Package 'tsg'

May 8, 2026

Type Package

Title Generate Publication-Ready Statistical Tables

Version 0.1.4

Description A collection of functions for generating frequency tables and cross-tabulations of categorical variables. The resulting tables can be exported to various formats (Excel, PDF, HTML, etc.) with extensive formatting and layout customization options.

Author Bhas Abdulsamad [aut, cre, cph] (ORCID: <https://orcid.org/0009-0002-5891-8124>)

Maintainer Bhas Abdulsamad <aeabdulsamad@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports haven, openxlsx, fs, forcats, purrr, tidyr, glue, jsonlite (>= 2.0.0), yaml (>= 2.3.10), dplyr (>= 1.1.0), stringr (>= 1.4.0), rlang (>= 1.0.2), gt, lifecycle, stats, utils, officer, flextable, qpdf

Suggests testthat (>= 3.0.0), cli, devtools, knitr, rmarkdown, tibble, webshot2, withr

Config/testthat/edition 3

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

VignetteBuilder knitr

URL https://yng-me.github.io/tsg/, https://github.com/yng-me/tsg

BugReports https://github.com/yng-me/tsg/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2026-05-05 11:10:02 UTC

Contents

`add_column_total` *Add a column total*

Description

Add a column total

Usage

```
add_column_total(data, name = "total", label_total = "Total", ...)
```

Arguments

`data`	A data frame, tibble, or `tsg` object to which a column row will be added.
`name`	Column name for total. Default value is `"total"`.
`label_total`	Label for the total column. Default is "Total".
`...`	Additional named arguments to be added as columns alongside the total column.

Value

The input data frame with an additional column representing the total of each row.

Examples

```
# Example data frame
df <- data.frame(
 category = c("A", "B", "C"),
 value1 = c(10, 20, 30),
 value2 = c(5, 15, 25)
 )
add_column_total(df)
```

`add_facade` *Add a facade to a tsg table*

Description

This function adds a facade to a `tsg` table object. A facade is a set of styling options that can be applied to the table to customize its appearance. For Excel output, see `openxlsx::createStyle()` for all valid values.

Usage

```
add_facade(
  data,
  table.offsetRow = 0,
  table.offsetCol = 0,
  table.gridLines = NULL,
  table.tabColour = NULL,
  table.fontName = NULL,
  table.fontSize = NULL,
  table.fontColour = NULL,
  table.bgFill = NULL,
  table.fgFill = NULL,
  table.halign = NULL,
  table.valign = NULL,
  table.wrapText = FALSE,
  table.indent = NULL,
  table.locked = NULL,
  table.hidden = NULL,
  table.decimalPrecision = NULL,
  table.decimalCols = NULL,
  table.lastRowBold = NULL,
  table.width = NULL,
  table.widthOffset = NULL,
```

```
  title.fontName = NULL,
  title.fontSize = NULL,
  title.fontColour = NULL,
  title.border = NULL,
  title.borderColour = NULL,
  title.borderStyle = NULL,
  title.bgFill = NULL,
  title.fgFill = NULL,
  title.halign = NULL,
  title.valign = NULL,
  title.textDecoration = NULL,
  title.wrapText = NULL,
  title.indent = NULL,
  title.height = NULL,
  subtitle.fontName = NULL,
  subtitle.fontSize = NULL,
  subtitle.fontColour = NULL,
  subtitle.border = NULL,
  subtitle.borderColour = NULL,
  subtitle.borderStyle = NULL,
  subtitle.bgFill = NULL,
  subtitle.fgFill = NULL,
  subtitle.halign = NULL,
  subtitle.valign = NULL,
  subtitle.textDecoration = NULL,
  subtitle.wrapText = NULL,
  subtitle.indent = NULL,
  subtitle.height = NULL,
  header.fontName = NULL,
  header.fontSize = NULL,
  header.fontColour = NULL,
  header.border = NULL,
  header.borderColour = NULL,
  header.borderStyle = NULL,
  header.bgFill = NULL,
  header.fgFill = NULL,
  header.halign = NULL,
  header.valign = NULL,
  header.textDecoration = NULL,
  header.wrapText = NULL,
  header.indent = NULL,
  header.height = NULL,
  spanner.fontName = NULL,
  spanner.fontSize = NULL,
  spanner.fontColour = NULL,
  spanner.border = NULL,
  spanner.borderColour = NULL,
  spanner.borderStyle = NULL,
```

```
  spanner.bgFill = NULL,
  spanner.fgFill = NULL,
  spanner.halign = NULL,
  spanner.valign = NULL,
  spanner.textDecoration = NULL,
  spanner.wrapText = NULL,
  spanner.indent = NULL,
  spanner.height = NULL,
  body.fontName = NULL,
  body.fontSize = NULL,
  body.fontColour = NULL,
  body.numFmt = NULL,
  body.border = NULL,
  body.borderColour = NULL,
  body.borderStyle = NULL,
  body.bgFill = NULL,
  body.fgFill = NULL,
  body.halign = NULL,
  body.valign = NULL,
  body.textDecoration = NULL,
  body.wrapText = NULL,
  body.indent = NULL,
  body.height = NULL,
  col_first.fontName = NULL,
  col_first.fontSize = NULL,
  col_first.fontColour = NULL,
  col_first.numFmt = NULL,
  col_first.border = NULL,
  col_first.borderColour = NULL,
  col_first.borderStyle = NULL,
  col_first.bgFill = NULL,
  col_first.fgFill = NULL,
  col_first.halign = NULL,
  col_first.valign = NULL,
  col_first.textDecoration = NULL,
  col_first.wrapText = NULL,
  col_first.indent = NULL,
  col_first.width = NULL,
  col_last.fontName = NULL,
  col_last.fontSize = NULL,
  col_last.fontColour = NULL,
  col_last.numFmt = NULL,
  col_last.border = NULL,
  col_last.borderColour = NULL,
  col_last.borderStyle = NULL,
  col_last.bgFill = NULL,
  col_last.fgFill = NULL,
  col_last.halign = NULL,
```

```
  col_last.valign = NULL,
  col_last.textDecoration = NULL,
  col_last.wrapText = NULL,
  col_last.indent = NULL,
  col_last.width = NULL,
  row_group.fontName = NULL,
  row_group.fontSize = NULL,
  row_group.fontColour = NULL,
  row_group.border = NULL,
  row_group.borderColour = NULL,
  row_group.borderStyle = NULL,
  row_group.bgFill = NULL,
  row_group.fgFill = NULL,
  row_group.halign = NULL,
  row_group.valign = NULL,
  row_group.textDecoration = NULL,
  row_group.wrapText = NULL,
  row_group.indent = NULL,
  row_group.width = NULL,
  row_group.height = NULL,
  source_note.fontName = NULL,
  source_note.fontSize = NULL,
  source_note.fontColour = NULL,
  source_note.border = NULL,
  source_note.borderColour = NULL,
  source_note.borderStyle = NULL,
  source_note.bgFill = NULL,
  source_note.fgFill = NULL,
  source_note.halign = NULL,
  source_note.valign = NULL,
  source_note.textDecoration = NULL,
  source_note.wrapText = NULL,
  source_note.indent = NULL,
  source_note.height = NULL,
  footnotes.fontName = NULL,
  footnotes.fontSize = NULL,
  footnotes.fontColour = NULL,
  footnotes.border = NULL,
  footnotes.borderColour = NULL,
  footnotes.borderStyle = NULL,
  footnotes.bgFill = NULL,
  footnotes.fgFill = NULL,
  footnotes.halign = NULL,
  footnotes.valign = NULL,
  footnotes.textDecoration = NULL,
  footnotes.wrapText = NULL,
  footnotes.indent = NULL,
  footnotes.height = NULL,
```

```
  border_header.border = NULL,
  border_header.borderColour = NULL,
  border_header.borderStyle = NULL,
  border_outer.borderColour = NULL,
  border_bottom.height = NULL
)
```

Arguments

data	A tsg table object to which the facade will be added. This is typically a data frame or tibble that has been processed using tsg functions.
table.offsetRow	Row offset of the table
table.offsetCol	Column offset of the table
table.gridLines	Boolean indicating whether to show grid lines in the table
table.tabColour	Color of the table tab (Excel worksheet) in the output file. Can be a hexadecimal color code (e.g., "#FF0000" for red) or a named color (e.g., "red").
table.fontName, title.fontName, subtitle.fontName, header.fontName, spanner.fontName, body.fontName, row_group.fontName, col_first.fontName, col_last.fontName, source_note.fontName, footnotes.fontName	Font name or font family for the table, title, subtitle, header, spanner, body, row group header, source note, and footnotes respectively.
table.fontSize, title.fontSize, subtitle.fontSize, header.fontSize, spanner.fontSize, body.fontSize, col_first.fontSize, col_last.fontSize, row_group.fontSize, source_note.fontSize, footnotes.fontSize	Font size for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively.
table.fontColour, title.fontColour, subtitle.fontColour, header.fontColour, spanner.fontColour, body.fontColour, col_first.fontColour, col_last.fontColour, row_group.fontColour, source_note.fontColour, footnotes.fontColour	Font color for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a hexadecimal color code (e.g., "#FF0000" for red) or a named color (e.g., "red").
table.bgFill, title.bgFill, subtitle.bgFill, header.bgFill, spanner.bgFill, body.bgFill, col_first.bgFill, col_last.bgFill, row_group.bgFill, source_note.bgFill, footnotes.bgFill	Background fill color for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a hexadecimal color code (e.g., "#FF0000" for red) or a named color (e.g., "red").

`table.fgFill`, `title.fgFill`, `subtitle.fgFill`, `header.fgFill`, `spanner.fgFill`, `body.fgFill`, `col_first.fgFill`, `col_last.fgFill`, `row_group.fgFill`, `source_note.fgFill`, `footnotes.fgFill`
: Foreground fill color for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a hexadecimal color code (e.g., "#FF0000" for red) or a named color (e.g., "red").

`table.halign`, `title.halign`, `subtitle.halign`, `header.halign`, `spanner.halign`, `body.halign`, `col_first.halign`, `col_last.halign`, `row_group.halign`, `source_note.halign`, `footnotes.halign`
: Horizontal alignment for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be "left", "center", or "right".

`table.valign`, `title.valign`, `subtitle.valign`, `header.valign`, `spanner.valign`, `body.valign`, `col_first.valign`, `col_last.valign`, `row_group.valign`, `source_note.valign`, `footnotes.valign`
: Vertical alignment for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be "top", "middle", or "bottom".

`table.wrapText`, `title.wrapText`, `subtitle.wrapText`, `header.wrapText`, `spanner.wrapText`, `body.wrapText`, `col_first.wrapText`, `col_last.wrapText`, `row_group.wrapText`, `source_note.wrapText`, `footnotes.wrapText`
: Logical indicating whether to wrap text in the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively.

`table.indent`, `title.indent`, `subtitle.indent`, `header.indent`, `spanner.indent`, `body.indent`, `col_first.indent`, `col_last.indent`, `row_group.indent`, `source_note.indent`, `footnotes.indent`
: Indentation for the table, title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a numeric value indicating the number of spaces to indent. Defaults to NULL.

`table.locked`
: Logical indicating whether the table is locked.

`table.hidden`
: Logical indicating whether the table (Excel worksheet) is hidden.

`table.decimalPrecision`
: Numeric value indicating the number of decimal places to display in numeric columns.

`table.decimalCols`
: Character vector of column names that should have decimal formatting applied.

`table.lastRowBold`
: Logical indicating whether the last row of the table should be bold.

`table.width`, `col_first.width`, `col_last.width`, `row_group.width`
: Column widths for the table, first column, last column, and row group header respectively. Can be a numeric value indicating the width in points.

`table.widthOffset`
: Numeric value indicating the width offset for the table.

`title.border`, `subtitle.border`, `header.border`, `spanner.border`, `body.border`, `col_first.border`, `col_last.border`, `row_group.border`, `source_note.border`, `footnotes.border`, `border_header.border`
: Border style for the title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a string representing the border style. The `border_header.border` is used for the header border style.

`title.borderColour`, `subtitle.borderColour`, `header.borderColour`, `spanner.borderColour`, `body.borderColour`, `col_first.borderColour`, `col_last.borderColour`, `row_group.borderColour`, `source_note.borderColour`, `footnotes.borderColour`, `border_header.borderColour`, `border_outer.borderColour`
: Border color for the title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. Can be a hexadecimal color code (e.g., "#FF0000" for red) or a named color (e.g., "red"). The `border_header.borderColour` and `border_outer.borderColour` are used for the header and outer borders of the table.

`title.borderStyle`, `subtitle.borderStyle`, `header.borderStyle`, `spanner.borderStyle`, `body.borderStyle`, `col_first.borderStyle`, `col_last.borderStyle`, `row_group.borderStyle`, `source_note.borderStyle`, `footnotes.borderStyle`, `border_header.borderStyle`
: Border style for the title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively. The `border_header.borderStyle` is used for the header border style.

`title.textDecoration`, `subtitle.textDecoration`, `header.textDecoration`, `spanner.textDecoration`, `body.textDecoration`, `col_first.textDecoration`, `col_last.textDecoration`, `row_group.textDecoration`, `source_note.textDecoration`, `footnotes.textDecoration`
: Text decoration for the title, subtitle, header, spanner, body, first column, last column, row group header, source note, and footnotes respectively.

`title.height`, `subtitle.height`, `header.height`, `spanner.height`, `body.height`, `row_group.height`, `source_note.height`, `footnotes.height`, `border_bottom.height`
: Height for the title, subtitle, header, spanner, body, row group, source note, footnotes, and bottom border of the table respectively. Can be a numeric value indicating the height in points.

`body.numFmt`, `col_first.numFmt`, `col_last.numFmt`
: Numeric format for the body, first column, and last column respectively. Can be a string representing the numeric format.

Value

A `tsg` object with the specified facade settings applied as attributes.

Examples

```
person_record |>
```

```
generate_frequency(sex) |>
add_facade(table.offsetRow = 2, table.offsetCol = 1)
```

`add_facade_alt` *Add a facade to a tsg table (alternative way)*

Description

This function adds a facade to a `tsg`, an alternative way to allow dynamic values and programmatic evaluation.

Usage

```
add_facade_alt(data, ...)
```

Arguments

`data`	a `tsg` data frame
`...`	List of supported facade items (see `add_facade()`)

Value

A `tsg` object with the specified facade settings applied as attributes.

Examples

```
person_record |>
 generate_frequency(sex) |>
 add_facade_alt(table.offsetRow = 2, table.offsetCol = 1)
```

`add_footnote` *Add a footnote attribute to a table*

Description

Add a footnote attribute to a table

Usage

```
add_footnote(
  data,
  footnote,
  locations = NULL,
  placement = c("auto", "right", "left")
)
```

Arguments

data	A data frame, tibble, or tsg object to which a footnote attribute will be added.
footnote	The footnote text to be added (a single character string).
locations	Optional character vector of **column names** to anchor the footnote marker. When supplied, a footnote reference symbol is placed in those column headers. Column-level anchoring is supported in HTML and PDF output (via **gt**); XLSX and Word output include the text without cell-level markers. Default NULL renders the footnote as a table-level footer with no marker.
placement	Horizontal alignment of the footnote in the output footer. One of "auto" (default, equivalent to "left"), "right", or "left". Respected by all three output formats (HTML/PDF, XLSX, Word).

Value

The input data frame with an updated footnotes attribute (a list with elements $text, $placement, and $locations).

Examples

```
tbl <- person_record |> generate_frequency(sex)

# Whole-table footer, left-aligned (default)
tbl |> add_footnote("Source: National Survey 2023.")

# Right-aligned footer note
tbl |> add_footnote("Weighted estimates.", placement = "right")

# Footnote anchored to a specific column header (HTML/PDF)
tbl |> add_footnote("Unweighted count.", locations = "frequency")
```

---

add_row_total *Add a row total*

Description

Add a row total

Usage

```
add_row_total(
  data,
  position = c("bottom", "top"),
  label_total = "Total",
  fill = "-"
)
```

Arguments

data	A data frame, tibble, or tsg object to which a total row will be added.
position	Position to add the total row. Either "bottom" (default) or "top".
label_total	Label for the total row in the category column. Default is "Total".
fill	Character. Value to fill in for missing numeric columns in the total row. Default is "-".

Value

The input data frame with an additional row representing the total of numeric columns.

Examples

```
# Example data frame
df <- data.frame(
 category = c("A", "B", "C"),
 value1 = c(10, 20, 30),
 value2 = c(5, 15, 25)
)

df_with_total <- add_row_total(df)
df_with_total_top <- add_row_total(df, position = "top")
```

---

add_source_note *Add a source note attribute to a table*

Description

Add a source note attribute to a table

Usage

```
add_source_note(data, source_note)
```

Arguments

data	A data frame, tibble, or tsg object to which a source note attribute will be added.
source_note	The source note text to be added.

Value

The input data frame with an added source note attribute.

Examples

```
add_source_note(
  dplyr::starwars,
  source_note = "Source: Star Wars API (SWAPI)."
)
```

---

add_table_subtitle *Add a subtitle attribute to a table*

Description

Add a subtitle attribute to a table

Usage

```
add_table_subtitle(data, subtitle)
```

Arguments

data	A data frame, tibble, or tsg object to which a subtitle attribute will be added.
subtitle	The subtitle text to be added.

Value

The input data frame with an added subtitle attribute.

Examples

```
add_table_subtitle(
  dplyr::starwars,
  subtitle = "Star Wars Character Data"
)
```

---

add_table_title *Add a title attribute to a table*

Description

Add a title attribute to a table

Usage

```
add_table_title(data, title)
```

Arguments

data	A data frame, tibble, or tsg object to which a title attribute will be added.
title	The title text to be added.

Value

The input data frame with an added title attribute.

Examples

```
add_table_title(
  dplyr::starwars,
  title = "Star Wars Character Data"
)
```

---

collapse_list *Collapse a list of data frames or tibbles into a single data frame*

Description

Collapse a list of data frames or tibbles into a single data frame

Usage

```
collapse_list(
  data,
  ...,
  col_id = "category",
  label = NULL,
  pluck = NULL,
  as_proportion = FALSE,
  name_separator = "_",
  label_separator = "__"
)
```

Arguments

data	A list of data frames or tibbles to be collapsed.
...	Additional arguments passed to dplyr::filter().
col_id	The name of the column to be created for the category.
label	A label for the category column. If NULL, defaults to "Category".
pluck	A character vector of column names to pluck from the data frames. If NULL, all columns are retained.
as_proportion	If TRUE, the frequency values will be converted to proportions. Default is FALSE.
name_separator	A string to separate the names of the columns in the output data frame. Default is "_".

label_separator
: A string to separate the labels of the columns in the output data frame. Default is "__".

Value

A data frame with the specified category column and the frequency and percent columns for each category, along with any additional columns specified in pluck.

Examples

```
person_record |>
  generate_frequency(
    seeing,
    hearing,
    walking,
    remembering,
    self_caring,
    communicating
  ) |>
  collapse_list()
```

---

convert_factor *Convert labelled factors to regular factors*

Description

Convert labelled factors to regular factors

Usage

```
convert_factor(data)
```

Arguments

data	A data frame, tibble, or tsg object containing labelled factors.

Value

A data frame with labelled factors converted to regular factors.

Examples

```
df <- data.frame(
  category = haven::labelled(
    c(1, 2, 3),
    c("One" = 1, "Two" = 2, "Three" = 3)
   )
 )

df_converted <- convert_factor(df)
```

generate_crosstab *Generate cross-tabulation*

Description

Generate cross-tabulation

Usage

```
generate_crosstab(
  data,
  x,
  ...,
  add_total = TRUE,
  add_total_row = TRUE,
  add_total_column = TRUE,
  add_percent = TRUE,
  as_proportion = FALSE,
  percent_by_column = FALSE,
  name_separator = "_",
  label_separator = "__",
  label_total = "Total",
  label_total_column = NULL,
  label_total_row = NULL,
  label_na = "Not reported",
  label_as_group_name = TRUE,
  label_group_hierarchy = "All",
  include_na = TRUE,
  recode_na = "auto",
  group_separator = " - ",
  group_as_list = FALSE,
  group_as_hierarchy = FALSE,
  calculate_per_group = TRUE,
  expand_categories = TRUE,
  position_total = "bottom",
  sort_column_names = TRUE,
  collapse_list = FALSE,
  convert_factor = FALSE,
  multiple_columns = FALSE,
  multiple_columns_type = c("filtered", "stacked"),
  multiple_columns_filter = 1L,
  metadata = NULL
)
```

Arguments

data	A data frame (typically tibble) containing the variables to summarize.

x The variable to use for the rows of the cross-tabulation.

... Additional variable(s) to use for the columns of the cross-tabulation. If none are provided, a frequency table for x will be returned.

add_total Logical. If TRUE, adds total row and/or column.

add_total_row Logical. If TRUE, adds a total row.

add_total_column
Logical. If TRUE, adds a total column.

add_percent Logical. If TRUE, adds percent or proportion values to the table.

as_proportion Logical. If TRUE, displays proportions instead of percentages (range 0–1).

percent_by_column
Logical. If TRUE, percentages are calculated by column; otherwise, by row.

name_separator Character. Separator used when constructing variable names in the output.

label_separator
Character. Separator used when constructing labels in the output.

label_total Character. Label used for the total row/category.

label_total_column
Character. Label used for the total column/category.

label_total_row
Character. Label used for the total row/category.

label_na Character. Label to use for missing (NA) values.

label_as_group_name
Logical. If TRUE, uses the variable label of the grouping variable(s) as the name in the output list.

label_group_hierarchy
Character. Label applied to grand-total entries when group_as_hierarchy = TRUE. Can be a single string (applied to all group levels) or a named character vector keyed by group column name for per-variable labels (e.g. c(sex = "All sexes", employed = "All workers")). Defaults to "All".

include_na Logical. If TRUE, includes missing values in the cross table.

recode_na Character or NULL. Value used to replace missing values in labelled vectors; "auto" will determine a code automatically.

group_separator
Character. Separator used when concatenating group values in list output (if group_as_list = TRUE with a single group).

group_as_list Logical. If TRUE, output is a named list of cross-tabulation tables keyed by group value. With a single group the list is flat; with 2+ groups the list is nested. When combined with group_as_hierarchy = TRUE, a nested list with totals at each level is returned.

group_as_hierarchy
Logical. When TRUE (without group_as_list), inserts grand-total rows into the output. When TRUE together with group_as_list = TRUE, returns a nested named list with a total entry at each level; the total key is formatted as "{var_label}: {label_group_hierarchy}".

`calculate_per_group`
: Logical. If `TRUE`, calculates the cross-tabulation separately for each group defined by the grouping variable(s).

`expand_categories`
: Logical. If `TRUE`, ensures that all categories of `x` are represented in the output, even if they have zero counts.

`position_total`
: Character. Position of the total row/column; either `"bottom"` or `"top"` for rows, and `"right"` or `"left"` for columns.

`sort_column_names`
: Logical. If `TRUE`, sorts the column names in the output.

`collapse_list`
: Logical (NOT YET IMPLEMENTED). If `TRUE` and `group_as_list = TRUE`, collapses the list of frequency tables into a single data frame with group identifiers. See also `collapse_list()`.

`convert_factor`
: Logical. If `TRUE`, converts labelled variables to factors in the output. See also `convert_factor()`.

`multiple_columns`
: **[Experimental]** Logical or `NULL`. If `TRUE`, each column in `...` is treated as a binary indicator variable. Rows where the column equals `multiple_columns_filter` are counted per `x` category and presented as side-by-side frequency/percent columns in a single wide table. Requires at least 2 columns in `...`; if fewer are supplied a warning is issued and the function falls back to regular cross-tabulation mode.

`multiple_columns_type`
: Character. Controls how `multiple_columns = TRUE` handles the additional columns. `"filtered"` (default) treats each column as a binary indicator and produces a wide table with one column-pair per variable. `"stacked"` stacks results hierarchically: each column in `...` becomes a row group with `x` categories as columns; `multiple_columns_filter` is ignored in this mode.

`multiple_columns_filter`
: Scalar value (default `1L`). The value to filter on when `multiple_columns = TRUE` and `multiple_columns_type = "filtered"`. Ignored when `multiple_columns_type = "stacked"`.

`metadata`
: A named list with optional metadata to attach as attributes, e.g. `title`, `subtitle`, and `source_note`.

Value

A data frame or a list of data frames containing the cross-tabulation results. If `group_as_list` is `TRUE`, the output will be a list of data frames, one for each combination of grouping variable(s). Otherwise, a single data frame is returned. Each data frame includes counts and, if specified, percentages or proportions for each combination of `x` and the additional variables provided in `...`.

See Also

`generate_frequency()`, `generate_output()`, `rename_label()`, `remove_label()`

Examples

```
# Using built-in dataset `person_record`

# Basic usage
person_record |>
 generate_crosstab(marital_status, sex)


# Multiple variables
person_record |>
 generate_crosstab(
  sex,
  seeing,
  hearing,
  walking,
  remembering,
  self_caring,
  communicating
 )

 # Grouping
 person_record |>
   dplyr::group_by(sex) |>
   generate_crosstab(marital_status, employed, group_as_list = TRUE)

 # Nested list with totals at each level (group_as_list + group_as_hierarchy)
 person_record |>
   dplyr::group_by(sex) |>
   generate_crosstab(marital_status, employed,
     group_as_list = TRUE, group_as_hierarchy = TRUE)

# # Percent or proportion by row or column
person_record |>
 generate_crosstab(
   marital_status,
   sex,
   percent_by_column = TRUE
 )
```

`generate_frequency` *Generate frequency table*

Description

Creates frequency tables for one or more categorical variables, optionally grouped by other variables. The function supports various enhancements such as sorting, totals, percentages, cumulative statistics, handling of missing values, and label customization. It returns a single table or a list of frequency tables.

Usage

```
generate_frequency(
  data,
  ...,
  sort_value = TRUE,
  sort_desc = TRUE,
  sort_except = NULL,
  add_total = TRUE,
  add_percent = TRUE,
  add_cumulative = FALSE,
  add_cumulative_percent = FALSE,
  as_proportion = FALSE,
  include_na = TRUE,
  recode_na = "auto",
  position_total = c("bottom", "top"),
  calculate_per_group = TRUE,
  group_separator = " - ",
  group_as_list = FALSE,
  group_as_hierarchy = FALSE,
  label_group_hierarchy = "All",
  label_as_group_name = TRUE,
  label_stub = NULL,
  label_na = "Not reported",
  label_total = "Total",
  expand_categories = TRUE,
  convert_factor = FALSE,
  collapse_list = FALSE,
  top_n = NULL,
  top_n_only = FALSE,
  metadata = NULL
)
```

Arguments

data	A data frame (typically tibble) containing the variables to summarize.
...	One or more unquoted variable names (passed via tidy evaluation) for which to compute frequency tables.
sort_value	Logical. If TRUE, frequency values will be sorted.
sort_desc	Logical. If TRUE, sorts in descending order of frequency. If sort_value = FALSE, the category is sorted in ascending order.
sort_except	Optional character vector. Variables to exclude from sorting.
add_total	Logical. If TRUE, adds a total row or value to the frequency table.
add_percent	Logical. If TRUE, adds percent or proportion values to the table.
add_cumulative	Logical. If TRUE, adds cumulative frequency counts.
add_cumulative_percent	Logical. If TRUE, adds cumulative percentages (or proportions if as_proportion = TRUE).

as_proportion Logical. If TRUE, displays proportions instead of percentages (range 0–1).

include_na Logical. If TRUE, includes missing values in the frequency table.

recode_na Character or NULL. Value used to replace missing values in labelled vectors; "auto" will determine a code automatically.

position_total Character. Where to place the total row: "top" or "bottom".

calculate_per_group
Logical. If TRUE, calculates frequencies within groups defined in data (from group_by() or existing grouping).

group_separator
Character. Separator used when concatenating group values in list output (if group_as_list = TRUE with a single group).

group_as_list Logical. If TRUE, output is a named list of frequency tables keyed by group value. With a single group the list is flat; with 2+ groups the list is nested. When combined with group_as_hierarchy = TRUE, a nested list with totals at each level is returned.

group_as_hierarchy
Logical. When TRUE (without group_as_list), inserts grand-total rows into the output. When TRUE together with group_as_list = TRUE, returns a nested named list with a total entry at each level; the total key is formatted as "{var_label}: {label_group_hierarchy}".

label_group_hierarchy
Character. Label applied to grand-total entries when group_as_hierarchy = TRUE. Can be a single string (applied to all group levels) or a named character vector keyed by group column name for per-variable labels (e.g. c(sex = "All sexes", employed = "All workers")). Defaults to "All".

label_as_group_name
Logical. If TRUE, uses variable labels as names in the output list; otherwise, uses variable names.

label_stub Optional character vector used for labeling output tables (e.g., for export or display).

label_na Character. Label to use for missing (NA) values.

label_total Character. Label used for the total row/category.

expand_categories
Logical. If TRUE, ensures all categories (including those with zero counts) are included in the output.

convert_factor Logical. If TRUE, converts labelled variables to factors in the output. See also convert_factor().

collapse_list Logical. If TRUE and group_as_list = TRUE, collapses the list of frequency tables into a single data frame with group identifiers. See also collapse_list().

top_n Integer or NULL. If specified, limits the output to the top n categories by frequency.

top_n_only Logical. If TRUE and top_n is specified, only the top n categories are included, excluding others.

metadata A named list with optional metadata to attach as attributes, e.g. title, subtitle, and source_note.

Value

A frequency table (`tibble`, possibly nested) or a list of such tables. Additional attributes such as labels, metadata, and grouping information may be attached. The returned object is of class `"tsg"`.

See Also

`generate_crosstab()`, `generate_output()`, `rename_label()`, `remove_label()`

Examples

```
# Using built-in dataset `person_record`


# Basic usage
person_record |>
 generate_frequency(sex)

# Multiple variables
person_record |>
  generate_frequency(sex, age, marital_status)

# Grouping
person_record |>
  dplyr::group_by(sex) |>
  generate_frequency(marital_status)

# Output group as list
person_record |>
  dplyr::group_by(sex) |>
  generate_frequency(marital_status, group_as_list = TRUE)

# Nested list with totals at each level (group_as_list + group_as_hierarchy)
person_record |>
  dplyr::group_by(sex) |>
  generate_frequency(marital_status, group_as_list = TRUE, group_as_hierarchy = TRUE)

# Sorting

# default is TRUE
person_record |>
  generate_frequency(age, sort_value = TRUE)

# If FALSE, the output will be sorted by the variable values in ascending order.
person_record |>
  generate_frequency(age, sort_value = FALSE)

# See vignettes for more examples.
```

generate_output *Generate output in specified format (e.g., xlsx, html, pdf, word)*

Description

Generate output in specified format (e.g., xlsx, html, pdf, word)

Usage

```
generate_output(data, path, ..., format = c("xlsx", "html", "pdf", "word"))
```

Arguments

data	Preferably a tsg class object for best results. A data frame, tibble, and list are also supported.
path	File path to save the output. For HTML/PDF with a list and separate_files = TRUE (HTML) or any list (PDF), this is treated as a directory path.
...	Additional arguments passed to specific format functions: • "xlsx": passed to write_xlsx • "html": passed to write_html (requires the **gt** package) • "pdf": passed to write_pdf (requires the **gt** and **webshot2** packages) • "word": passed to write_docx (requires the **officer** and **flextable** packages)
format	Output format. One of "xlsx", "html", "pdf", or "word".

Value

Invisibly returns NULL. Called for its side-effect of writing output file(s).

Examples

```
# Generate an xlsx file from a tsg object
data <- generate_frequency(dplyr::starwars, sex)

dir_to <- tempdir()
generate_output(
  data,
  file.path(dir_to, "starwars_frequency.xlsx"),
  format = "xlsx"
)

unlink(file.path(dir_to, "starwars_frequency.xlsx"))
```

generate_template *Generate a template*

Description

Generate a template

Usage

```
generate_template(path, template = c("facade", "table-list"))
```

Arguments

path	A character string specifying the path where the template should be saved. If a directory is provided, the template will be saved with a default name based on the template type.
template	A character string specifying the type of template to generate. Options are "facade" for a YAML facade template or "table-list" for an Excel table list template.

Value

Void. A file path where the template has been saved.

Examples

```
template_path_facade <- tempfile(fileext = ".yaml")
generate_template(template_path_facade, template = "facade")

template_path_table_list <- tempfile(fileext = ".xlsx")
generate_template(template_path_table_list, template = "table-list")

unlink(template_path_facade)
unlink(template_path_table_list)
```

get_tsg_facade *Get a facade from the package or a file*

Description

Get a facade from the package or a file

Usage

```
get_tsg_facade(facade = "default", which = c("xlsx", "html", "docx", "pdf"))
```

Arguments

facade	A character string specifying the name of the facade to retrieve. Defaults to "default". The facade is a YAML file that defines the styling and layout of the table
which	A character string specifying the format of the facade to retrieve. Options are "xlsx", "pdf", or "html". Defaults to "xlsx".

Value

A list containing the facade settings for the specified format. The facade includes styling attributes such as font size, color, border styles, and background fills for different parts of the table.

Examples

```
# Default facade
get_tsg_facade()

# Other built-in facade
get_tsg_facade("yolo")
```

---

person_record *Sample dataset of persons*

Description

This is a synthetic dataset containing person information for demonstration purposes.

Usage

```
person_record
```

Format

A labelled data frame with 2918 rows and 11 variables:

person_id Numeric identifier for each person

sex Factor indicating the sex of the person

age Numeric age of the person

marital_status Factor indicating marital status

employed Employment status

seeing Functional difficulty in seeing

hearing Functional difficulty in hearing

walking Functional difficulty in walking

remembering Functional difficulty in remembering

self_caring Functional difficulty in self-caring

communicating Functional difficulty in communicating

Examples

```
person_record
```

`remove_label` *Remove data labels*

Description

Remove data labels

Usage

```
remove_label(data, ...)
```

Arguments

`data`	A data frame or tibble from which to remove labels.
`...`	A character vector of column names from which to remove labels. If no columns are specified, all labels will be removed.

Value

A data frame or tibble with the specified labels removed. If no columns are specified, all labels will be removed.

Examples

```
person_record |>
  generate_frequency(
    seeing,
    hearing,
    walking,
    remembering,
    self_caring,
    communicating
  ) |>
  collapse_list() |>
  remove_label()
```

remove_labels *Remove all labels*

Description

Remove all labels

Usage

```
remove_labels(data, ...)
```

Arguments

data	A data frame or tibble from which to remove all labels.
...	A character vector of column names from which to remove labels. If no columns are specified, all labels will be removed.

Value

A data frame or tibble with all labels removed. If no columns are specified, all labels will be removed.

Examples

```
person_record |>
  generate_frequency(
    seeing,
    hearing,
    walking,
    remembering,
    self_caring,
    communicating
  ) |>
  collapse_list() |>
  remove_labels()
```

rename_label *Rename data labels*

Description

Rename data labels

Usage

```
rename_label(data, ...)
```

Arguments

data	A data frame or tibble to rename labels in.
...	A named list of labels to rename. The names should match the column names in the data, and the values should be the new labels.

Value

A data frame or tibble with the specified labels renamed.

Examples

```
person_record |>
  generate_frequency(
    seeing,
    hearing,
    walking,
    remembering,
    self_caring,
    communicating
  ) |>
  collapse_list() |>
  rename_label(category = "Functional difficulty")
```

---

write_docx *Write a tsg table (or list of tables) to a Word (.docx) file*

Description

Saves one or more `tsg` tables as a Word document using the **officer** and **flextable** packages (both must be installed).

Usage

```
write_docx(
  data,
  path,
  ...,
  title = NULL,
  subtitle = NULL,
  source_note = NULL,
  footnotes = NULL,
  separate_files = FALSE,
  names_separator = "__",
  facade = get_tsg_facade(which = "docx")
)
```

Arguments

data	A tsg or data frame, or a named list of them.
path	File path for the Word output. A .docx extension is added if missing. When separate_files = TRUE the path (minus extension) is used as the directory.
...	Currently unused; reserved for future arguments.
title	Optional title string (overrides data attribute).
subtitle	Optional subtitle string.
source_note	Optional source note string.
footnotes	Optional character vector of footnotes.
separate_files	Logical. When data is a list, FALSE (default) writes all tables into a single combined .docx; TRUE writes one .docx per table inside a subdirectory.
names_separator	Column name separator for detecting cross-tab spanners. Default "__".
facade	Styling options. Defaults to the global tsg facade.

Details

When data is a named list and separate_files = FALSE (default), all tables are written into a single .docx document, one per page. Set separate_files = TRUE to write one .docx file per table inside a subdirectory derived from path.

Value

Invisibly returns NULL.

---

write_html *Write a tsg table (or list of tables) to an HTML file*

Description

Saves one or more tsg tables as an HTML file using the **gt** package (which is already a hard dependency of **tsg**).

Usage

```
write_html(
  data,
  path,
  ...,
  title = NULL,
  subtitle = NULL,
  source_note = NULL,
  footnotes = NULL,
  separate_files = FALSE,
```

```
  include_table_list = FALSE,
  names_separator = "__",
  facade = get_tsg_facade(which = "html")
)
```

Arguments

data	A tsg or data frame, or a named list of them.
path	File path for the HTML output. A .html extension is added if missing. When separate_files = TRUE the path (minus extension) is used as the directory.
...	Additional arguments passed to tsg_to_gt().
title	Optional title string (overrides data attribute).
subtitle	Optional subtitle string.
source_note	Optional source note string.
footnotes	Optional character vector of footnotes.
separate_files	Logical. When TRUE and data is a list, each table is saved to its own HTML file inside a subdirectory derived from path.
include_table_list	Logical. When TRUE and data is a named list with separate_files = FALSE, prepends a clickable table of contents.
names_separator	Column name separator for spanners. Default "__".
facade	Styling options. Defaults to the global tsg facade.

Details

When data is a named list and separate_files = FALSE (default), all tables are written into a single self-contained HTML document. Set include_table_list = TRUE to prepend a clickable table-of-contents. Set separate_files = TRUE to write one HTML file per table into a subdirectory.

Value

Invisibly returns NULL.

---

write_pdf *Write a tsg table (or list of tables) to a PDF file*

Description

Saves tables as PDF using gt::gtsave(), which requires the **webshot2** package (and a Chromium installation reachable by **chromote**).

Usage

```
write_pdf(
  data,
  path,
  ...,
  title = NULL,
  subtitle = NULL,
  source_note = NULL,
  footnotes = NULL,
  separate_files = TRUE,
  names_separator = "__",
  facade = get_tsg_facade(which = "html")
)
```

Arguments

data	A tsg or data frame, or a named list of them.
path	File path for the PDF output. A .pdf extension is added if missing. When data is a list and separate_files = TRUE, this is used as a directory.
...	Additional arguments passed to tsg_to_gt().
title	Optional title string (overrides data attribute).
subtitle	Optional subtitle string.
source_note	Optional source note string.
footnotes	Optional character vector of footnotes.
separate_files	Logical. When data is a list, TRUE (default) saves each table as a separate PDF file inside a subdirectory; FALSE merges them into a single PDF (requires **qpdf**).
names_separator	Column name separator for spanners. Default "__".
facade	Styling options. Defaults to the global tsg facade.

Details

When data is a named list, the default (separate_files = TRUE) writes each table to its own .pdf file inside a directory. Set separate_files = FALSE to merge all tables into a single PDF (requires the **qpdf** package).

Value

Invisibly returns NULL.

write_xlsx *Write Data to Excel with Titles, Notes, and Styling*

Description

Exports a data frame or a list of data frames to one or multiple Excel files, with support for titles, subtitles, source notes, footnotes, grouping, and custom styles. It leverages the openxlsx package to create styled Excel reports suitable for presentation.

Usage

```
write_xlsx(
  data,
  path,
  ...,
  sheet_name = NULL,
  title = NULL,
  subtitle = NULL,
  source_note = NULL,
  footnotes = NULL,
  separate_files = FALSE,
  collapse_list = FALSE,
  row_group_as_column = FALSE,
  names_separator = "__",
  include_table_list = FALSE,
  table_list_reference = NULL,
  facade = get_tsg_facade()
)
```

Arguments

data	A data.frame, tibble, or a named list of them. When a list is provided: • If separate_files = FALSE, each element is written to a separate sheet in one Excel file. • If separate_files = TRUE, each element is written to its own Excel file.
path	A file path (if separate_files = FALSE) or directory path (if separate_files = TRUE) where the Excel file(s) will be saved. File extension .xlsx is automatically added if missing.
...	Additional arguments passed to openxlsx::createWorkbook() and openxlsx::addWorksheet().
sheet_name	Optional name for the Excel sheet. Ignored if data is a list and separate_files = FALSE.
title	Optional title displayed above the data in each sheet or file.
subtitle	Optional subtitle displayed under the title.
source_note	Optional source note displayed below the data.

footnotes	Optional character vector of footnotes to display below the source note.
separate_files	Logical. If TRUE, each list item in data is saved as a separate Excel file.
collapse_list	Logical. If TRUE, a list of data frames will be merged into one sheet (if applicable).
row_group_as_column	Logical. If TRUE, row groupings are included as columns instead of grouped titles.
names_separator	Character used to separate column names when dealing with nested or grouped headers.
include_table_list	Logical. If TRUE, a table list reference is included in the Excel file.
table_list_reference	A data frame containing the table list reference. If NULL, it will be generated from data.
facade	A list of styling options (colors, fonts, sizes, border styles, etc.). Defaults to the global option tsg.options.facade.

Details

This function supports advanced Excel formatting including:

- Grouped headers
- Dynamic column widths
- Styled titles, subtitles, source notes, and footnotes
- Border styling (inner, outer, header)

The function is designed to handle export needs in professional and reporting contexts.

Value

Invisibly returns NULL. The function is called for its side-effect of writing Excel file(s).

Examples

```
data <- tsg::generate_frequency(dplyr::starwars, sex)

dir_to <- tempfile()
write_xlsx(
  data,
  file.path(dir_to, "starwars_frequency.xlsx")
 )
```

Index