Package: rtables.officer (via r-universe)

January 17, 2025

Title Exporting Tools for 'rtables'

Version 0.0.2

Date 2025-01-14

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 (>= 0.5.10), magrittr (>= 1.5), methods, R (>= 2.10), rtables (>= 0.6.11)

Imports checkmate (>= 2.1.0), flextable (>= 0.9.6), lifecycle (>= 0.2.0), officer (>= 0.6.6), stats, stringi (>= 1.6)

Suggests broom (>= 1.0.5), car (>= 3.0-13), dplyr (>= 1.0.5), knitr (>= 1.42), r2rtf (>= 0.3.2), rmarkdown (>= 2.23), survival (>= 3.3-1), testthat (>= 3.0.4), 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, tidyverse/magrittr, mllg/checkmate, rstudio/htmltools, gagolews/stringi, tidymodels/broom, 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 7.3.2

Collate 'package.R' 'export_as_docx.R' 'as_flextable.R'

NeedsCompilation no

Author Gabriel Becker [ctb], Davide Garolini [aut], Emily de la Rua [aut], Abinaya Yogasekaram [aut], Joe Zhu [aut, cre] (<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 2025-01-17 10:50:01 UTC

Config/pak/sysreqs libcairo2-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev

Contents

`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,
  doc_metadata = NULL,
  titles_as_header = FALSE,
  footers_as_text = TRUE,
  template_file = NULL,
  section_properties = section_properties_default(),
  ...
```

```
)

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.
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.
titles_as_header	(flag) whether the table should be self-contained with additional header rows created for titles and subtitles (TRUE), or titles and subtitles should be added as a paragraph of text above the table (FALSE). Defaults to FALSE.
footers_as_text	(flag) whether footers should be added as a new paragraph after the table (TRUE) or the table should be self-contained, implementing flextable-style footnotes (FALSE) with the same style but a smaller font. Defaults to TRUE.
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.
section_properties	(officer::prop_section) an officer::prop_section() object which sets margins and page size. Defaults to section_properties_default().
...	(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".

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 `footer_as_text` 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(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,
  footers_as_text = FALSE,
  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 or related class) a TableTree object representing a populated table.
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)
whether the table should be self-contained and additional header rows created for the formatters::main_title() string and formatters::subtitles() character vector (one row per element). Defaults to TRUE. If FALSE, titles and subtitles are added as a paragraph of text above 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.

footers_as_text
(flag)
whether footers should be added as a new paragraph after the table (TRUE) or the table should be self-contained, implementing flextable-style footnotes (FALSE) with the same style but a smaller font. Defaults to FALSE.

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())
```

Index