Package: rtables.officer (via r-universe)

November 12, 2024

Title Reporting Tables

Version 0.0.1.9003

Date 2024-11-01

Description Rtables to officer

License Apache License 2.0 | file LICENSE

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

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

Depends formatters (>= 0.5.9), magrittr (>= 1.5), methods, R (>= 2.10), rtables (>= 0.6.9)

Imports checkmate (>= 2.1.0), htmltools (>= 0.5.4), lifecycle (>= 0.2.0), stats, stringi (>= 1.6)

Suggests broom (>= 1.0.6), car (>= 3.0-13), dplyr (>= 1.0.5), flextable (>= 0.9.6), knitr (>= 1.42), officer (>= 0.6.6), 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

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Collate 'package.R' 'tt_export.R'

Config/pak/sysreqs libicu-dev

Repository https://insightsengineering.r-universe.dev

RemoteUrl https://github.com/insightsengineering/rtables.officer

RemoteRef main

RemoteSha 7c4503d4182c7d063eff4f9f1aa030aba36bfb9d

Contents

`export_as_docx` *Export as word document*

Description

From a 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

`file`	(`string`) string that indicates the final file output. Must have `.docx` extension.
`doc_metadata`	(`list of strings`) 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.
`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`	(`character(1)`) page size. Can be `"letter"` or `"A4"`. Defaults to `"letter"`.
`orientation`	(`character(1)`) page orientation. Can be `"portrait"` or `"landscape"`. Defaults to `"portrait"`.

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`. Only the `title_as_header` and `footer_as_text` parameters must be re-specified if the table is changed first using `tt_to_flextable()`.

See Also

`tt_to_flextable()`

Examples

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

tbl <- build_table(lyt, ex_adsl)

# See how section_properties_portrait function is built for custom
```

```
tf <- tempfile(fileext = ".docx")
export_as_docx(tbl,
  file = tf,
  section_properties = section_properties_default(orientation = "landscape")
)
```

`export_as_tsv` *Create enriched flat value table with paths*

Description

This function creates a flat tabular file of cell values and corresponding paths via `path_enriched_df()`. It then writes that data frame out as a `tsv` file.

Usage

```
export_as_tsv(
  tt,
  file = NULL,
  path_fun = collapse_path,
  value_fun = collapse_values,
  sep = "\t",
  ...
)

import_from_tsv(file)
```

Arguments

`file`	(`string`) the path of the file to written to or read from.
`sep`	(`string`) defaults to `\t`. See `utils::write.table()` for more details.
`...`	(`any`) additional arguments to be passed to `utils::write.table()`.

Details

By default (i.e. when `value_func` is not specified, list columns where at least one value has length > 1 are collapsed to character vectors by collapsing the list element with `"|"`.

Value

- `export_as_tsv` returns `NULL` silently.
- `import_from_tsv` returns a `data.frame` with re-constituted list values.

Note

There is currently no round-trip capability for this type of export. You can read values exported this way back in via `import_from_tsv` but you will receive only the `data.frame` version back, NOT a `TableTree`.

See Also

`path_enriched_df()` for the underlying function that does the work.

Index