

# Package: tabstats (via r-universe)

June 7, 2026

**Type** Package

**Title** A Lightweight Toolkit for Displaying Customizable Tables

**Version** 0.2.0.9000

**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

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**URL** <https://github.com/s7-stats/tabstats>

**BugReports** <https://github.com/s7-stats/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

**Repository** <https://s7-stats.r-universe.dev>

**Date/Publication** 2026-06-07 05:04:09 UTC

**RemoteUrl** <https://github.com/s7-stats/tabstats>

**RemoteRef** HEAD

**RemoteSha** 4418bc05489357c3eebb3c211d9829bcf87bd869

## Contents

cm_style . . . . .	2
cross_table . . . . .	3
ct_style . . . . .	4
new_pairwise_data . . . . .	5
pairwise_matrix . . . . .	6
sm_style . . . . .	7
table_default . . . . .	8
table_summary . . . . .	10
tabstats_options . . . . .	12
td_style . . . . .	13
<b>Index</b>	<b>14</b>

---

cm_style	<i>Style specification for corr_matrix()</i>
----------	--

---

### 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))
```

---

cross_table	<i>Generate and Display a Cross Tabulation Table</i>
-------------	--

---

**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 <code>ct_style()</code> .
...	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")
```

---

<code>ct_style</code>	<i>Style specification for <code>cross_table()</code></i>
-----------------------	---

---

### 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

<code>observed</code>	Style for observed frequency cells.
<code>expected</code>	Style for expected frequency values.
<code>row_percentage</code>	Style for row percentage values.
<code>col_percentage</code>	Style for column percentage values.
<code>total_percentage</code>	Style for grand total percentage values.
<code>total</code>	Style for total row/column cells.
<code>title</code>	Style for the table title.
<code>border_text</code>	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_pairwise_data	<i>Build a pairwise matrix display specification</i>
-------------------	--

---

**Description**

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

**Usage**

```
new_pairwise_data(var1, var2, ...)
```

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

**Arguments**

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

**Details**

Pattern	Fills
<code>var1 &lt; var2</code>	Lower triangle only
<code>var1 &lt;= var2</code>	Lower triangle + diagonal
<code>var1 &gt; var2</code>	Upper triangle only
<code>var1 &gt;= var2</code>	Upper triangle + diagonal
<code>var1 != var2</code>	Both triangles, no diagonal
<code>var1 == var2</code>	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 `pairwise_spec`.

**Examples**

```
new_pairwise_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")
)
```

---

pairwise_matrix	<i>Display a Pairwise Matrix Table in the Console</i>
-----------------	---

---

## Description

Display a Pairwise Matrix Table in the Console

## Usage

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

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 pairwise_spec object from new_pairwise_data(), or a plain symmetric matrix (e.g. from cor()).
title	Label shown above the table. Auto-detected from a title attribute on the spec if present; falls back to name.
name	Short name used in the layout legend box header. Default "Pairwise Matrix".
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 <code>getOption("tab_default")</code> .
style	A style list. Keys match the extra field names passed to <code>new_pairwise_data()</code> (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
pairwise_matrix(cor(mtcars[, 1:4]), title = "Pearson Correlation Matrix")

# Customizable example
spec = new_pairwise_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")
)
pairwise_matrix(spec, title = "Pearson Correlation Matrix", layout_view = TRUE)
```

---

sm_style	<i>Style specification for table_summary()</i>
----------	--

---

### 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) \dots
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	<i>Display a formatted table in the console</i>
---------------	---

---

**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 <code>td_style()</code> , 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 <code>\(ctx) ...</code> receiving a context list.
style_colnames	Styling for column header cells. A <code>td_style</code> object from <code>td_style()</code> , 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 <code>\(ctx) ...</code> receiving a context list.
style_columns	Styling for data cells. A <code>td_style</code> object from <code>td_style()</code> , 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 <code>\(ctx) ...</code> receiving a context list with elements <code>value</code> , <code>formatted_value</code> , <code>col_name</code> , <code>col_index</code> , <code>is_header</code> , <code>data</code> , <code>justify</code> , and <code>width</code> .
nrows	Max rows to display before truncation.
vb	Vertical border spec: <code>list(char = "\u2502", after = c(1, 3))</code> .
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	<i>Summarize and Display a Two-Column Data Frame as a Formatted Table</i>
---------------	---

---

### 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

<code>data</code>	A data frame with exactly two columns. The data to be summarized and displayed.
<code>title</code>	A character string. An optional title to be displayed above the table.
<code>l</code>	An integer. The number of rows to include in the left part of a split table. If <code>NULL</code> , the table is not split.
<code>header</code>	A logical value. If <code>TRUE</code> , the column names of data are displayed as a header.
<code>center_table</code>	A logical value. If <code>TRUE</code> , the table is centered in the terminal.
<code>border_char</code>	Character used for borders. Default is <code>"\u2500"</code> .
<code>style</code>	A list controlling the visual styling of table elements using ANSI formatting. Can include the following components: <ul style="list-style-type: none"> <li><code>left_col</code>: Styling for the left column values.</li> <li><code>right_col</code>: Styling for the right column values.</li> <li><code>border_text</code>: Styling for the border.</li> <li><code>title</code>: Styling for the title.</li> <li><code>sep</code>: Separator character between left and right column.</li> </ul> Each style component can be either a predefined style string (e.g., <code>"blue"</code> , <code>"red_italic"</code> , <code>"bold"</code> ) 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	<i>Manage package options</i>
------------------	-------------------------------

---

### 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

<code>category</code>	A character string specifying the option category (e.g., "tab_default", "tab_digits"). If omitted, returns all option categories.
<code>option</code>	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.
<code>value</code>	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	<i>Style specification for table_default()</i>
----------	--

---

## 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

`cm_style`, 2  
`corr_matrix(pairwise_matrix)`, 6  
`corr_matrix()`, 2  
`cross_table`, 3  
`cross_table()`, 4  
`ct_style`, 4  
`ct_style()`, 3  
  
`new_corr_data(new_pairwise_data)`, 5  
`new_corr_data()`, 2  
`new_pairwise_data`, 5  
  
`pairwise_matrix`, 6  
  
`sm_style`, 7  
  
`table_default`, 8  
`table_default()`, 13  
`table_summary`, 10  
`table_summary()`, 7  
`tabstats_options`, 12  
`td_style`, 13  
`td_style()`, 9