Package: tinytable (via r-universe)

December 25, 2024

Type Package

Title Simple and Configurable Tables in 'HTML', 'LaTeX', 'Markdown', 'Word', 'PNG', 'PDF', and 'Typst' Formats

Description Create highly customized tables with this simple and dependency-free package. Data frames can be converted to 'HTML', 'LaTeX', 'Markdown', 'Word', 'PNG', 'PDF', or 'Typst' tables. The user interface is minimalist and easy to learn. The syntax is concise. 'HTML' tables can be customized using the flexible 'Bootstrap' framework, and 'LaTeX' code with the 'tabularray' package.

Version 0.6.1

Imports methods

Depends R (>= 4.1.0)

Enhances knitr

Suggests base64enc, data.table (>= 1.15.2), fansi, ggplot2, gh, marginaleffects, markdown, pandoc, quarto, rmarkdown, rstudioapi, scales, stringi, tibble, tinysnapshot, tinytest, tinytex, webshot2

URL https://vincentarelbundock.github.io/tinytable/

BugReports https://github.com/vincentarelbundock/tinytable/issues

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Vincent Arel-Bundock [aut, cre] (<https://orcid.org/0000-0003-2042-7063>)

Maintainer Vincent Arel-Bundock <vincent.arel-bundock@umontreal.ca>

Repository CRAN

Date/Publication 2024-11-18 12:50:18 UTC

Contents

`tinytable-package` *Simple and Configurable Tables in 'HTML', 'LaTeX', 'Markdown', 'Word', 'PNG', 'PDF', and 'Typst' Formats*

Description

Create highly customized tables with this simple and dependency-free package. Data frames can be converted to 'HTML', 'LaTeX', 'Markdown', 'Word', 'PNG', 'PDF', or 'Typst' tables. The user interface is minimalist and easy to learn. The syntax is concise. 'HTML' tables can be customized using the flexible 'Bootstrap' framework, and 'LaTeX' code with the 'tabularray' package.

Package Content

Index of help topics:

```
format_tt               Format columns of a data frame
group_tt                Spanning labels to identify groups of rows or
                        columns
plot_tt                 Insert images and inline plots into tinytable
                        objects
print.tinytable         Print, display, or convert a tinytable object
rbind2,tinytable,tinytable-method
                        Combine 'tinytable' objects by rows
                        (vertically)
save_tt                 Save a Tiny Table to File
style_tt                Style a Tiny Table
theme_tt                Themes for 'tinytable'
tinytable-package       Simple and Configurable Tables in 'HTML',
                        'LaTeX', 'Markdown', 'Word', 'PNG', 'PDF', and
                        'Typst' Formats
tt                      Draw a Tiny Table
```

Maintainer

Vincent Arel-Bundock <vincent.arel-bundock@umontreal.ca>

Author(s)

Vincent Arel-Bundock [aut, cre] (<https://orcid.org/0000-0003-2042-7063>)

`format_tt` *Format columns of a data frame*

Description

This function formats the columns of a data frame based on the column type (logical, date, numeric). It allows various formatting options like significant digits, decimal points, and scientific notation. It also includes custom formatting for date and boolean values. If this function is applied several times to the same cell, the last transformation is retained and the previous calls are ignored, except for the `escape` argument which can be applied to previously transformed data.

Usage

```
format_tt(
  x,
  i = NULL,
  j = NULL,
  digits = get_option("tinytable_format_digits", default = NULL),
  num_fmt = get_option("tinytable_format_num_fmt", default = "significant"),
  num_zero = get_option("tinytable_format_num_zero", default = FALSE),
  num_suffix = get_option("tinytable_format_num_suffix", default = FALSE),
  num_mark_big = get_option("tinytable_format_num_mark_big", default = ""),
  num_mark_dec = get_option("tinytable_format_num_mark_dec", default =
    getOption("OutDec", default = ".")),
  date = get_option("tinytable_format_date", default = "%Y-%m-%d"),
  bool = get_option("tinytable_format_bool", default = function(column)
    tools::toTitleCase(tolower(column))),
  math = get_option("tinytable_format_math", default = FALSE),
  other = get_option("tinytable_format_other", default = as.character),
  replace = get_option("tinytable_format_replace", default = TRUE),
  escape = get_option("tinytable_format_escape", default = FALSE),
  markdown = get_option("tinytable_format_markdown", default = FALSE),
  quarto = get_option("tinytable_format_quarto", default = FALSE),
  fn = get_option("tinytable_format_fn", default = NULL),
  sprintf = get_option("tinytable_format_sprintf", default = NULL),
  ...
)
```

Arguments

x	A data frame or a vector to be formatted.
i	Row indices where the formatting should be applied.
j	Column indices where the styling should be applied. Can be: • Integer vectors indicating column positions. • Character vector indicating column names. • A single string specifying a Perl-style regular expression used to match column names.
digits	Number of significant digits or decimal places.
num_fmt	The format for numeric values; one of 'significant', 'significant_cell', 'decimal', or 'scientific'.
num_zero	Logical; if TRUE, trailing zeros are kept in "decimal" format (but not in "significant" format).
num_suffix	Logical; if TRUE display short numbers with `digits` significant digits and K (thousands), M (millions), B (billions), or T (trillions) suffixes.
num_mark_big	Character to use as a thousands separator.
num_mark_dec	Decimal mark character. Default is the global option 'OutDec'.
date	A string passed to the `format()` function, such as "%Y-%m-%d". See the "Details" section in `?strptime`
bool	A function to format logical columns. Defaults to title case.
math	Logical. If TRUE, wrap cell values in math mode `$..$`. This is useful for LaTeX output or with HTML MathJax `options(tinytable_html_mathjax=TRUE)`.
other	A function to format columns of other types. Defaults to `as.character()`.
replace	Logical, String or Named list of vectors • TRUE: Replace `NA` by an empty string. • FALSE: Print `NA` as the string "NA". • String: Replace `NA` entries by the user-supplied string. • Named list: Replace matching elements of the vectors in the list by theirs names. Example: – `list("-" = c(NA, NaN), "Tiny" = -Inf, "Massive" = Inf)`
escape	Logical or "latex" or "html". If TRUE, escape special characters to display them as text in the format of the output of a `tt()` table. • If `i` and `j` are both `NULL`, escape all cells, column names, caption, notes, and spanning labels created by `group_tt()`.
markdown	Logical; if TRUE, render markdown syntax in cells. Ex: `_italicized text_` is properly italicized in HTML and LaTeX.
quarto	Logical. Enable Quarto data processing and wrap cell content in a `data-qmd` span (HTML) or `\QuartoMarkdownBase64{}` macro (LaTeX). See warnings in the Global Options section below.
fn	Function for custom formatting. Accepts a vector and returns a character vector of the same length.
sprintf	String passed to the `?sprintf` function to format numbers or interpolate strings with a user-defined pattern (similar to the `glue` package, but using Base R).
...	Additional arguments are ignored.

Value

A data frame with formatted columns.

Global options

Many global options can be used to set the default argument values of `tinytable` functions. For a full list, see:

https://vincentarelbundock.github.io/tinytable/vignettes/options.html

Quarto:

Figure environment:

- `options("tinytable_quarto_figure" = FALSE)`: Typst only. Normally, it is best to allow Quarto to define the figure environment, so the default behavior is to not include one.
- `options(tinytable_print_rstudio_notebook = "inline")`: Display tables "inline" or in the "viewer" in RStudio notebooks.

Data Processing:

The `format_tt(quarto=TRUE)` argument activates Quarto data processing for specific cells. This funcationality comes with a few warnings:

1. Currently, Quarto provides a `\QuartoMarkdownBase64{}` LaTeX macro, but it does not appear to do anything with it. References and markdown codes may not be processed as expected in LaTeX.
2. Quarto data processing can enter in conflict with `tinytable` styling or formatting options. See below for how to disable it.

`options(tinytable_quarto_disable_processing = TRUE)`

Disable Quarto processing of cell content. Setting this global option to `FALSE` may lead to conflicts with some `tinytable` features, but it also allows use of markdown and Quarto-specific code in table cells, such as cross-references.

```
x <- data.frame(Math = "x^2^", Citation = "@Lovelace1842")
fn <- function(z) sprintf("<span data-qmd='%s'></span>", z)
tt(x) |> format_tt(i = 1, fn = fn)
```

See this link for more details: https://quarto.org/docs/authoring/tables.html#disabling-quarto-table-processing

HTML:

- `options(tinytable_html_mathjax = TRUE)`
  - insert MathJax scripts in the HTML document. Warning: This may conflict with other elements of the page if MathJax is otherwise loaded.
- `options(tinytable_html_portable = TRUE)`
  - `plot_tt()` inserts base 64 encoded images directly in the HTML file rather than use external links.

PDF:

- `options(tinytable_pdf_clean = TRUE)`
  - deletes temporary and log files.
- `options(tinytable_pdf_engine = "xelatex")`
  - "xelatex", "pdflatex", "lualatex"

Examples

```
dat <- data.frame(
  a = rnorm(3, mean = 10000),
  b = rnorm(3, 10000))
tab <- tt(dat)
format_tt(tab,
 digits = 2,
 num_mark_dec = ",",
 num_mark_big = " ")

k <- tt(data.frame(x = c(0.000123456789, 12.4356789)))
format_tt(k, digits = 2, num_fmt = "significant_cell")

dat <- data.frame(
   a = c("Burger", "Halloumi", "Tofu", "Beans"),
   b = c(1.43202, 201.399, 0.146188, 0.0031),
   c = c(98938272783457, 7288839482, 29111727, 93945))
tt(dat) |>
 format_tt(j = "a", sprintf = "Food: %s") |>
 format_tt(j = 2, digits = 1, num_fmt = "decimal", num_zero = TRUE) |>
 format_tt(j = "c", digits = 2, num_suffix = TRUE)

y <- tt(data.frame(x = c(123456789.678, 12435.6789)))
format_tt(y, digits=3, num_mark_big=" ")

x <- tt(data.frame(Text = c("_italicized text_", "__bold text__")))
format_tt(x, markdown=TRUE)

tab <- data.frame(a = c(NA, 1, 2), b = c(3, NA, 5))
tt(tab) |> format_tt(replace = "-")

dat <- data.frame(
   "LaTeX" = c("Dollars $", "Percent %", "Underscore _"),
   "HTML" = c("<br>", "<sup>4</sup>", "<emph>blah</emph>")
)
tt(dat) |> format_tt(escape = TRUE)
```

`group_tt` *Spanning labels to identify groups of rows or columns*

Description

Spanning labels to identify groups of rows or columns

Usage

```
group_tt(x, i = NULL, j = NULL, indent = 1, ...)
```

Arguments

x	A data frame or data table to be rendered as a table.
i	A vector of labels with length equal to the number of rows in x, or a named list of row indices to group. The names of the list will be used as labels. The indices represent the position where labels should be inserted in the original table. For example, • i=list("Hello"=5): insert the "Hello" label after the 4th row in the original table. • i=list("Hello"=2, "World"=2): insert the two labels consecutively after the 1st row in the original table. • i=list("Foo Bar"=0): insert the label in the first row after the header.
j	A named list of column indices to group. The names of the list will be used as labels. See examples below. Note: empty labels must be a space: " ".
indent	integer number of pt to use when indenting the non-labelled rows.
...	Other arguments are ignored.

Details

Warning: The style_tt() can normally be used to style the group headers, as expected, but that feature is not available for Markdown and Word tables.

Value

An object of class tt representing the table.

Examples

```
# vector of row labels
dat <- data.frame(
    label = c("a", "a", "a", "b", "b", "c", "a", "a"),
    x1 = rnorm(8),
    x2 = rnorm(8))
tt(dat[, 2:3]) |> group_tt(i = dat$label)

# named lists of labels
tt(mtcars[1:10, 1:5]) |>
  group_tt(
    i = list(
      "Hello" = 3,
      "World" = 8),
    j = list(
      "Foo" = 2:3,
      "Bar" = 4:5))

dat <- mtcars[1:9, 1:8]
tt(dat) |>
  group_tt(i = list(
    "I like (fake) hamburgers" = 3,
    "She prefers halloumi" = 4,
```

```
    "They love tofu" = 7))

tt(dat) |>
  group_tt(
    j = list(
      "Hamburgers" = 1:3,
      "Halloumi" = 4:5,
      "Tofu" = 7))

x <- mtcars[1:5, 1:6]
tt(x) |>
  group_tt(j = list("Hello" = 1:2, "World" = 3:4, "Hello" = 5:6)) |>
  group_tt(j = list("Foo" = 1:3, "Bar" = 4:6))
```

---

plot_tt *Insert images and inline plots into tinytable objects*

Description

The plot_tt() function allows for the insertion of images and inline plots into tinytable objects. This function can handle both local and web-based images.

Usage

```
plot_tt(
  x,
  i = NULL,
  j = NULL,
  fun = NULL,
  data = NULL,
  color = "black",
  xlim = NULL,
  height = 1,
  asp = 1/3,
  images = NULL,
  assets = "tinytable_assets",
  ...
)
```

Arguments

x	A tinytable object.
i	Integer vector, the row indices where images are to be inserted. If NULL, images will be inserted in all rows.
j	Integer vector, the column indices where images are to be inserted. If NULL, images will be inserted in all columns.

fun	String or function to generate inline plots. • String: "histogram", "density", "bar", "line" • Functions that return ggplot2 objects. • Functions that return another function which generates a base R plot, ex: function(x) {function() hist(x)} • See the tutorial on the tinytable website for more information.
data	a list of data frames or vectors to be used by the plotting functions in fun.
color	string Name of color to use for inline plots (passed to the col argument base graphics plots in R).
xlim	Numeric vector of length 2.
height	Numeric, the height of the images in the table in em units.
asp	Numeric, aspect ratio of the plots (height / width).
images	Character vector, the paths to the images to be inserted. Paths are relative to the main table file or Quarto (Rmarkdown) document.
assets	Path to the directory where generated assets are stored. This path is relative to the location where a table is saved.
...	Extra arguments are passed to the function in fun. Important: Custom plotting functions must always have ... as an argument.

Details

The plot_tt() can insert images and inline plots into tables.

Value

A modified tinytable object with images or plots inserted.

---

print.tinytable *Print, display, or convert a tinytable object*

Description

This function is called automatically by R whenever a tinytable object is anprinted to the console or in an HTML viewer pane.

Usage

```
## S3 method for class 'tinytable'
print(x, output = get_option("tinytable_print_output", default = NULL), ...)
```

Arguments

x	A data frame or data table to be rendered as a table.
output	format in which a Tiny Table is printed: NULL or one of "latex", "markdown", "html", "typst", "dataframe". If NULL, the output is chosen based on these rules: • When called from a script in non-interactive mode, the default is "markdown" (interactive() == FALSE). • When called interactively in RStudio, the default is to display an HTML table in the viewer pane. • When called interactively in another development environment, the default is "markdown". • The default print output can be changed for an entire R session by calling: options(tinytable_print_output = "html") • The default print output can be changed for a single tinytable object by modifying the output S4 slot.
...	Other arguments are ignored.

Value

launch a browser window or cat() the table to console.

---

rbind2,tinytable,tinytable-method

Combine tinytable *objects by rows (vertically)*

Description

Combine tinytable objects by rows (vertically)

Usage

```
## S4 method for signature 'tinytable,tinytable'
rbind2(x, y, use_names = TRUE, headers = TRUE, ...)
```

Arguments

x	tinytable object
y	tinytable object
use_names	'TRUE' binds by matching column name, 'FALSE' by position
headers	Logical. TRUE inserts the colnames of y as an extra row between the two tables.
...	Additional arguments are ignored.

Details

format_tt() calls applied to x or y are evaluated before binding, to allow distinct formatting for each panel.

Calls to other tinytable functions such as style_tt() or group_tt() are ignored when applied to x or y. These functions should be applied to the final table instead.

Information in these S4 slots is carried over from x to the combined table:

- x@output
- x@caption
- x@width

Information in these S4 slots is concatenated and carried over to the combined table:

- c(x@notes, y@notes)

This function relies on the rbindlist() function from the data.table package.

Examples

```
library(tinytable)
x = tt(mtcars[1:3, 1:2], caption = "Combine two tiny tables.")
y = tt(mtcars[4:5, 8:10])

# rbind() does not support additional aarguments
# rbind2() supports additional arguments

# basic combination
rbind(x, y)

rbind(x, y) |> format_tt(replace = "")

# omit y header
rbind2(x, y, headers = FALSE)

# bind by position rather than column names
rbind2(x, y, use_names = FALSE)
```

---

save_tt *Save a Tiny Table to File*

Description

This function saves an object of class tinytable to a specified file and format, with an option to overwrite existing files.

Usage

```
save_tt(
  x,
  output,
  overwrite = get_option("tinytable_save_overwrite", default = FALSE)
)
```

Arguments

x	The tinytable object to be saved.
output	String or file path. • If output is "markdown", "latex", "html", "html_portable", or "typst", the table is returned in a string as an R object. • If output is a valid file path, the table is saved to file. The supported extensions are: .docx, .html, .png, .pdf, .tex, .typ, and .md (with aliases .txt, .Rmd and .qmd). • If output is "html_portable" or the global option tinytable_html_portable is TRUE, the images are included in the HTML as base64 encoded string instead of link to a local file.
overwrite	A logical value indicating whether to overwrite an existing file.

Value

A string with the table when output is a format, and the file path when output is a valid path.

Examples

```
library(tinytable)
x <- mtcars[1:4, 1:5]

fn <- file.path(tempdir(), "test.html")
tt(x) |> save_tt(fn, overwrite = TRUE)

library(tinytable)
filename <- file.path(tempdir(), "table.tex")
tt(mtcars[1:4, 1:4]) |> save_tt(filename)
```

---

style_tt *Style a Tiny Table*

Description

Style a Tiny Table

Usage

```
style_tt(
  x,
  i = NULL,
  j = NULL,
  bold = FALSE,
  italic = FALSE,
  monospace = FALSE,
  underline = FALSE,
  strikeout = FALSE,
  color = NULL,
  background = NULL,
  fontsize = NULL,
  align = NULL,
  alignv = NULL,
  colspan = NULL,
  rowspan = NULL,
  indent = NULL,
  line = NULL,
  line_color = "black",
  line_width = 0.1,
  finalize = NULL,
  tabularray_inner = NULL,
  tabularray_outer = NULL,
  bootstrap_class = NULL,
  bootstrap_css = NULL,
  bootstrap_css_rule = NULL,
  output = NULL,
  ...
)
```

Arguments

x	A table object created by `tt()`.
i	Row indices where the styling should be applied. Can be a single value, a vector, or a logical matrix with the same number of rows and columns as `x`. `i=0` is the header, and negative values are higher level headers. Row indices refer to rows *after* the insertion of row labels by `group_tt()`, when applicable.
j	Column indices where the styling should be applied. Can be: • Integer vectors indicating column positions. • Character vector indicating column names. • A single string specifying a Perl-style regular expression used to match column names.
bold	Logical; if `TRUE`, text is styled in bold.
italic	Logical; if `TRUE`, text is styled in italic.
monospace	Logical; if `TRUE`, text is styled in monospace font.

`underline` Logical; if `TRUE`, text is underlined.

`strikeout` Logical; if `TRUE`, text has a strike through line.

`color` Text color. There are several ways to specify colors, depending on the output format.

- HTML:
  - Hex code composed of # and 6 characters, ex: #CC79A7.
  - Keywords: black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua
- LaTeX:
  - Hex code composed of # and 6 characters, ex: "#CC79A7". See the section below for instructions to add in LaTeX preambles.
  - Keywords: black, blue, brown, cyan, darkgray, gray, green, lightgray, lime, magenta, olive, orange, pink, purple, red, teal, violet, white, yellow.
  - Color blending using xcolor, `ex:` white!80!blue, green!20!red‘.
  - Color names with luminance levels from the `ninecolors` package (ex: "azure4", "magenta8", "teal2", "gray1", "olive3").

`background` Background color. Specified as a color name or hexadecimal code. Can be `NULL` for default color.

`fontsize` Font size in em units. Can be `NULL` for default size.

`align` A single character or a string with a number of characters equal to the number of columns in `j`. Valid characters include 'c' (center), 'l' (left), 'r' (right), 'd' (decimal). Decimal alignment is only available in LaTeX via the `siunitx` package. The width of columns is determined by the maximum number of digits to the left and to the right in all cells specified by `i` and `j`.

`alignv` A single character specifying vertical alignment. Valid characters include 't' (top), 'm' (middle), 'b' (bottom).

`colspan` Number of columns a cell should span. `i` and `j` must be of length 1.

`rowspan` Number of rows a cell should span. `i` and `j` must be of length 1.

`indent` Text indentation in em units. Positive values only.

`line` String determines if solid lines (rules or borders) should be drawn around the cell, row, or column.

- "t": top
- "b": bottom
- "l": left
- "r": right
- Can be combined such as: "lbt" to draw borders at the left, bottom, and top.

`line_color` Color of the line. See the `color` argument for details.

`line_width` Width of the line in em units (default: 0.1).

`finalize` A function applied to the table object at the very end of table-building, for post-processing. For example, the function could use regular expressions to add LaTeX commands to the text version of the table hosted in `x@table_string`, or it could programmatically change the caption in `x@caption`.

tabularray_inner
: A string that specifies the "inner" settings of a tabularray LaTeX table.

tabularray_outer
: A string that specifies the "outer" settings of a tabularray LaTeX table.

bootstrap_class
: String. Bootstrap table class such as `"table"`, `"table table-dark"` or `"table table-dark table-hover"`. See the bootstrap documentation.

bootstrap_css
: Character vector. CSS style declarations to be applied to every cell defined by `i` and `j` (ex: `"font-weight: bold"`).

bootstrap_css_rule
: String. Complete CSS rules (with curly braces, semicolon, etc.) that apply to the table class specified by the `bootstrap_class` argument.

output
: Apply style only to the output format specified by this argument. `NULL` means that we apply to all formats.

...
: extra arguments are ignored

Details

This function applies styling to a table created by `tt()`. It allows customization of text style (bold, italic, monospace), text and background colors, font size, cell width, text alignment, column span, and indentation. The function also supports passing native instructions to LaTeX (tabularray) and HTML (bootstrap) formats.

Note: Markdown and Word tables only support these styles: italic, bold, strikeout. Moreover, the `style_tt()` function cannot be used to style headers inserted by the `group_tt()` function; instead, you should style the headers directly in the header definition using markdown syntax: `group_tt(i = list("*italic header*" = 2))`. These limitations are due to the fact that there is no markdown syntax for the other options, and that we create Word documents by converting a markdown table to .docx via the Pandoc software.

Value

An object of class `tt` representing the table.

Examples

```
if (knitr::is_html_output()) options(tinytable_print_output = "html")

library(tinytable)

tt(mtcars[1:5, 1:6])

# Alignment
tt(mtcars[1:5, 1:6]) |>
  style_tt(j = 1:5, align = "lcccr")

# Colors and styles
tt(mtcars[1:5, 1:6]) |>
  style_tt(i = 2:3, background = "black", color = "orange", bold = TRUE)
```

```
# column selection with `j``
tt(mtcars[1:5, 1:6]) |>
  style_tt(j = 5:6, background = "pink")

tt(mtcars[1:5, 1:6]) |>
  style_tt(j = "drat|wt", background = "pink")

tt(mtcars[1:5, 1:6]) |>
  style_tt(j = c("drat", "wt"), background = "pink")

tt(mtcars[1:5, 1:6], theme = "void") |>
  style_tt(
    i = 2, j = 2,
    colspan = 3,
    rowspan = 2,
    align="c",
    alignv = "m",
    color = "white",
    background = "black",
    bold = TRUE)

tt(mtcars[1:5, 1:6], theme = "void") |>
  style_tt(
    i=0:3,
    j=1:3,
    line="tblr",
    line_width=0.4,
    line_color="teal")

tt(mtcars[1:5, 1:6], theme = "bootstrap") |>
    style_tt(
      i = c(2,5),
      j = 3,
      strikeout = TRUE,
      fontsize = 0.7)

tt(mtcars[1:5, 1:6]) |>
  style_tt(bootstrap_class = "table table-dark table-hover")


inner <- "
column{1-4}={halign=c},
hlines = {fg=white},
vlines = {fg=white},
cell{1,6}{odd} = {bg=teal7},
cell{1,6}{even} = {bg=green7},
cell{2,4}{1,4} = {bg=red7},
cell{3,5}{1,4} = {bg=purple7},
cell{2}{2} = {r=4,c=2}{bg=azure7},
"
tt(mtcars[1:5, 1:4], theme = "void") |>
  style_tt(tabularray_inner = inner)
```

theme_tt *Themes for* tinytable

Description

A theme is a function which applies a collection of transformations to a tinytable object. Whereas the other tinytable functions such as format_tt() and style_tt() aim to be output-agnostic, themes can be output-specific, only applying to LaTeX, HTML, or Typst, as needed.

Each theme can have specific arguments, which are passed to the theme_tt() function. See the "Arguments" section below.

Usage

```
theme_tt(x, theme, ...)
```

Arguments

x	A tinytable object
theme	String. Name of the theme to apply. One of: • "bootstrap": Similar appearance to the default Bootstrap theme in HTML • "grid": Vertical and horizontal rules around each cell. • "multipage": Long tables continue on the next page (LaTeX only) • "placement": Position of the table environment (LaTeX) • "rotate": Rotate a LaTeX or Typst table. • "resize": Scale a LaTeX tinytable to fit the width argument. • "striped": Grey stripes on alternating rows • "tabular": Remove table environment (LaTeX) or Javascript/CSS (HTML) • "void": No rules
...	Additional arguments passed the themeing function. See the "Arguments" section below for a list of supported arguments for each theme.

Value

A modified tinytable object

Arguments

multipage

- rowhead: Non-negative integer. The number of header rows to repeat on each page.
  - Set globally with options("tinytable_theme_multipage_rowhead" = 1L)
- rowfoot: Non-negative integer. The number of footer rows to repeat on each page.
  - Set globally with options("tinytable_theme_multipage_rowfoot" = 1L)

tabular

- `style`:
  - "tabular": Drop all LaTeX dependencies and floating environments, except `\\begin{tabular}`
  - "tabularray": Drop all LaTeX dependencies and floating environments, except `\\begin{tblr}`
  - Set globally with `options("tinytable_theme_tabular_style" = "tblr")`

placement

- `horizontal` (Typst only): "l", "c", or "r" to align the table horizontally in the page.
  - Set globally with `options("tinytable_theme_placement_horizontal" = "l")`
- `latex_float`: String to insert in square brackets after the LaTeX table environment, ex: "H", "htbp". The default value is controlled by a global option:
  - Set globally with `options("tinytable_theme_placement_latex_float" = "H")`

resize

- `width`: A numeric value between 0.01 and 1, representing the proportion of the line width to use
  - Set globally with `options("tinytable_theme_resize_width" = 0.9)`
- `direction`: "down", "up", "both" A string indicating if the table should be scaled in one direction. For example, "down" will only resize the table if it exceeds `\linewidth`
  - Set globally with `options("tinytable_theme_resize_direction" = "down")`

rotate

- `angle`: Angle of the rotation. For example, 'angle=90" applies a half counter-clockwise turn.
- Caveats:
  - LaTeX and Typst only.
  - Typst: In Quarto documents, rotation does not work because Quarto takes over the figure environment.
  - LaTeX: In Quarto documents, captions must be specified using the `caption` argument in `tt()` rather than via Quarto chunk options.

Examples

```
library(tinytable)

x <- mtcars[1:4, 1:4]

# equivalent calls
tt(x, theme = "striped")

tt(x) |> theme_tt("striped")

# resize w/ argument
x <- cbind(mtcars[1:10,], mtcars[1:10,])
tt(x) |>
  theme_tt("resize", width = .9) |>
  print("latex")
```

tt *Draw a Tiny Table*

Description

The tt function renders a table in different formats with various styling options: HTML, Markdown, LaTeX, Word, PDF, PNG, or Typst. The table can be customized with additional functions:

- style_tt(): style fonts, colors, alignment, etc.
- format_tt(): format numbers, dates, strings, etc.
- group_tt(): row or column group labels.
- theme_tt(): apply a collection of transformations to a tinytable.
- save_tt(): save the table to a file or return the table as a string.
- print(): print to a specific format, ex: print(x, "latex")

tinytable attempts to determine the appropriate way to print the table based on interactive use, RStudio availability, and output format in RMarkdown or Quarto documents. Users can call print(x, output="markdown") to print the table in a specific format. Alternatively, they can set a global option: options("tinytable_print_output"="markdown")

Usage

```
tt(
  x,
  digits = get_option("tinytable_tt_digits", default = NULL),
  caption = get_option("tinytable_tt_caption", default = NULL),
  notes = get_option("tinytable_tt_notes", default = NULL),
  width = get_option("tinytable_tt_width", default = NULL),
  theme = get_option("tinytable_tt_theme", default = "default"),
  rownames = get_option("tinytable_tt_rownames", default = FALSE),
  escape = get_option("tinytable_tt_escape", default = FALSE),
  ...
)
```

Arguments

x	A data frame or data table to be rendered as a table.
digits	Number of significant digits to keep for numeric variables. When digits is an integer, tt() calls format_tt(x, digits = digits) before proceeding to draw the table. Note that this will apply all default argument values of format_tt(), such as replacing NA by "". Users who need more control can use the format_tt() function instead.
caption	A string that will be used as the caption of the table. This argument should *not* be used in Quarto or Rmarkdown documents. In that context, please use the appropriate chunk options.

notes — Notes to append to the bottom of the table. This argument accepts several different inputs:

- Single string insert a single note: `"blah blah"`
- Multiple strings insert multiple notes sequentially: `list("Hello world", "Foo bar")`
- A named list inserts a list with the name as superscript: `list("a" = list("Hello World"))`
- A named list with positions inserts markers as superscripts inside table cells: `list("a" = list(i = 0:1, j = 2, text = "Hello World"))`

width — Table or column width.

- Single numeric value smaller than or equal to 1 determines the full table width, in proportion of line width.
- Numeric vector of length equal to the number of columns in `x` determines the width of each column, in proportion of line width. If the sum of `width` exceeds 1, each element is divided by `sum(width)`. This makes the table full-width with relative column sizes.

theme — Function or string.

- String: bootstrap, grid, multipage, placement, revealjs, resize, rotate, spacing, striped, tabular, void
- Function: Applied to the `tinytable` object.

rownames — Logical. If `TRUE`, rownames are included as the first column

escape — Logical. If `TRUE`, escape special characters in the table. Equivalent to `format_tt(tt(x), escape = TRUE)`.

... — Additional arguments are ignored

Value

An object of class `tt` representing the table.

The table object has S4 slots which hold information about the structure of the table. Relying on or modifying the contents of these slots is strongly discouraged. Their names and contents could change at any time, and the `tinytable` developers do not consider changes to the internal structure of the output object to be a "breaking change" for versioning or changelog purposes.

LaTeX preamble

`tinytable` uses the `tabularray` package from your LaTeX distribution to draw tables. `tabularray`, in turn, uses the special `tblr`, `talltblr`, and `longtblr` environments.

When rendering a document from Quarto or Rmarkdown directly to PDF, `tinytable` will populate the LaTeX preamble automatically with all the required packages. For standalone LaTeX documents, these commands should be inserted in the preamble manually:

Note: Your document will fail to compile to PDF in Quarto if you enable caching and you use tinytable due to missing LaTeX headers. To avoid this problem, set the option `#| cache: false` for the chunk(s) where you use tinytable.

```
\usepackage{tabularray}
\usepackage{float}
\usepackage{graphicx}
\usepackage{rotating}
\usepackage[normalem]{ulem}
\UseTblrLibrary{booktabs}
\UseTblrLibrary{siunitx}
\newcommand{\tinytableTabularrayUnderline}[1]{\underline{#1}}
\newcommand{\tinytableTabularrayStrikeout}[1]{\sout{#1}}
\NewTableCommand{\tinytableDefineColor}[3]{\definecolor{#1}{#2}{#3}}
```

Global options

Many global options can be used to set the default argument values of `tinytable` functions. For a full list, see:

https://vincentarelbundock.github.io/tinytable/vignettes/options.html

Quarto:

Figure environment:

- `options("tinytable_quarto_figure" = FALSE)`: Typst only. Normally, it is best to allow Quarto to define the figure environment, so the default behavior is to not include one.
- `options(tinytable_print_rstudio_notebook = "inline")`: Display tables "inline" or in the "viewer" in RStudio notebooks.

Data Processing:
The `format_tt(quarto=TRUE)` argument activates Quarto data processing for specific cells. This funcationality comes with a few warnings:

1. Currently, Quarto provides a `\QuartoMarkdownBase64{}` LaTeX macro, but it does not appear to do anything with it. References and markdown codes may not be processed as expected in LaTeX.
2. Quarto data processing can enter in conflict with `tinytable` styling or formatting options. See below for how to disable it.

`options(tinytable_quarto_disable_processing = TRUE)`
Disable Quarto processing of cell content. Setting this global option to `FALSE` may lead to conflicts with some `tinytable` features, but it also allows use of markdown and Quarto-specific code in table cells, such as cross-references.

```
x <- data.frame(Math = "x^2^", Citation = "@Lovelace1842")
fn <- function(z) sprintf("<span data-qmd='%s'></span>", z)
tt(x) |> format_tt(i = 1, fn = fn)
```

See this link for more details: https://quarto.org/docs/authoring/tables.html#disabling-quarto-table-processing

HTML:

- `options(tinytable_html_mathjax = TRUE)`
  - insert MathJax scripts in the HTML document. Warning: This may conflict with other elements of the page if MathJax is otherwise loaded.
- `options(tinytable_html_portable = TRUE)`

  - `plot_tt()` inserts base 64 encoded images directly in the HTML file rather than use external links.

PDF:

- `options(tinytable_pdf_clean = TRUE)`
  - deletes temporary and log files.
- `options(tinytable_pdf_engine = "xelatex")`
  - "xelatex", "pdflatex", "lualatex"

Examples

```
library(tinytable)
x <- mtcars[1:4, 1:5]

tt(x)

tt(x,
   theme = "striped",
   width = 0.5,
   caption = "Data about cars.")

tt(x, notes = "Hello World!")

fn <- list(i = 0:1, j = 2, text = "Hello World!")
tab <- tt(x, notes = list("*" = fn))
print(tab, "latex")

k <- data.frame(x = c(0.000123456789, 12.4356789))
tt(k, digits=2)
```

Index