arguments

`tbl`	A gt table object created with `gt::gt()`

Details

This function replicates the visual appearance of `gtsummary::theme_gtsummary_compact("jama")` for use with regular gt tables. Key styling includes:

- Reduced font size (13px) for compact appearance
- Minimal padding (1px) on all row types
- Bold column headers and table titles
- Hidden top and bottom table borders
- Consistent spacing that matches JAMA journal standards

Value

A gt table object with compact JAMA-style formatting applied

See Also

- `sumExtras::use_jama_theme()` for complimentary table styling
- `gt::tab_options()` for additional gt table styling options

Examples

```
# Basic usage with a data frame
mtcars |>
  head() |>
  gt::gt() |>
  theme_gt_compact()

# Combine with other gt functions
mtcars |>
```

```
  head() |>
  gt::gt() |>
  gt::tab_header(title = "Vehicle Data") |>
  theme_gt_compact()

# Use alongside gtsummary tables with sumExtras for consistency
# Set JAMA theme first
use_jama_theme()

# Then both tables will have matching appearance
summary_table <- gtsummary::trial |>
  gtsummary::tbl_summary()

data_table <- gtsummary::trial |>
  head() |>
  gt::gt() |>
  theme_gt_compact()
```

`use_jama_theme` *Apply JAMA Compact Theme to gtsummary Tables*

Description

Sets the global gtsummary theme to the JAMA (Journal of the American Medical Association) compact style. Reduces padding and applies JAMA journal styling. The theme stays active for the entire R session or until changed with another theme.

Usage

```
use_jama_theme(quiet = TRUE)
```

Arguments

`quiet`	Logical. If `FALSE`, prints a message confirming theme application. Default is `TRUE` (silent).

Details

The JAMA compact theme applies formatting standards from the Journal of the American Medical Association: 13px font, 1px cell padding, bold column headers, and clean borders.

The function checks for the gtsummary package and will stop with an informative error if it is not installed. The theme is applied globally and will affect all gtsummary tables created after calling this function, including `tbl_summary()`, `tbl_regression()`, `tbl_cross()`, `tbl_strata()`, and related functions.

For visual consistency with regular gt tables, use `theme_gt_compact()` which replicates the same styling for non-gtsummary tables.

Value

Invisibly returns the theme list object from `gtsummary::theme_gtsummary_compact("jama")`. The theme is applied globally via `gtsummary::set_gtsummary_theme()`, affecting all subsequent gtsummary tables created in the session.

See Also

- `theme_gt_compact` for JAMA-style gt tables
- `extras` for standard sumExtras table formatting
- `gtsummary::theme_gtsummary_compact()` for other compact theme options
- `gtsummary::set_gtsummary_theme()` for setting custom themes
- `gtsummary::reset_gtsummary_theme()` for resetting to default theme

Examples

```
# Apply theme at the start of your analysis
use_jama_theme()

# All subsequent gtsummary tables will use JAMA formatting
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt)

# Works with all gtsummary table types
lm(age ~ trt + grade, data = gtsummary::trial) |>
  gtsummary::tbl_regression()

# Combine with sumExtras styling functions
use_jama_theme()
gtsummary::trial |>
  gtsummary::tbl_summary(by = trt, include = c(age, marker, stage)) |>
  extras() |>
  add_group_styling()

# Reset to default theme if needed
gtsummary::reset_gtsummary_theme()
```

Index