

# Package: `tablespan` (via `r-universe`)

September 16, 2026

**Type** Package

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

**Version** 0.4.0

**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), gt, huxtable, openxlsx, flextable,  
knitr, rmarkdown, officer, googlesheets4, rvest, purrr

**Depends** R (>= 4.2.0)

**Config/testthat/edition** 3

**Imports** dplyr, grDevices, methods, rlang, scales, tibble, utils

**URL** <https://github.com/jhorzek/tablespan>,  
<https://jhorzek.github.io/tablespan/>

**BugReports** <https://github.com/jhorzek/tablespan/issues>

**VignetteBuilder** knitr

**Repository** <https://jhorzek.r-universe.dev>

**Date/Publication** 2026-04-19 07:59:33 UTC

**RemoteUrl** <https://github.com/jhorzek/tablespan>

**RemoteRef** HEAD

**RemoteSha** 3b445e571a015c6f93d15187b745e57ca8c62905

Contents

|                                  |           |
|----------------------------------|-----------|
| add_fake_sheet . . . . .         | 2         |
| as_excel . . . . .               | 3         |
| as_flextable.Tablespan . . . . . | 4         |
| as_googleSheet_request . . . . . | 5         |
| as_gt . . . . .                  | 7         |
| as_huxtable.Tablespan . . . . .  | 9         |
| as_string . . . . .              | 10        |
| fake_gs4_dribble . . . . .       | 11        |
| format_column . . . . .          | 11        |
| format_date . . . . .            | 13        |
| format_number . . . . .          | 13        |
| format_text . . . . .            | 14        |
| gs_border_style . . . . .        | 14        |
| print.Tablespan . . . . .        | 15        |
| require_flextable . . . . .      | 16        |
| require_googleSheets4 . . . . .  | 16        |
| require_gt . . . . .             | 17        |
| require_huxtable . . . . .       | 17        |
| require_openxlsx . . . . .       | 18        |
| style_column . . . . .           | 19        |
| style_footnote . . . . .         | 20        |
| style_header . . . . .           | 22        |
| style_header_cells . . . . .     | 23        |
| style_hline . . . . .            | 25        |
| style_subtitle . . . . .         | 26        |
| style_title . . . . .            | 28        |
| style_vline . . . . .            | 29        |
| tablespan . . . . .              | 31        |
| <b>Index</b>                     | <b>35</b> |

---

|                |                       |
|----------------|-----------------------|
| add_fake_sheet | <i>add_fake_sheet</i> |
|----------------|-----------------------|

---

Description

Adds a fake sheet to a fake\_sheet dribble.

Usage

add\_fake\_sheet(fake\_sheet, sheet\_name)

Arguments

- |            |                                                  |
|------------|--------------------------------------------------|
| fake_sheet | fake sheet dribble created with fake_gs4_dribble |
| sheet_name | name of the new sheet                            |

**Examples**

```
library(tablespace)
fake_sheet <- fake_gs4_dribble()
add_fake_sheet(fake_sheet, sheet_name = "new_sheet")
```

as\_excel

*as\_excel***Description**

Write a tablespace table to an excel workbook.

**Usage**

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

**Arguments**

|                |                                                           |
|----------------|-----------------------------------------------------------|
| tbl            | table created with tablespace::tablespace                 |
| 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                        |
| merge_rownames | should row names with identical entries be merged?        |

**Value**

openxlsx workbook object that can be edited and saved with openxlsx

**Examples**

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

tbl <- tablespace(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")

# 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.")
if(require_openxlsx(throw = FALSE))
  wb <- as_excel(tbl = tbl)

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

```

---

as\_flextable.Tablespan

*as\_flextable*


---

## Description

Translates a table created with tablespan to a flextable. See <<https://ardata-fr.github.io/flextable-book/>>.

## Usage

```

## S3 method for class 'Tablespan'
as_flextable(x, theme = flextable::theme_booktabs, ...)

```

**Arguments**

|       |                                                                                             |
|-------|---------------------------------------------------------------------------------------------|
| x     | table created with <code>tablespan::tablespan</code>                                        |
| theme | a theme to apply to the flextable. Use one of the <code>flextable::theme_*</code> functions |
| ...   | additional arguments passed to <code>flextable::as_flextable</code>                         |

**Details**

Flextable is an extremely versatile table creator for R with great support to export to a variety of formats.

**Value**

flextable that can be further adapted with the flextable 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)))
if(require_flextable(throw = FALSE)){
  library(flextable)
  flex_tbl <- as_flextable(tbl)
  flex_tbl
}
```

---

as\_googlesheet\_request

*as\_googlesheet\_request*


---

**Description**

Creates a googlesheets4 request to write the tablespan table to a Google Sheet.

**Usage**

```
as_google_sheet_request(
  tbl,
  google_sheet,
  sheet = "Table",
  start_row = 1,
  start_col = 1,
  merge_rownames = TRUE,
  dry_run = is(google_sheet, "fake_sheet"),
  token = NULL,
  escape_formulas = TRUE,
  silent = FALSE
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>tbl</code>             | table created with <code>tablespan::tablespan</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>google_sheet</code>    | Google Sheet spreadsheet dribble created with <code>googlesheets4::gs4_get()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>sheet</code>           | name of the sheet to which the table should be written to                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>start_row</code>       | row at which to start the table                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>start_col</code>       | column at which to start the table                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>merge_rownames</code>  | should row names with identical entries be merged?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>dry_run</code>         | if set to <code>TRUE</code> , no API calls will be made. This is useful when just testing the function                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>token</code>           | optional token for authenticated requests. If <code>NULL</code> and <code>'dry_run = FALSE'</code> , <code>tablespan</code> will use <code>'googlesheets4::gs4_token()'</code> automatically. You can also pass an explicit token from <code>'googlesheets4::gs4_token()'</code> .                                                                                                                                                                                                                                                                         |
| <code>escape_formulas</code> | should formulas be escaped to prevent execution in Google Sheets? Default is <code>TRUE</code> , which means that any cell content starting with <code>"="</code> will be escaped by prefixing it with a single quote ( <code>'</code> ). This ensures that the content is treated as text rather than a formula in Google Sheets. If set to <code>FALSE</code> , cells starting with <code>"="</code> will be written as formulas and executed in Google Sheets. Use with caution if your data may contain content that could be interpreted as formulas. |
| <code>silent</code>          | suppress messages when <code>TRUE</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Details**

`Tablespan` will not directly write to the google sheet. Instead, it will return a `googlesheets4` request that can be used to write the table to a google sheet with `googlesheets4::request_make`

**Value**

A request that can be passed to `googlesheets4::request_make` to write the `tablespan` to a google sheet

## Examples

```
## Not run:
library(tablespace)
library(dplyr)

# 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, create a table with grouping variables as row names and spanners:
tbl <- tablespace(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 tablespace",
                 footnote = "Data from the infamous mtcars data set.")

if(require_googlesheets4(throw = FALSE)) {
  # Get the Google Sheet (replace with your actual sheet URL)
  # google_sheet <- googlesheets4::gs4_get(ss = "link-to-your-google-sheet")
  google_sheet <- fake_gs4_dribble()

  # Create a request to write the data to the googlesheet
  req <- as_googlesheet_request(tbl = tbl,
                               google_sheet = google_sheet,
                               sheet = "Sheet1")

  # For real (non-dry-run) requests, authenticate first and pass a token:
  # googlesheets4::gs4_auth()
  # token <- googlesheets4::gs4_token()
  # req <- as_googlesheet_request(tbl = tbl,
  #                               google_sheet = google_sheet,
  #                               sheet = "Sheet1",
  #                               dry_run = FALSE,
  #                               token = token)

  # Make the actual request:
  # googlesheets4::request_make(req)
}

## End(Not run)
```

## Description

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

## Usage

```
as_gt(
  tbl,
  groupname_col = NULL,
  separator_style = NULL,
  auto_format = TRUE,
  ...
)
```

## Arguments

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

## Details

`Tablesplan` itself does not provide any printing of tables as HTML table. However, with `as_gt`, `tablesplan` 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(tablesplan)
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 <- tablesplan(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)))
if(require_gt(throw = FALSE)){
  gt_tbl <- as_gt(tbl)
  gt_tbl
}

```

---

as\_huxtable.Tablesspan *as\_huxtable*


---

## Description

Translates a table created with `tablespan` to a huxtable. See <https://hughjonesd.github.io/huxtable/index.html>.

## Usage

```

## S3 method for class 'Tablespan'
as_huxtable(x, ...)

```

## Arguments

|                  |                                                                                                                                                    |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>   | table created with <code>tables<span style="font-variant: small-caps;">pan</span>::tables<span style="font-variant: small-caps;">pan</span></code> |
| <code>...</code> | additional arguments passed to <code>huxtable::as_huxtable</code>                                                                                  |

## Details

Huxtable is an extremely versatile table creator for R. Once translated to a huxtable, the `tablespan` table is easy to export to all formats directly supported by huxtable.

## Value

huxtable 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)))
if(require_huxtable(throw = FALSE)){
  library(huxtable)
  hux_tbl <- as_huxtable(tbl)
  hux_tbl
}

```

---

as\_string

as\_string

---

## Description

as\_string

## Usage

```
as_string(tbl, digits = 2, n = 3, ...)
```

## Arguments

|        |                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------|
| tbl    | result from <code>tablespan</code>                                                                              |
| digits | number of digits to round doubles to                                                                            |
| n      | number of rows to print to print the <code>tablespan</code> table. This allows for styling to be printed        |
| ...    | additional arguments passed to <code>prmatrix</code> or <code>huxtable</code> (if <code>use_hux = TRUE</code> ) |

## Value

nothing

## 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)))
cat(as_string(tbl))
```

---

|                  |                         |
|------------------|-------------------------|
| fake_gs4_dribble | <i>fake_gs4_dribble</i> |
|------------------|-------------------------|

---

### Description

Creates a fake googlesheets4 dribble to use as a placeholder in the `as_googlesheets_request` function.

### Usage

```
fake_gs4_dribble()
```

### Value

fake dribble

### Examples

```
library(tablesn)
fake_sheet <- fake_gs4_dribble()
add_fake_sheet(fake_sheet, sheet_name = "new_sheet")
```

---

|               |                      |
|---------------|----------------------|
| format_column | <i>format_column</i> |
|---------------|----------------------|

---

### Description

Change the formatting of a column or single cells within columns.

### Usage

```
format_column(
  tbl,
  columns = dplyr::everything(),
  rows = NULL,
  fmt,
  stack = TRUE
)
```

**Arguments**

|                      |                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>tbl</code>     | tables <span>pan</span> table                                                                                                                                                                                       |
| <code>columns</code> | the columns to style. Must be a tidyselect selector expression (e.g., <code>starts_with("hp_")</code> )                                                                                                             |
| <code>rows</code>    | indices of the rows which should be styled. When set to <code>NULL</code> , the style is applied to all rows                                                                                                        |
| <code>fmt</code>     | formatting object. Use <code>format_number</code> to format numeric values, <code>format_text</code> for text elements, and <code>format_date</code> for dates.                                                     |
| <code>stack</code>   | When set to <code>TRUE</code> , the style is added on top of the existing styles. This is mostly relevant for <code>openxlsx</code> . When set to <code>FALSE</code> , the new style replaces all previous styling. |

**Value**

the tablespan table with added styles

**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.")

if(require_gt(throw = FALSE)){
  tbl |>
    format_column(columns = mean_hp,
                  rows = c(1,3),
                  fmt = format_number(decimals = 4)) |>
  as_gt()
}
```

---

|             |                    |
|-------------|--------------------|
| format_date | <i>format_date</i> |
|-------------|--------------------|

---

**Description**

Implements simple formatting for dates in `tablespan`.

**Usage**

```
format_date(fmt = "%Y-%m-%d")
```

**Arguments**

|     |             |
|-----|-------------|
| fmt | date format |
|-----|-------------|

**Value**

a `tablespan_format`

---

|               |                      |
|---------------|----------------------|
| format_number | <i>format_number</i> |
|---------------|----------------------|

---

**Description**

Implements simple formatting for numbers in `tablespan`.

**Usage**

```
format_number(decimals, sep_mark = ",", dec_mark = ".")
```

**Arguments**

|          |                                            |
|----------|--------------------------------------------|
| decimals | the number of decimals to show             |
| sep_mark | optional symbol used to separate thousands |
| dec_mark | symbol used to separate decimals           |

**Value**

a `tablespan_format`

---

|             |                    |
|-------------|--------------------|
| format_text | <i>format_text</i> |
|-------------|--------------------|

---

**Description**

Implements simple formatting for text in `tablespan`.

**Usage**

```
format_text()
```

**Value**

a `tablespan_format`

---

|                 |                                                              |
|-----------------|--------------------------------------------------------------|
| gs_border_style | <i>Create a Google Sheets API border style specification</i> |
|-----------------|--------------------------------------------------------------|

---

**Description**

Creates a border style specification that can be used with `gs_border_request()` to apply borders to cells in a Google Sheet.

**Usage**

```
gs_border_style(  
  style = c("SOLID", "DOTTED", "DASHED", "SOLID_MEDIUM", "SOLID_THICK", "NONE", "DOUBLE"),  
  width = 1,  
  color  
)
```

**Arguments**

- |       |                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------|
| style | Character. The style of the border. Must be one of: "SOLID", "DOTTED", "DASHED", "SOLID_MEDIUM", "SOLID_THICK", "NONE", or "DOUBLE". |
| width | Numeric. The width of the border in pixels. Default is 1.                                                                            |
| color | The color of the border. Can be a color name, hex code, or a list with red, green, and blue components (values between 0 and 1).     |

**Value**

A list containing the border style specification with elements: `style` (The border style), `width` (The border width), and `color` (The border color as a Google Sheets API color object)

**Examples**

```
library(tablespace)
# Create a solid red border
border_style <- gs_border_style(style = "SOLID", width = 2, color = "red")

# Create a dotted blue border
border_style <- gs_border_style(style = "DOTTED", color = "#0000FF")
```

---

|                 |                        |
|-----------------|------------------------|
| print.Tablespan | <i>print.Tablespan</i> |
|-----------------|------------------------|

---

**Description**

print.Tablespan

**Usage**

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

**Arguments**

|         |                                                                                                                                     |
|---------|-------------------------------------------------------------------------------------------------------------------------------------|
| x       | result from tablespan                                                                                                               |
| digits  | number of digits to round doubles to                                                                                                |
| n       | number of rows to print                                                                                                             |
| use_hux | if set to TRUE and huxtable is installed, huxtable will be used to print the tablespan table. This allows for styling to be printed |
| ...     | additional arguments passed to prmatrix or huxtable (if use_hux = TRUE)                                                             |

**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)
```

---

|                   |                          |
|-------------------|--------------------------|
| require_flextable | <i>require_flextable</i> |
|-------------------|--------------------------|

---

**Description**

Check that flextable is installed

**Usage**

```
require_flextable(throw = TRUE)
```

**Arguments**

|       |                                             |
|-------|---------------------------------------------|
| throw | throw error if the package is not installed |
|-------|---------------------------------------------|

**Value**

boolean or error

**Examples**

```
library(tablespace)
require_flextable()
```

---

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| require_googlesheets4 | <i>Check if googlesheets4 package is available</i> |
|-----------------------|----------------------------------------------------|

---

**Description**

This function checks if the googlesheets4 package is available.

**Usage**

```
require_googlesheets4(throw = TRUE)
```

**Arguments**

|       |                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------|
| throw | logical. If TRUE (default), the function will throw an error if googlesheets4 is not available. If FALSE, it will return FALSE instead. |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------|

**Value**

logical. Returns TRUE if googlesheets4 is available, FALSE if it's not available and throw=FALSE.

**Examples**

```
## Not run:  
# Check if googlesheets4 is available, throw error if not  
require_googlesheets4()  
  
# Check if googlesheets4 is available, return FALSE if not  
require_googlesheets4(throw = FALSE)  
  
## End(Not run)
```

---

*require\_gt**require\_gt*

---

**Description**

Check that gt is installed

**Usage**

```
require_gt(throw = TRUE)
```

**Arguments**

throw                      throw error if the package is not installed

**Value**

boolean or error

**Examples**

```
library(tablespace)  
require_gt()
```

---

*require\_hxtable**require\_hxtable*

---

**Description**

Check that hxtable is installed

**Usage**

```
require_hxtable(throw = TRUE)
```

**Arguments**

throw                      throw error if the package is not installed

**Value**

boolean or error

**Examples**

```
library(tablespace)
require_huxtable()
```

---

|                  |                         |
|------------------|-------------------------|
| require_openxlsx | <i>require_openxlsx</i> |
|------------------|-------------------------|

---

**Description**

Check that openxlsx is installed

**Usage**

```
require_openxlsx(throw = TRUE)
```

**Arguments**

throw                      throw error if the package is not installed

**Value**

boolean or error

**Examples**

```
library(tablespace)
require_openxlsx()
```

---

|              |              |
|--------------|--------------|
| style_column | style_column |
|--------------|--------------|

---

**Description**

Change the style of a column or single cells within columns.

**Usage**

```
style_column(  
  tbl,  
  columns = dplyr::everything(),  
  rows = NULL,  
  background_color = NULL,  
  text_color = NULL,  
  font_size = NULL,  
  bold = FALSE,  
  italic = FALSE,  
  color_scale = NULL,  
  stack = TRUE,  
  ...  
)
```

**Arguments**

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tbl              | tablespan table                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| columns          | the columns to style. Must be a tidyselect selector expression (e.g., starts_with("hp_"))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| rows             | indices of the rows which should be styled. When set to NULL, the style is applied to all rows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| background_color | hex code for the background color                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| text_color       | hex code for the text color                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| font_size        | font size                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| bold             | set to TRUE for bold                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| italic           | set to TRUE for italic                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| color_scale      | a named vector of length 2 or 3 to define a color scale. Example for two colors: color_scale = c("#EE2F43" = -1, "#37E65A" = 1). Example for three colors: color_scale = c("#EE2F43" = -1, "#FFFFFF" = 0, "#37E65A" = 1). If a value is set as NA, it will be replaced with the minimum, mean, or maximum respectively (e.g., color_scale = c("#EE2F43" = -1, "#FFFFFF" = 0, "#37E65A" = 1) will be replaced by color_scale = c("#EE2F43" = min(data), "#FFFFFF" = 0, "#37E65A" = max(data))). NOTE: When exporting to gt, make sure to apply the color scale before you change the text color; otherwise, gt will overwrite the text color. |

|       |                                                                                                                                                                           |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stack | When set to TRUE, the style is added on top of the existing styles. This is mostly relevant for openxlsx. When set to FALSE, the new style replaces all previous styling. |
| ...   | optional additional arguments. Currently not used                                                                                                                         |

### Value

the `tablespan` table with added styles

### 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.")

if(require_gt(throw = FALSE))
tbl |>
  style_column(columns = mean_hp,
              bold = TRUE) |>
  as_gt()
```

---

style\_footnote

*style\_footnote*

---

### Description

Set the style used for the footnote of the `tablespan` table.

**Usage**

```
style_footnote(
  tbl,
  background_color = NULL,
  text_color = NULL,
  font_size = NULL,
  bold = FALSE,
  italic = FALSE,
  ...
)
```

**Arguments**

|                  |                                                   |
|------------------|---------------------------------------------------|
| tbl              | tables <span>pan</span> table                     |
| background_color | hex code for the background color                 |
| text_color       | hex code for the text color                       |
| font_size        | font size                                         |
| bold             | set to TRUE for bold                              |
| italic           | set to TRUE for italic                            |
| ...              | optional additional arguments. Currently not used |

**Details**

The styling for openxlsx and gt works differently:

- openxlsx\_style must be a style object created with openxlsx::createStyle. This style will then be applied to the footnote - gt\_style must be a list of gt::tab\_style objects to be applied to the table

**Value**

the tablespan table with added styles

**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 <- tablesan(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 tablesan",
  footnote = "Data from the infamous mtcars data set.")

if(require_gt(throw = FALSE))
tbl |>
  style_footnote(bold = TRUE) |>
  as_gt()
```

---

|              |                     |
|--------------|---------------------|
| style_header | <i>style_header</i> |
|--------------|---------------------|

---

**Description**

Set the style used for the header of the tablesan table.

**Usage**

```
style_header(
  tbl,
  background_color = NULL,
  text_color = NULL,
  font_size = NULL,
  bold = FALSE,
  italic = FALSE,
  ...
)
```

**Arguments**

|                  |                                                   |
|------------------|---------------------------------------------------|
| tbl              | tablesan table                                    |
| background_color | hex code for the background color                 |
| text_color       | hex code for the text color                       |
| font_size        | font size                                         |
| bold             | set to TRUE for bold                              |
| italic           | set to TRUE for italic                            |
| ...              | optional additional arguments. Currently not used |

**Details**

The styling for openxlsx and gt works differently:

- openxlsx\_style must be a style object created with openxlsx::createStyle. This style will then be applied to the header - gt\_style must be a list of gt::tab\_style objects to be applied to the table

**Value**

the tablespan table with added styles

**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.")

if(require_gt(throw = FALSE))
tbl |>
  style_header(
    openxlsx_style = openxlsx::createStyle(
      fontSize = 8,
      fgFill = "#ffffff"),
    gt_style = list(gt::cell_text(size = 8))) |>
  as_gt()
```

---

style\_header\_cells      style\_header\_cells

---

**Description**

Set the style used for the cells in the openxlsx export. This function is used to create the borders around cells in openxlsx.

**Usage**

```

style_header_cells(
    tbl,
    background_color = NULL,
    text_color = NULL,
    font_size = NULL,
    bold = FALSE,
    italic = FALSE,
    border_color = "#000000",
    top = FALSE,
    bottom = TRUE,
    left = TRUE,
    right = TRUE,
    ...
)

```

**Arguments**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <code>tbl</code>              | tablespan table                                   |
| <code>background_color</code> | hex code for the background color                 |
| <code>text_color</code>       | hex code for the text color                       |
| <code>font_size</code>        | font size                                         |
| <code>bold</code>             | set to TRUE for bold                              |
| <code>italic</code>           | set to TRUE for italic                            |
| <code>border_color</code>     | set the color of the border for the header cells  |
| <code>top</code>              | boolean. Set to TRUE to add a top border          |
| <code>bottom</code>           | boolean. Set to TRUE to add a bottom border       |
| <code>left</code>             | boolean. Set to TRUE to add a left border         |
| <code>right</code>            | boolean. Set to TRUE to add a right border        |
| <code>...</code>              | optional additional arguments. Currently not used |

**Details**

- `openxlsx_style` must be a style object created with `openxlsx::createStyle`. This style will then be applied to the header

**Value**

the tablespan table with added styles

**Examples**

```

library(tablespace)
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 tablespace:
tbl <- tablespace(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 tablespace",
                  footnote = "Data from the infamous mtcars data set.")

if(require_openxlsx(throw = FALSE))
wb <- tbl |>
  style_header_cells(text_color = "#345364") |>
  as_excel()
# save workbook to see the effect

```

---

style\_hline

---

style\_hline

---

**Description**

Set the style used for the horizontal lines of the tablespace table. Currently only supported for excel export.

**Usage**

```
style_hline(tbl, color = "#000000", ...)
```

**Arguments**

|       |                                                   |
|-------|---------------------------------------------------|
| tbl   | tablespace table                                  |
| color | used for the border                               |
| ...   | optional additional arguments. Currently not used |

Details

- openxlsx\_style must be a style object created with openxlsx::createStyle. This style will then be applied to the horizontal lines

Value

the tablesan table with added styles

Examples

```
library(tablesan)
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 tablesan:
tbl <- tablesan(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 tablesan",
               footnote = "Data from the infamous mtcars data set.")

if(require_openxlsx(throw = FALSE))
wb <- tbl |>
  style_hline(
    openxlsx_style = openxlsx::createStyle(
      border = "Top",
      borderColour = "#928505",
      borderStyle = "thin")) |>
  as_excel()
# save workbook to see effect
```

---

|                |                |
|----------------|----------------|
| style_subtitle | style_subtitle |
|----------------|----------------|

---

Description

Set the style used for the subtitle of the tablesan table.

**Usage**

```
style_subtitle(
  tbl,
  background_color = NULL,
  text_color = NULL,
  font_size = NULL,
  bold = FALSE,
  italic = FALSE,
  ...
)
```

**Arguments**

|                  |                                                   |
|------------------|---------------------------------------------------|
| tbl              | tablespan table                                   |
| background_color | hex code for the background color                 |
| text_color       | hex code for the text color                       |
| font_size        | font size                                         |
| bold             | set to TRUE for bold                              |
| italic           | set to TRUE for italic                            |
| ...              | optional additional arguments. Currently not used |

**Details**

The styling for openxlsx and gt works differently:

- openxlsx\_style must be a style object created with openxlsx::createStyle. This style will then be applied to the subtitle
- gt\_style must be a list of gt::tab\_style objects to be applied to the table

**Value**

the tablespan table with added styles

**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 <- tablesan(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 tablesan",
  footnote = "Data from the infamous mtcars data set.")

if(require_gt(throw = FALSE))
tbl |>
  style_subtitle(bold = TRUE) |>
  as_gt()
```

---

|                          |                    |
|--------------------------|--------------------|
| <code>style_title</code> | <i>style_title</i> |
|--------------------------|--------------------|

---

**Description**

Set the style used for the title of the tablesan table.

**Usage**

```
style_title(
  tbl,
  background_color = NULL,
  text_color = NULL,
  font_size = NULL,
  bold = FALSE,
  italic = FALSE,
  ...
)
```

**Arguments**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <code>tbl</code>              | tablesan table                                    |
| <code>background_color</code> | hex code for the background color                 |
| <code>text_color</code>       | hex code for the text color                       |
| <code>font_size</code>        | font size                                         |
| <code>bold</code>             | set to TRUE for bold                              |
| <code>italic</code>           | set to TRUE for italic                            |
| <code>...</code>              | optional additional arguments. Currently not used |

**Details**

The styling for openxlsx and gt works differently:

- openxlsx\_style must be a style object created with openxlsx::createStyle. This style will then be applied to the title
- gt\_style must be a list of gt::tab\_style objects to be applied to the table

**Value**

the tablespan table with added styles

**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.")
if(require_gt(throw = FALSE))
  tbl |>
    style_title(bold = TRUE) |>
    as_gt()
```

---

style\_vline

*style\_vline*


---

**Description**

Set the style used for the vertical lines of the tablespan table. Currently only supported for excel export.

**Usage**

```
style_vline(tbl, color = "#000000", ...)
```

**Arguments**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>tbl</code>   | tables <span>pan</span> table                     |
| <code>color</code> | color used for the border                         |
| <code>...</code>   | optional additional arguments. Currently not used |

**Details**

- `openxlsx_style` must be a style object created with `openxlsx::createStyle`. This style will then be applied to the vertical lines

**Value**

the tablespan table with added styles

**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.")

if(require_openxlsx(throw = FALSE))
wb <- tbl |>
  style_vline(
    openxlsx_style = openxlsx::createStyle(
      border = "Top",
      borderColour = "#928505",
      borderStyle = "thin") |>
    as_excel()
# save workbook to see effect
```

---

|                      |                            |
|----------------------|----------------------------|
| tables <span></span> | <i>tables<span></span></i> |
|----------------------|----------------------------|

---

## Description

Create complex table spanners with a simple formula.

## Usage

```
tables(
  data,
  formula = 1 ~ .,
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  max_digits = 4
)
```

## Arguments

|                         |                                                                  |
|-------------------------|------------------------------------------------------------------|
| <code>data</code>       | data set                                                         |
| <code>formula</code>    | formula to create table                                          |
| <code>title</code>      | string specifying the title of the table                         |
| <code>subtitle</code>   | string specifying the subtitle of the table                      |
| <code>footnote</code>   | string specifying the footnote of the table                      |
| <code>max_digits</code> | the maximal number of digits to print for floating point numbers |

## Details

`tables` 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 and other formats. To this end, `tables` leverages the awesome packages `openxlsx`, `gt`, `flextable`, and `huxtable`.

Following the `tibble` approach, `tables` assumes that all items that you may want to use as row names are just columns in your data set (see example). That is, `tables` 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`: Schauburger P, Walker A (2023). `_openxlsx`: Read, Write and Edit xlsx Files\_. R package version 4.2.5.2, <<https://ycphs.github.io/openxlsx/>>.
- `flextable`: Gohel D, Skintzos P (2025). `_flextable`: Functions for Tabular Reporting\_. R package version 0.9.10, <<https://CRAN.R-project.org/package=flextable>>.
- `huxtable`: Hugh-Jones D (2025). `_huxtable`: Easily Create and Style Tables for LaTeX, HTML and Other Formats\_. R package version 5.8.0, <<https://CRAN.R-project.org/package=huxtable>>.

**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

# Add styling:
tbl <- tbl |>
  style_header(background_color = "#000000", text_color = "ffffff") |>
  style_column(columns = where(is.double), bold = TRUE)

# Export as Excel table:
if(require_openxlsx(throw = FALSE))
  wb <- as_excel(tbl = tbl)

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

# Export as gt:
if(require_gt(throw = FALSE)) {
  as_gt(tbl)
}
# Export as flextable:
if(require_flextable(throw = FALSE)) {
  flextable::as_flextable(tbl)
}
# Export as gt:
if(require_huxtable(throw = FALSE)) {
```

```
    huxtable::as_huxtable(tbl)  
  }
```

# Index

`add_fake_sheet`, [2](#)  
`as_excel`, [3](#)  
`as_flextable.Tablespan`, [4](#)  
`as_google_sheet_request`, [5](#)  
`as_gt`, [7](#)  
`as_huxtable.Tablespan`, [9](#)  
`as_string`, [10](#)  
  
`fake_gs4_dribble`, [11](#)  
`format_column`, [11](#)  
`format_date`, [13](#)  
`format_number`, [13](#)  
`format_text`, [14](#)  
  
`gs_border_style`, [14](#)  
  
`print.Tablespan`, [15](#)  
  
`require_flextable`, [16](#)  
`require_googlesheets4`, [16](#)  
`require_gt`, [17](#)  
`require_huxtable`, [17](#)  
`require_openxlsx`, [18](#)  
  
`style_column`, [19](#)  
`style_footnote`, [20](#)  
`style_header`, [22](#)  
`style_header_cells`, [23](#)  
`style_hline`, [25](#)  
`style_subtitle`, [26](#)  
`style_title`, [28](#)  
`style_vline`, [29](#)  
  
`tablespan`, [31](#)