Package 'tabstats'

May 8, 2026

Type Package

Title A Lightweight Toolkit for Displaying Customizable Tables

Version 0.1.0

Maintainer Joshua Marie <joshua.marie.k@gmail.com>

Description A lightweight toolkit that provides functions for printing tables from input data in the R console or terminal with customizable formatting. Supported outputs include American Psychological Association (APA)-style tables (American Psychological Association, 2020, ISBN:9781433832178), correlation matrices, contingency tables, and two-column summary tables.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

URL https://github.com/joshuamarie/tabstats

BugReports https://github.com/joshuamarie/tabstats/issues

Imports cli, tibble, vctrs, utils, stats

Suggests broom, rstatix, glue, tidyr, testthat (>= 3.0.0), knitr, rmarkdown, fansi, crayon, dplyr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Joshua Marie [aut, cre]

Repository CRAN

Date/Publication 2026-03-24 10:10:03 UTC

Contents

`cm_style` *Style specification for corr_matrix()*

Description

Constructs a validated style object for use with `corr_matrix()`.

Usage

```
cm_style(...)
```

Arguments

`...`	Named style entries. Names should match the extra field names passed to `new_corr_data()` (e.g. `rho`, `pval`, `bf`), or the reserved keys `title` and `border_text`.

Value

An object of class `c("cm_style", "tabstats_style")`.

Examples

```
cm_style(rho = "blue_bold", pval = "red", title = "bold")
cm_style(rho = \(x) cli::col_cyan(x))
```

corr_matrix *Display a Correlation Matrix Table in the Console*

Description

Display a Correlation Matrix Table in the Console

Usage

```
corr_matrix(
  display,
  title = NULL,
  diag_1 = TRUE,
  digits = 3,
  layout_view = FALSE,
  layout_center = FALSE,
  center_table = FALSE,
  border_char = getOption("tab_default")$border_char,
  style = list(),
  ...
)
```

Arguments

display	A corr_spec object from new_corr_data(), or a plain symmetric matrix (e.g. from cor()).
title	Label shown in the title (e.g. "Pearson Correlation Matrix"). Auto-detected from a title attribute on the spec if present.
diag_1	If TRUE, diagonal cells always show "1". Default TRUE.
digits	Decimal places for numeric formatting. Default 3.
layout_view	Show a layout legend box above the table? Default FALSE.
layout_center	Center the layout box in the terminal? Default FALSE.
center_table	Center table in terminal? Default FALSE.
border_char	Border character. Default from getOption("tab_default").
style	A cm_style() object. Keys match the extra field names passed to new_corr_data() (e.g. rho, pval, bf), plus title and border_text.
...	Reserved for future use.

Value

Invisibly returns the rendered character matrix.

Examples

```
# From a plain correlation matrix, e.g. using `cor()`
corr_matrix(cor(mtcars[, 1:4]), title = "Pearson Correlation Matrix")

# Customizable example
spec = new_corr_data(
    var1 = c("a", "a", "b"),
    var2 = c("b", "c", "c"),
    rho = c("0.89", "0.79", "0.66"),
    pval = c("<0.001", "<0.001", "<0.001")
)
corr_matrix(spec, title = "Pearson Correlation Matrix", layout_view = TRUE)
```

---

cross_table *Generate and Display a Cross Tabulation Table*

Description

Generate and Display a Cross Tabulation Table

Usage

```
cross_table(
  data,
  percentage = NULL,
  layout = TRUE,
  expected = TRUE,
  layout_center = FALSE,
  header_underline = FALSE,
  digits = NULL,
  center_table = FALSE,
  style = NULL,
  ...
)
```

Arguments

data	A matrix or table of observed frequencies.
percentage	Which percentages to show. Options: TRUE/"all", "by_row", "by_col", "by_total", FALSE/NULL.
layout	Show layout legend box? Default TRUE.
expected	Show expected frequencies? Default TRUE.
layout_center	Center the layout box? Default FALSE.
header_underline	Shorten the underline beneath the column header? Default FALSE.

digits	Named list with keys ex, row_pct, col_pct, total_pct. Falls back to getOption("ct_digits").
center_table	Center entire table in terminal? Default FALSE.
style	Named list supplied using ct_style().
...	Reserved for future use.

Value

Invisibly returns the formatted cross-tabulation matrix.

Examples

```
cross_table(matrix(c(10, 20, 30, 40), nrow = 2))
cross_table(matrix(c(10, 20, 30, 40), nrow = 2), percentage = "all")
```

---

ct_style *Style specification for cross_table()*

Description

Constructs a validated style object for use with cross_table().

Usage

```
ct_style(
  observed = NULL,
  expected = NULL,
  row_percentage = NULL,
  col_percentage = NULL,
  total_percentage = NULL,
  total = NULL,
  title = NULL,
  border_text = NULL
)
```

Arguments

observed	Style for observed frequency cells.
expected	Style for expected frequency values.
row_percentage	Style for row percentage values.
col_percentage	Style for column percentage values.
total_percentage	Style for grand total percentage values.
total	Style for total row/column cells.
title	Style for the table title.
border_text	Style for the horizontal border lines.

Value

An object of class `c("ct_style", "tabstats_style")`.

Examples

```
ct_style(observed = "blue", expected = "yellow", title = "bold")
ct_style(observed = \(ctx) cli::col_green(ctx$formatted_text))
```

`new_corr_data` *Build a correlation display specification*

Description

Constructs a structured spec object consumed by `corr_matrix()`. Always requires `var1` and `var2` — the pair pattern they encode determines which triangle(s) of the matrix are filled:

Usage

```
new_corr_data(var1, var2, ...)
```

Arguments

`var1`	Character vector of first variable names per pair.
`var2`	Character vector of second variable names per pair.
`...`	Named character vectors of equal length to `var1`/`var2`. Each becomes one display row inside the cell (e.g. `rho`, `pval`, `bf`).

Details

Pattern	Fills
`var1 < var2`	Lower triangle only
`var1 <= var2`	Lower triangle + diagonal
`var1 > var2`	Upper triangle only
`var1 >= var2`	Upper triangle + diagonal
`var1 != var2`	Both triangles, no diagonal
`var1 == var2`	Full matrix (diag forced to 1)

All additional named vectors become display rows inside each cell, rendered in the order they are supplied.

Value

An object of class `corr_spec`.

Examples

```
new_corr_data(
    var1 = c("a", "a", "b"),
    var2 = c("b", "c", "c"),
    rho = c("0.89", "0.79", "0.66"),
    pval = c("<0.001", "<0.001", "<0.001")
)
```

`sm_style` *Style specification for table_summary()*

Description

Constructs a validated style object for use with `table_summary()`.

Usage

```
sm_style(
  left_col = NULL,
  right_col = NULL,
  border_text = NULL,
  title = NULL,
  sep = NULL
)
```

Arguments

`left_col`	Style for the left column. A string (e.g. `"blue_bold"`) or a function `\(x) ...`.
`right_col`	Style for the right column.
`border_text`	Style for the horizontal border lines.
`title`	Style for the title text.
`sep`	A single character used as the column separator (e.g. `"|"`).

Value

An object of class `c("sm_style", "tabstats_style")`.

Examples

```
sm_style(left_col = "blue_bold", right_col = "green", title = "bold")
sm_style(left_col = \(x) cli::col_red(x), sep = "|")
```

`table_default` *Display a formatted table in the console*

Description

Display a formatted table in the console

Usage

```
table_default(
  x,
  justify_cols = "center",
  digits = 3,
  digits_by_col = NULL,
  scientific = FALSE,
  na_print = "",
  min_width = NULL,
  border_char = options("tab_default")$tab_default$border_char,
  show_row_names = FALSE,
  center_table = FALSE,
  n_space = 2,
  title = NULL,
  style_colnames = NULL,
  style_columns = NULL,
  nrows = getOption("tab_default")$nrows,
  vb = list(),
  auto_wrap = TRUE,
  wrap_threshold = 1,
  ...
)
```

Arguments

`x`	A data frame or tibble.
`justify_cols`	Alignment: a single string, vector, or named list of "left"/"right"/"center".
`digits`	Digits to round numeric columns to. Default `3`.
`digits_by_col`	Named list of per-column digit overrides.
`scientific`	Display numerics in scientific notation? Default `FALSE`.
`na_print`	String for missing values. Default `""`.
`min_width`	Minimum column width. Default `NULL`.
`border_char`	Character for borders. Default `"\u2500"`.
`show_row_names`	Show row names? Default `FALSE`.
`center_table`	Center table in terminal? Default `FALSE`.
`n_space`	Spaces between columns. Default `2`.

title	Optional title string above the table. from td_style(), or a named list where each name is a column name or "title", and each value is either a cli style string (e.g. "blue_bold") or a function \(ctx) ... receiving a context list.
style_colnames	Styling for column header cells. A td_style object from td_style(), or a named list where each name is a column name or "title", and each value is either a cli style string (e.g. "blue_bold") or a function \(ctx) ... receiving a context list.
style_columns	Styling for data cells. A td_style object from td_style(), or a named list where each name is a column name or column index as a string, and each value is a cli style string or a function \(ctx) ... receiving a context list with elements value, formatted_value, col_name, col_index, is_header, data, justify, and width.
nrows	Max rows to display before truncation.
vb	Vertical border spec: list(char = "\u2502", after = c(1, 3)).
auto_wrap	Auto-wrap wide tables? Default TRUE.
wrap_threshold	Fraction of console width before wrapping. Default 1.
...	Reserved for future use.

Value

Invisibly returns the input data as a character matrix after formatting has been applied. The function is called primarily for its side effect of printing a styled table to the R console. Returns invisible(NULL) early if the input has 0 rows and 0 columns, or if it has 0 columns.

Examples

```
table_default(head(mtcars))
table_default(head(mtcars), style_columns = td_style(mpg = "cyan", cyl = "magenta"))
```

---

table_summary *Summarize and Display a Two-Column Data Frame as a Formatted Table*

Description

This function takes a two-column data frame and formats it into a summary-like table. The table can be optionally split into two parts, centered, and given a title. It is useful for displaying summary information in a clean, tabular format. The function also supports styling with ANSI colors and text formatting through the {cli} package and column alignment options.

Usage

```
table_summary(
  data,
  title = NULL,
  l = NULL,
  header = FALSE,
  center_table = FALSE,
  border_char = "-",
  style = list(),
  align = NULL,
  ...
)
```

Arguments

`data`	A data frame with exactly two columns. The data to be summarized and displayed.
`title`	A character string. An optional title to be displayed above the table.
`l`	An integer. The number of rows to include in the left part of a split table. If `NULL`, the table is not split.
`header`	A logical value. If `TRUE`, the column names of `data` are displayed as a header.
`center_table`	A logical value. If `TRUE`, the table is centered in the terminal.
`border_char`	Character used for borders. Default is `"\u2500"`.
`style`	A list controlling the visual styling of table elements using ANSI formatting. Can include the following components: • `left_col`: Styling for the left column values. • `right_col`: Styling for the right column values. • `border_text`: Styling for the border. • `title`: Styling for the title. • `sep`: Separator character between left and right column. Each style component can be either a predefined style string (e.g., "blue", "red_italic", "bold") or a function that takes a context list with/without a `value` element and returns the styled text.
`align`	Controls the alignment of column values. Can be specified in three ways: • A single string: affects only the left column (e.g., "left", "center", "right"). • A vector of two strings: affects both columns in order (e.g., c("left", "right")). • A list with named components: explicitly specifies alignment for each column
`...`	Additional arguments (currently unused).

Value

This function does not return a value. It prints the formatted table to the console.

Examples

```
# Create a sample data frame
df = data.frame(
    Category = c("A", "B", "C", "D", "E"),
    Value = c(10, 20, 30, 40, 50)
)

# Display the table with a title and header
table_summary(df, title = "Sample Table", header = TRUE)

# Split the table after the second row and center it
table_summary(df, l = 2, center_table = TRUE)

# Use styling and alignment
table_summary(
    df,
    header = TRUE,
    style = list(
        left_col = "blue_bold",
        right_col = "red",
        title = "green",
        border_text = "yellow"
    ),
    align = c("center", "right")
)

# Use custom styling with lambda functions
table_summary(
    df,
    header = TRUE,
    style = sm_style(
        left_col = \(ctx) cli::col_red(ctx), # ctx$value is another option
        right_col = \(ctx) cli::col_blue(ctx)
    ),
    align = list(left_col = "left", right_col = "right")
)
```

`tabstats_options` *Manage package options*

Description

This function allows retrieving or modifying the package options across different categories. If called without arguments, it returns all option categories and their values. If `category` is provided alone, it returns all options in that category. If `category` and `option` are provided, it returns the specific option value. If all three parameters are provided, it updates the specified option.

Usage

```
tabstats_options(category = NULL, option = NULL, value = NULL)
```

Arguments

category	A character string specifying the option category (e.g., "tab_default", "tab_digits"). If omitted, returns all option categories.
option	A character string specifying the option to retrieve or modify within the category. For backward compatibility, you can also use a specific option name directly as the category parameter.
value	The new value to assign to the specified option. If NULL, the function returns the current value.

Value

If no arguments are provided, returns all option categories and their values. If only category is provided, returns all options in that category. If category and option are provided without value, returns the current value of that option. If all parameters are provided, updates the option and returns the updated option category list invisibly.

Examples

```
# Get all options across all categories
tabstats_options()

# Get all options in the "tab_default" category
tabstats_options("tab_default")

# Get all options in the "tab_digits" category
tabstats_options("tab_digits")

# Get a specific option
tabstats_options("tab_default", "vb_top")
tabstats_options("tab_digits", "ex")

# Using backward compatibility (system will find the right category)
tabstats_options("vb_top")
tabstats_options("ex")

# Modify an option
tabstats_options("tab_default", "border_char", "+")
tabstats_options("tab_digits", "ex", 2)

# Using backward compatibility for modification
tabstats_options("border_char", "+")
tabstats_options("ex", 2)
```

---

td_style *Style specification for table_default()*

Description

Constructs a validated style object for use with `table_default()`.

Usage

```
td_style(...)
```

Arguments

`...`	Named style entries. Each name must be a column name, a column index as a string (e.g. `"1"`), or `"title"`. Each value is a string or a function `\(ctx) ...` where `ctx` is a context list.

Value

An object of class `c("td_style", "tabstats_style")`.

Examples

```
td_style(mpg = "cyan", cyl = "magenta")
td_style(mpg = \(ctx) cli::col_red(ctx$value), title = "bold")
```

Index