Package 'rtables.officer'

May 9, 2026

Title Exporting Tools for 'rtables'

Version 0.1.2

Date 2026-01-07

Description Designed to create and display complex tables with R, the 'rtables' R package allows cells in an 'rtables' object to contain any high-dimensional data structure, which can then be displayed with cell-specific formatting instructions. Additionally, the 'rtables.officer' package supports export formats related to the Microsoft Office software suite, including Microsoft Word ('docx') and Microsoft PowerPoint ('pptx').

License Apache License 2.0

URL https://github.com/insightsengineering/rtables.officer, https://insightsengineering.github.io/rtables.officer/

BugReports https://github.com/insightsengineering/rtables.officer/issues

Depends formatters (>= 0.5.12), R (>= 4.4.0), rlistings (>= 0.2.13), rtables (>= 0.6.15)

Imports checkmate (>= 2.1.0), flextable (>= 0.9.10), lifecycle (>= 0.2.0), officer (>= 0.7.1), stats, stringi (>= 1.6), systemfonts

Suggests car (>= 3.0-13), dplyr (>= 1.0.5), knitr (>= 1.42), rmarkdown (>= 2.23), tern (>= 0.9.10), testthat (>= 3.3.0), tibble (>= 3.2.1), tidyr (>= 1.1.3), withr (>= 2.0.0), xml2 (>= 1.1.0)

VignetteBuilder knitr, rmarkdown

Config/Needs/verdepcheck insightsengineering/formatters, insightsengineering/rtables, mllg/checkmate, rstudio/htmltools, gagolews/stringi, cran/car, tidyverse/dplyr, davidgohel/flextable, yihui/knitr, r-lib/lifecycle, davidgohel/officer, Merck/r2rtf, rstudio/rmarkdown, therneau/survival, r-lib/testthat, tidyverse/tibble, tidyverse/tidyr, r-lib/withr, r-lib/xml2

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Collate 'add_flextable_separators.R' 'as_flextable.R'
'export_as_docx.R' 'package.R' 'theme_defaults.R'

NeedsCompilation no

Author Gabriel Becker [ctb],
Davide Garolini [aut] (ORCID: <https://orcid.org/0000-0002-1445-1369>),
Emily de la Rua [aut] (ORCID: <https://orcid.org/0009-0000-8738-5561>),
Abinaya Yogasekaram [aut] (ORCID:
<https://orcid.org/0009-0005-2083-1105>),
Joe Zhu [aut, cre] (ORCID: <https://orcid.org/0000-0001-7566-2787>),
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Joe Zhu <joe.zhu@roche.com>

Repository CRAN

Date/Publication 2026-01-08 08:40:48 UTC

Contents

add_flextable_separators 2
export_as_docx 3
tt_to_flextable 6

Index **13**

`add_flextable_separators` *Add Conditional Separators (horizontal line or padding) to* `flextable` *Rows*

Description

Modifies an existing `flextable` object by adding visual separators (horizontal lines or bottom padding) after specific rows based on a control vector, without changing the table's row count.

Usage

```
add_flextable_separators(
  ft,
  trailing_sep,
  border = officer::fp_border(width = 1, color = "grey60"),
  padding = 10
)
```

Arguments

ft	A flextable object.
trailing_sep	A vector specifying separators. Its length must equal the number of rows in the body of ft. Allowed values are NA (no separator), "-" (adds a horizontal line), or " " (adds bottom padding).
border	The fp_border object to use for horizontal lines when trailing_sep is "-". Defaults to a gray line of width 1.
padding	The amount of bottom padding (in points) to add when trailing_sep is " ". Defaults to 10.

Value

The modified flextable object, or the original ft if all trailing_sep values are NA. Throws an error for invalid inputs or invalid characters in trailing_sep.

Examples

```
content <- data.frame(
  USUBJID = c("S1", "S1", "S1", "S2", "S2", "S2", "S3"),
  ARM = c("A", "A", "B", "A", "A", "B", "A"),
  VAL = round(rnorm(7), 2)
)
ft <- flextable::as_flextable(content)
ft <- flextable::theme_booktabs(ft)

# Define separators: line, space, NA, line, space, NA, NA
sep_ctrl <- c("-", " ", NA, "-", " ", NA, NA)

ft_modified <- add_flextable_separators(ft, sep_ctrl)
print(ft_modified)

# Example: All NA - should return original ft
ft_all_na <- add_flextable_separators(ft, rep(NA, 7))
identical(ft, ft_all_na) # Should be TRUE

# Example: Invalid character - should throw error
tryCatch(
  add_flextable_separators(ft, c("-", "x", NA, "-", " ", NA, NA)),
  error = function(e) print(e)
)
```

---

export_as_docx *Export to a Word document*

Description

From an rtables table, produce a self-contained Word document or attach it to a template Word file (template_file). This function is based on the tt_to_flextable() transformer and the officer package.

Usage

```
export_as_docx(
  tt,
  file,
  add_page_break = FALSE,
  add_template_page_numbers = TRUE,
  titles_as_header = TRUE,
  integrate_footers = TRUE,
  section_properties = section_properties_default(),
  doc_metadata = NULL,
  template_file = NULL,
  ...
)

section_properties_default(
  page_size = c("letter", "A4"),
  orientation = c("portrait", "landscape")
)

margins_potrait()

margins_landscape()
```

Arguments

tt	(TableTree or related class) a TableTree object representing a populated table.
file	(string) output file. Must have .docx extension.
add_page_break	(flag) whether to add a page break after the table (TRUE) or not (FALSE).
add_template_page_numbers	(flag) whether to add page numbers to the word document as page footer. This uses templates to achieve it. Defaults to TRUE. Consider adding your own template file if you want more customization.
titles_as_header	(flag) Controls how titles are rendered relative to the table. If TRUE (default), the main title (formatters::main_title()) and subtitles (formatters::subtitles()) are added as distinct header rows within the flextable object itself. If FALSE,

titles are rendered as a separate paragraph of text placed immediately before the table.

`integrate_footers`
: (`flag`)
Controls how footers are rendered relative to the table. If `TRUE` (default), footers (e.g., `formatters::main_footer()`, `formatters::prov_footer()`) are integrated directly into the `flextable` object, typically appearing as footnotes below the table body with a smaller font. If `FALSE`, footers are rendered as a separate paragraph of text placed immediately after the table.

`section_properties`
: (`officer::prop_section`)
an `officer::prop_section()` object which sets margins and page size. Defaults to `section_properties_default()`.

`doc_metadata`
: (`list` of `string`)
any value that can be used as metadata by `officer::set_doc_properties()`. Important text values are `title`, `subject`, `creator`, and `description`, while `created` is a date object.

`template_file`
: (`string`)
template file that `officer` will use as a starting point for the final document. Document attaches the table and uses the defaults defined in the template file.

`...`
: (`any`)
additional arguments passed to `tt_to_flextable()`.

`page_size`
: (`string`) page size. Can be `"letter"` or `"A4"`. Defaults to `"letter"`.

`orientation`
: (`string`) page orientation. Can be `"portrait"` or `"landscape"`. Defaults to `"portrait"`.

Details

Pagination Behavior for Titles and Footers (this behavior is experimental at the moment):

The rendering of titles and footers interacts with table pagination as follows:

- **Titles:** When `titles_as_header = TRUE` (default), the integrated title header rows typically repeat at the top of each new page if the table spans multiple pages. Setting `titles_as_header = FALSE` renders titles as a separate paragraph only once before the table begins.
- **Footers:** Regardless of the `integrate_footers` setting, footers appear only once. Integrated footnotes (`integrate_footers = TRUE`) appear at the very end of the complete table, and separate text paragraphs (`integrate_footers = FALSE`) appear after the complete table. Footers do not repeat on each page.

Value

No return value, called for side effects

Functions

- `section_properties_default()`: Helper function that defines standard portrait properties for tables.

- `margins_potrait()`: Helper function that defines standard portrait margins for tables.
- `margins_landscape()`: Helper function that defines standard landscape margins for tables.

Note

`export_as_docx()` has few customization options available. If you require specific formats and details, we suggest that you use `tt_to_flextable()` prior to `export_as_docx()`. If the table is modified first using `tt_to_flextable()`, the `titles_as_header` and `integrate_footers` parameters must be re-specified.

See Also

`tt_to_flextable()`

Examples

```
lyt <- basic_table() %>%
  split_cols_by("ARM") %>%
  analyze(c("AGE", "BMRKR2", "COUNTRY"))

tbl <- build_table(lyt, ex_adsl)

# See how the section_properties_portrait() function is built for customization
tf <- tempfile(tmpdir = tempdir(check = TRUE), fileext = ".docx")
export_as_docx(tbl,
  file = tf,
  section_properties = section_properties_default(orientation = "landscape")
)
```

---

tt_to_flextable *Create a* flextable *from an* rtables *table*

Description

Principally used within `export_as_docx()`, this function produces a `flextable` from an `rtables` table. If `theme = theme_docx_default()` (default), a `.docx`-friendly table will be produced. If `theme = NULL`, the table will be produced in an `rtables`-like style.

Usage

```
tt_to_flextable(
  tt,
  theme = theme_docx_default(),
  border = flextable::fp_border_default(width = 0.5),
  indent_size = NULL,
  titles_as_header = TRUE,
  bold_titles = TRUE,
```

```
  integrate_footers = TRUE,
  counts_in_newline = FALSE,
  paginate = FALSE,
  fontspec = NULL,
  lpp = NULL,
  cpp = NULL,
  ...,
  colwidths = NULL,
  tf_wrap = !is.null(cpp),
  max_width = cpp,
  total_page_height = 10,
  total_page_width = 10,
  autofit_to_page = TRUE
)

theme_docx_default(
  font = "Arial",
  font_size = 9,
  cell_margins = c(word_mm_to_pt(1.9), word_mm_to_pt(1.9), 0, 0),
  bold = c("header", "content_rows", "label_rows", "top_left"),
  bold_manual = NULL,
  border = flextable::fp_border_default(width = 0.5)
)

theme_html_default(
  font = "Courier",
  font_size = 9,
  cell_margins = 0.2,
  remove_internal_borders = "label_rows",
  border = flextable::fp_border_default(width = 1, color = "black")
)

word_mm_to_pt(mm)
```

Arguments

`tt`	(`TableTree`, `listing_df`, or related class) a `TableTree` or `listing_df` object representing a populated table or listing.
`theme`	(`function` or `NULL`) a theme function designed to change the layout and style of a `flextable` object. Defaults to `theme_docx_default()`, the classic Microsoft Word output style. If `NULL`, a table with style similar to the `rtables` default will be produced. See Details below for more information.
`border`	(`flextable::fp_border()`) border style. Defaults to `flextable::fp_border_default(width = 0.5)`.
`indent_size`	(`numeric(1)`) indentation size. If `NULL`, the default indent size of the table (see `formatters::matrix_form()`

indent_size, default is 2) is used. To work with docx, any size is multiplied by 1 mm (2.83 pt) by default.

titles_as_header
: (flag)
Controls how titles are rendered relative to the table. If TRUE (default), the main title (formatters::main_title()) and subtitles (formatters::subtitles()) are added as distinct header rows within the flextable object itself. If FALSE, titles are rendered as a separate paragraph of text placed immediately before the table.

bold_titles
: (flag or integer)
whether titles should be bold (defaults to TRUE). If one or more integers are provided, these integers are used as indices for lines at which titles should be bold.

integrate_footers
: (flag)
Controls how footers are rendered relative to the table. If TRUE (default), footers (e.g., formatters::main_footer(), formatters::prov_footer()) are integrated directly into the flextable object, typically appearing as footnotes below the table body with a smaller font. If FALSE, footers are rendered as a separate paragraph of text placed immediately after the table.

counts_in_newline
: (flag)
whether column counts should be printed on a new line. In rtables, column counts (i.e. (N=xx)) are always printed on a new line (TRUE). For docx exports it may be preferred to print these counts on the same line (FALSE). Defaults to FALSE.

paginate
: (flag)
whether the rtables pagination mechanism should be used. If TRUE, this option splits tt into multiple flextables as different "pages". When using export_as_docx() we suggest setting this to FALSE and relying only on the default Microsoft Word pagination system as co-operation between the two mechanisms is not guaranteed. Defaults to FALSE.

fontspec
: (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

lpp
: (numeric(1))
maximum lines per page including (re)printed header and context rows.

cpp
: (numeric(1) or NULL)
width (in characters) of the pages for horizontal pagination. NA (the default) indicates cpp should be inferred from the page size; NULL indicates no horizontal pagination should be done regardless of page size.

...
: (any)
additional parameters to be passed to the pagination function. See rtables::paginate_table() for options. If paginate = FALSE this argument is ignored.

colwidths
: (numeric)
column widths for the resulting flextable(s). If NULL, the column widths estimated with formatters::propose_column_widths() will be used. When

exporting into .docx these values are normalized to represent a fraction of the total_page_width. If these are specified, autofit_to_page is set to FALSE.

tf_wrap (flag)
whether the text for title, subtitles, and footnotes should be wrapped.

max_width (integer(1), string or NULL)
width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")). If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.

total_page_height
(numeric(1))
total page height (in inches) for the resulting flextable(s). Used only to estimate number of lines per page (lpp) when paginate = TRUE. Defaults to 10.

total_page_width
(numeric(1))
total page width (in inches) for the resulting flextable(s). Any values added for column widths are normalized by the total page width. Defaults to 10. If autofit_to_page = TRUE, this value is automatically set to the allowed page width.

autofit_to_page
(flag)
whether column widths should be automatically adjusted to fit the total page width. If FALSE, colwidths is used to indicate proportions of total_page_width. Defaults to TRUE. See flextable::set_table_properties(layout) for more details.

font (string)
font. Defaults to "Arial". If the font given is not available, the flextable default is used instead. For options, consult the family column from systemfonts::system_fonts().

font_size (integer(1))
font size. Defaults to 9.

cell_margins (numeric(1) or numeric(4))
a numeric or a vector of four numbers indicating c("left", "right", "top", "bottom"). It defaults to 0 for top and bottom, and to 0.19 mm in Word pt for left and right.

bold (character)
parts of the table text that should be in bold. Can be any combination of c("header", "content_rows", "label_rows", "top_left"). The first one renders all column names bold (not topleft content). The second and third option use formatters::make_row_df() to render content or/and label rows as bold.

bold_manual (named list or NULL)
list of index lists. See example for needed structure. Accepted groupings/names are c("header", "body").

remove_internal_borders
(character)
where to remove internal borders between rows. Defaults to "label_rows". Currently there are no other options and this can be turned off by providing any other character value.

mm (numeric(1))
the value in mm to transform to pt.

Details

If you would like to make a minor change to a pre-existing style, this can be done by extending themes. You can do this by either adding your own theme to the theme call (e.g. `theme = c(theme_docx_default(), my_theme)`) or creating a new theme as shown in the examples below. Please pay close attention to the parameters' inputs.

It is possible to use some hidden values to build your own theme (hence the need for the `...` parameter). In particular, `tt_to_flextable()` uses the following variable: `tbl_row_class = rtables::make_row_df(tt)$node_class`. This is ignored if not used in the theme. See `theme_docx_default()` for an example on how to retrieve and use these values.

Value

A `flextable` object.

Functions

- `theme_docx_default()`: Main theme function for `export_as_docx()`.
- `theme_html_default()`: Theme function for html outputs.
- `word_mm_to_pt()`: Padding helper functions to transform mm to pt.

Note

Currently `cpp`, `tf_wrap`, and `max_width` are only used in pagination and should be used cautiously if used in combination with `colwidths` and `autofit_to_page`. If issues arise, please raise an issue on GitHub or communicate this to the package maintainers directly.

See Also

`export_as_docx()`

Examples

```
analysisfun <- function(x, ...) {
  in_rows(
    row1 = 5,
    row2 = c(1, 2),
    .row_footnotes = list(row1 = "row 1 - row footnote"),
    .cell_footnotes = list(row2 = "row 2 - cell footnote")
  )
}

lyt <- basic_table(
  title = "Title says Whaaaat", subtitles = "Oh, ok.",
  main_footer = "ha HA! Footer!"
) |>
  split_cols_by("ARM") |>
```

```
  analyze("AGE", afun = analysisfun)

tbl <- build_table(lyt, ex_adsl)

# Example 1: rtables style ------------------------------------------------
tt_to_flextable(tbl, theme = NULL)

# Example 2: docx style ---------------------------------------------------
tt_to_flextable(tbl, theme = theme_docx_default(font_size = 6))

# Example 3: Extending the docx theme -------------------------------------
my_theme <- function(x, ...) {
 flextable::border_inner(x, part = "body", border = flextable::fp_border_default(width = 0.5))
}
flx <- tt_to_flextable(tbl, theme = c(theme_docx_default(), my_theme))

# Example 4: Creating a custom theme --------------------------------------
special_bold <- list(
  "header" = list("i" = 1, "j" = c(1, 3)),
  "body" = list("i" = c(1, 2), "j" = 1)
)
custom_theme <- theme_docx_default(
  font_size = 10,
  font = "Brush Script MT",
  border = flextable::fp_border_default(color = "pink", width = 2),
  bold = NULL,
  bold_manual = special_bold
)
tt_to_flextable(tbl,
  border = flextable::fp_border_default(color = "pink", width = 2),
  theme = custom_theme
)

# Example 5: Extending the docx theme -------------------------------------
my_theme <- function(font_size = 6) { # here can pass additional arguments for default theme
  function(flx, ...) {
    # First apply theme_docx_default
    flx <- theme_docx_default(font_size = font_size)(flx, ...)

    # Then apply additional styling
    flx <- flextable::border_inner(flx,
      part = "body",
      border = flextable::fp_border_default(width = 0.5)
    )

    return(flx)
  }
}
flx <- tt_to_flextable(tbl, theme = my_theme())

# html theme

# Define a layout for the table
```

```
lyt <- basic_table() |>
  # Split columns by the "ARM" variable
  split_cols_by("ARM") |>
  # Analyze the "AGE", "BMRKR2", and "COUNTRY" variables
  analyze(c("AGE", "BMRKR2", "COUNTRY"))

# Build the table using the defined layout and example data 'ex_adsl'
tbl <- build_table(lyt, ex_adsl)

# Convert the table to a flextable object suitable for HTML,
# applying the default HTML theme and setting the orientation to landscape
tbl_html <- tt_to_flextable(
  tbl,
  theme = theme_html_default(),
  section_properties = section_properties_default(orientation = "landscape")
)

# Save the flextable as an HTML file named "test.html"
flextable::save_as_html(tbl_html,
  path = tempfile(tmpdir = tempdir(check = TRUE), fileext = ".html")
)
```

Index

add_flextable_separators, 2

export_as_docx, 3
export_as_docx(), *6*, *8*, *10*

font_spec(), *8*
formatters::main_footer(), *5*, *8*
formatters::main_title(), *4*, *8*
formatters::make_row_df(), *9*
formatters::matrix_form(), *7*
formatters::propose_column_widths(), *8*
formatters::prov_footer(), *5*, *8*
formatters::subtitles(), *4*, *8*

margins_landscape *(*export_as_docx*)*, 3
margins_potrait *(*export_as_docx*)*, 3

officer::prop_section(), *5*
officer::set_doc_properties(), *5*

rtables::paginate_table(), *8*

section_properties_default
        *(*export_as_docx*)*, 3
section_properties_default(), *5*

theme_docx_default *(*tt_to_flextable*)*, 6
theme_docx_default(), *10*
theme_html_default *(*tt_to_flextable*)*, 6
tt_to_flextable, 6
tt_to_flextable(), *4–6*, *10*

word_mm_to_pt *(*tt_to_flextable*)*, 6