Package: tablespan (via r-universe)

December 21, 2024

Type Package

Title Create Satisficing 'Excel', 'HTML', 'LaTeX', and 'RTF' Tables using a Simple Formula

Version 0.1.8

Maintainer Jannik H. Orzek <jannik.orzek@mailbox.org>

Description Create ``good enough'' tables with a single formula. 'tablespan' tables can be exported to 'Excel', 'HTML', 'LaTeX', and 'RTF' by leveraging the packages 'openxlsx' and 'gt'. See <https://jhorzek.github.io/tablespan/> for an introduction.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, gt, methods, openxlsx, rlang, utils

URL https://github.com/jhorzek/tablespan, https://jhorzek.github.io/tablespan/

BugReports https://github.com/jhorzek/tablespan/issues

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libnode-dev

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

RemoteUrl https://github.com/jhorzek/tablespan

RemoteRef HEAD

RemoteSha 316375a1bc24b9c7279489863ddae686afa0b017

Contents

`as_excel` *as_excel*

Description

Write a tablespan table to an excel workbook.

Usage

```
as_excel(
  tbl,
  workbook = openxlsx::createWorkbook(),
  sheet = "Table",
  start_row = 1,
  start_col = 1,
  styles = tbl_styles()
)
```

Arguments

`tbl`	table created with tablespan::tablespan
`workbook`	Excel workbook created with openxlsx::createWorkbook()
`sheet`	name of the sheet to which the table should be written to
`start_row`	row at which to start the table
`start_col`	column at which to start the table
`styles`	openxlsx style for the different table elements (see ?tablespan::tbl_styles). The styles element also allows applying custom styles to parts of the data shown in the table body.

Value

openxlsx workbook object that can be edited and saved with openxlsx

Examples

```
library(tablespan)
library(dplyr)
data("iris")

tbl <- tablespan(data = iris[iris$Species == "setosa", ],
          formula = Species ~ (Sepal = Sepal.Length + Sepal.Width) +
            (Petal = (Width = Petal.Length) + Petal.Width))

wb <- as_excel(tbl = tbl)

# saveWorkbook(wb, "iris.xlsx")

# To apply a custom style to some elements use the styles argument. The following
# applies the "bold" style to the rows 1-5 of the Sepal.Length column and
# the rows 9-10 of the Petal.Width column.
bold <- openxlsx::createStyle(textDecoration = "bold")

wb <- as_excel(tbl = tbl,
               styles = tbl_styles(cell_styles = list(cell_style(rows = 1:5,
                                                                 colnames = "Sepal.Length",
                                                                   style = bold),
                                                      cell_style(rows = 9:10,
                                                                 colnames = "Petal.Width",
                                                                   style = bold))))
# saveWorkbook(wb, "iris.xlsx")

# The main use case for tablespan is when you already have a summarized table
# that you now want to share using xlsx. The following shows an example using
# the dplyr package:

# First summarize the data:
summarized_table <- mtcars |>
  group_by(cyl, vs) |>
  summarise(N = n(),
            mean_hp = mean(hp),
            sd_hp = sd(hp),
            mean_wt = mean(wt),
            sd_wt = sd(wt))

# Now, we want to create a table, where we show the grouping variables
# as row names and also create spanners for the horse power (hp) and the
# weight (wt) variables:
tbl <- tablespan(data = summarized_table,
          formula = Cylinder:cyl + Engine:vs ~
            N +
            (`Horse Power` = Mean:mean_hp + SD:sd_hp) +
            (`Weight` = Mean:mean_wt + SD:sd_wt),
          title = "Motor Trend Car Road Tests",
          subtitle = "A table created with tablespan",
          footnote = "Data from the infamous mtcars data set.")
```

```
wb <- as_excel(tbl = tbl)

# Create the excel table:
# openxlsx::saveWorkbook(wb,
#                        file = "cars.xlsx", overwrite = TRUE)
```

---

as_gt *as_gt*

Description

Translates a table created with tablespan to a great table (gt). See <https://gt.rstudio.com/>.

Usage

```
as_gt(
  tbl,
  groupname_col = NULL,
  separator_style = gt::cell_borders(sides = c("right"), weight = gt::px(1), color =
    "gray"),
  auto_format = TRUE,
  ...
)
```

Arguments

tbl	table created with tablespan::tablespan
groupname_col	Provide column names to group data. See ?gt::gt for more details.
separator_style	style of the vertical line that separates the row names from the data.
auto_format	should the table be formatted automatically?
...	additional arguments passed to gt::gt().

Details

Tablespan itself does not provide any printing of tables as HTML table. However, with as_gt, tablespan can be translated to a great table which provides html and LaTeX output.

Value

gt table that can be further adapted with the gt package.

Examples

```
library(tablespan)
library(dplyr)
data("mtcars")

summarized_table <- mtcars |>
  group_by(cyl, vs) |>
  summarise(N = n(),
            mean_hp = mean(hp),
            sd_hp = sd(hp),
            mean_wt = mean(wt),
            sd_wt = sd(wt))

tbl <- tablespan(data = summarized_table,
                 formula = (LHS = Cylinder:cyl + Engine:vs) ~
                   N +
                   (Results = (`Horse Power` = Mean:mean_hp + SD:sd_hp) +
                      (`Weight` = Mean:mean_wt + SD:sd_wt)))

gt_tbl <- as_gt(tbl)
gt_tbl
```

`cell_style` *cell_style*

Description

cell_style

Usage

```
cell_style(rows, colnames, style, gridExpand = TRUE, stack = TRUE)
```

Arguments

`rows`	indices of the rows to which the style should be applied
`colnames`	names of the columns to which the style should be applied
`style`	style created with openxlsx::createStyle() that will be applied to the selected cells
`gridExpand`	see ?openxlsx::addStyle: Apply style only to the selected elements (set gridExpand = FALSE) or to all combinations?
`stack`	should the style be added to existing styles (TRUE) or overwrite existing styles (FALSE)

Value

list with specified styles

Examples

```
library(tablespan)
data("iris")

tbl <- tablespan(data = iris[iris$Species == "setosa", ],
          formula = Species ~ (Sepal = Sepal.Length + Sepal.Width) +
            (Petal = (Width = Petal.Length) + Petal.Width))


# To apply a custom style to some elements use the styles argument. The following
# applies the "bold" style to the rows 1-5 of the Sepal.Length column and
# the rows 9-10 of the Petal.Width column.
bold <- openxlsx::createStyle(textDecoration = "bold")

wb <- as_excel(tbl = tbl,
               styles = tbl_styles(cell_styles = list(cell_style(rows = 1:5,
                                                                 colnames = "Sepal.Length",
                                                                   style = bold),
                                                      cell_style(rows = 9:10,
                                                                 colnames = "Petal.Width",
                                                                   style = bold))))
# saveWorkbook(wb, "iris.xlsx")
```

`create_data_styles` *create_data_styles*

Description

This function sets some defaults for data_styles. See ?tbl_styles

Usage

```
create_data_styles(
  double = list(test = is.double, style = openxlsx::createStyle(numFmt = "0.00")),
  integer = list(test = is.integer, style = openxlsx::createStyle(numFmt = "0")),
  ...
)
```

Arguments

`double`	style for columns of type double
`integer`	style for columns of type integer
`...`	add further styles

Details

Styles are applied to the columns in the data set based on their classes (e.g., numeric, character, etc.). data_styles must be a list of lists. Each inner list must have two elements: a "test" that is used to determine the class of a data colum (e.g., is.double) and a style that is then applied to the columns where the test returns TRUE. Note that styles will be applied in the order of the list, meaning that a later style may overwrite an earlier style.

Value

a list of lists with styles

Examples

```
library(tablespan)
# Make all booleans bold:
create_data_styles(boolean = list(test = is.logical,
                   style = openxlsx::createStyle(textDecoration = "bold")))
```

---

print.Tablespan *print.Tablespan*

Description

print.Tablespan

Usage

```
## S3 method for class 'Tablespan'
print(x, digits = 2, n = 3, ...)
```

Arguments

x	result from tablespan
digits	number of digits to round doubles to
n	number of rows to print
...	additional arguments passed to prmatrix

Value

nothing

Examples

```
data("iris")
tbl <- tablespan(data = iris[iris$Species == "setosa", ],
          formula = Species ~ (Sepal = Sepal.Length + Sepal.Width) +
            (Petal = Petal.Length + Petal.Width))
print(tbl)
```

tablespan *tablespan*

Description

Create complex table spanners with a simple formula.

Usage

```
tablespan(data, formula, title = NULL, subtitle = NULL, footnote = NULL)
```

Arguments

data	data set
formula	formula to create table
title	string specifying the title of the table
subtitle	string specifying the subtitle of the table
footnote	string specifying the footnote of the table

Details

tablespan provides a formula based approach to adding headers and spanners to an existing data.frame. The objective is to provide a unified, easy to use, but good enough approach to building and exporting tables to Excel, HTML, and LaTeX. To this end, tablespan leverages the awesome packages openxlsx and gt.

Following the tibble approach, tablespan assumes that all items that you may want to use as row names are just columns in your data set (see example). That is, tablespan will allow you to pick some of your items as row names and then just write them in a separate section to the left of the data.

The table headers are defined with a basic formula approach inspired by tables. For example, Species ~ Sepal.Length + Sepal.Width defines a table with Species as the row names and Sepal.Length and Sepal.Width as columns. The output will be similar to the following:

```
|Species | Sepal.Length  Sepal.Width|
|:-------|------------: -----------:|
|setosa  |          5.1          3.5|
|setosa  |          4.9          3.0|
```

Note that the row names (Species) are in a separate block to the left.

You can add spanner labels with as follows:

Species ~ (Sepal = Sepal.Length + Sepal.Width) + (Petal = Sepal.Length + Sepal.Width)

This will result in an output similar to:

```
|        |           Sepal          |          Petal           |
|Species | Sepal.Length| Sepal.Width| Petal.Length| Petal.Width|
|:-------|------------:|-----------:|------------:|-----------:|
|setosa  |          5.1|         3.5|          1.4|         0.2|
```

You can also nest spanners (e.g., `Species ~ (Sepal = (Length = Sepal.Length) + (Width = Sepal.Width))`.

When exporting tables, you may want to rename some of you columns. For example, you may want to rename Sepal.Length and Petal.Length to Length and Sepal.Width and Petal.Width to Width. With `tablespan`, you can rename the item in the header using `new_name:old_name`. For example, `Species ~ (Sepal = Length:Sepal.Length + Width:Sepal.Width) + (Petal = Length:Sepal.Length + Width:Sepal.Width)` defines a table similar to the following:

```
|        |      Sepal     |      Petal     |
|Species | Length | Width | Length | Width |
|:-------|-------:|------:|-------:|------:|
|setosa  |     5.1|    3.5|     1.4|    0.2|
```

Finally, to create a table without row names, use `1 ~ (Sepal = Length:Sepal.Length + Width:Sepal.Width) + (Petal = Length:Sepal.Length + Width:Sepal.Width)` This defines as table similar to the following:

```
|      Sepal     |      Petal     |
| Length | Width | Length | Width |
|-------:|------:|-------:|------:|
|     5.1|    3.5|     1.4|    0.2|
```

Tables created with `tablespan` can be exported to Excel (using `openxlsx`), HTML (using `gt`), LaTeX (using `gt`), and RTF (using `gt`).

References:

- gt: Iannone R, Cheng J, Schloerke B, Hughes E, Lauer A, Seo J, Brevoort K, Roy O (2024). gt: Easily Create Presentation-Ready Display Tables. R package version 0.11.1.9000, <https://github.com/rstudio/gt>, <https://gt.rstudio.com>.
- tables: Murdoch D (2024). tables: Formula-Driven Table Generation. R package version 0.9.31, <https://dmurdoch.github.io/tables/>.
- openxlsx: Schauberger P, Walker A (2023). _openxlsx: Read, Write and Edit xlsx Files_. R package version 4.2.5.2, <https://ycphs.github.io/openxlsx/>.

Value

Object of class Tablespan with title, subtitle, header info, data, and footnote.

Examples

```
library(tablespan)
library(dplyr)
data("mtcars")

# We want to report the following table:
```

```
summarized_table <- mtcars |>
  group_by(cyl, vs) |>
  summarise(N = n(),
            mean_hp = mean(hp),
            sd_hp = sd(hp),
            mean_wt = mean(wt),
            sd_wt = sd(wt))

# Create a tablespan:
tbl <- tablespan(data = summarized_table,
                 formula = Cylinder:cyl + Engine:vs ~
                   N +
                   (`Horse Power` = Mean:mean_hp + SD:sd_hp) +
                   (`Weight` = Mean:mean_wt + SD:sd_wt),
                 title = "Motor Trend Car Road Tests",
                 subtitle = "A table created with tablespan",
                 footnote = "Data from the infamous mtcars data set.")

tbl

# Export as Excel table:
wb <- as_excel(tbl = tbl)

# Save using openxlsx
# openxlsx::saveWorkbook(wb, "iris.xlsx")

# Export as gt:
gt_tbl <- as_gt(tbl = tbl)
gt_tbl
```

---

tbl_styles *tbl_styles*

Description

Define styles for different elements of the table.

Usage

```
tbl_styles(
  background_style = openxlsx::createStyle(fgFill = "#ffffff"),
  hline_style = openxlsx::createStyle(border = "Top", borderColour =
    openxlsx::openxlsx_getOp("borderColour", "black"), borderStyle =
    openxlsx::openxlsx_getOp("borderStyle", "double")),
  vline_style = openxlsx::createStyle(border = "Left", borderColour =
    openxlsx::openxlsx_getOp("borderColour", "black"), borderStyle =
    openxlsx::openxlsx_getOp("borderStyle", "double")),
 title_style = openxlsx::createStyle(fontSize = 14, halign = "left", textDecoration =
    "bold"),
```

```
  subtitle_style = openxlsx::createStyle(fontSize = 11, halign = "left", textDecoration =
    "bold"),
  header_style = openxlsx::createStyle(fontSize = 11, halign = "center", border =
   "BottomLeftRight", borderColour = openxlsx::openxlsx_getOp("borderColour", "black"),
   borderStyle = openxlsx::openxlsx_getOp("borderStyle", "double"), textDecoration =
    "bold"),
  merge_rownames = TRUE,
  merged_rownames_style = createStyle(valign = "top"),
  footnote_style = openxlsx::createStyle(fontSize = 11, halign = "left"),
  data_styles = create_data_styles(),
  cell_styles = NULL
)
```

Arguments

background_style	color etc. for the entire background of the table
hline_style	style for the horizontal lines used in the table. Note: the style for the lines under spanners is defined in the title_style.
vline_style	style for the vertical lines used in the table. Note: the style for the lines under spanners is defined in the title_style.
title_style	style applied to the table title
subtitle_style	style applied to the table subtitle
header_style	style applied to the table header (column names)
merge_rownames	boolean: Should adjacent rows with identical names be merged?
merged_rownames_style	style applied to the merged rownames
footnote_style	style applied to the table footnote
data_styles	styles applied to the columns in the data set based on their classes (e.g., numeric, character, etc.). data_styles must be a list of lists. Each inner list must have two elements: a "test" that is used to determine the class of a data colum (e.g., is.double) and a style that is then applied to the columns where the test returns TRUE. Note that styles will be applied in the order of the list, meaning that a later style may overwrite an earlier style.
cell_styles	an optional list with styles for selected cells in the data frame.

Value

a list with styles for different elements of the table

Examples

```
tbl_styles()
```

Index