Package 'flextable'

November 8, 2026

Type Package

Title Functions for Tabular Reporting

Version 0.10.4

Description Use a grammar for creating and customizing pretty tables. The following formats are supported: 'HTML', 'PDF', 'Typst', 'RTF', 'Microsoft Word', 'Microsoft PowerPoint', R 'Grid Graphics' and 'patchwork'. 'R Markdown', 'Quarto' and the package 'officer' can be used to produce the result files. The syntax is the same for the user regardless of the type of output to be produced. A set of functions allows the creation, definition of cell arrangement, addition of headers or footers, formatting and definition of cell content with text and or images. The package also offers a set of high-level functions that allow tabular reporting of statistical models and the creation of complex cross tabulations.

License GPL-3

URL https://ardata-fr.github.io/flextable-book/, https://davidgohel.github.io/flextable/

BugReports https://github.com/davidgohel/flextable/issues

Imports data.table (>= 1.13.0), gdtools (>= 0.5.3), graphics, grDevices, grid, htmltools, knitr, officer (>= 0.7.8), ragg, rlang, rmarkdown (>= 2.0), stats, utils, uuid (>= 0.1-4), xml2

Suggests bookdown (>= 0.40), broom, broom.mixed, chromote, cluster, commonmark, doconv (>= 0.3.0), equatags (>= 0.3.0), formatters, ggplot2, gtable, jsonlite, lme4, magick, mgcv, nlme, officedown, patchwork, pdftools, pkgdown (>= 2.0.0), rtables, scales, svglite, tables (>= 0.9.17), testthat (>= 3.0.0), webshot2, withr, xtable

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Author David Gohel [aut, cre],
ArData [cph],
Clementine Jager [ctb],
Eli Daniels [ctb],
Panagiotis Skintzos [aut],
Quentin Fazilleau [ctb],
Maxim Nazarov [ctb],
Titouan Robert [ctb],
Michael Barrowman [ctb],
Atsushi Yasumoto [ctb],
Paul Julian [ctb],
Sean Browning [ctb],
Rémi Thériault [ctb] (ORCID: <https://orcid.org/0000-0003-4315-6788>),
Samuel Jobert [ctb],
Keith Newman [ctb]

Maintainer David Gohel <david.gohel@ardata.fr>

Repository CRAN

Date/Publication 2026-11-08 21:40:02 UTC

Contents

flextable-package 6
add_body 8
add_body_row 9
add_footer 10
add_footer_lines 12
add_footer_row 12
add_header 14
add_header_lines 15
add_header_row 16
align 18
append_chunks 19
as_b 20
as_bracket 21
as_chunk 22
as_equation 23
as_flextable 24
as_flextable.compact_summary 24
as_flextable.data.frame 25
as_flextable.gam 26
as_flextable.glm 27
as_flextable.grouped_data 28
as_flextable.htest 29
as_flextable.kmeans 30
as_flextable.lm 31

as_flextable.merMod . . . 31
as_flextable.pam . . . 33
as_flextable.summarizor . . . 33
as_flextable.table . . . 34
as_flextable.TableTree . . . 35
as_flextable.tabular . . . 36
as_flextable.tabulator . . . 38
as_flextable.xtable . . . 41
as_grouped_data . . . 42
as_highlight . . . 43
as_i . . . 44
as_image . . . 45
as_paragraph . . . 46
as_qmd . . . 47
as_strike . . . 49
as_sub . . . 50
as_sup . . . 51
as_word_field . . . 52
autofit . . . 53
before . . . 54
bg . . . 55
body_add_flextable . . . 57
body_replace_flextable_at_bkm . . . 59
bold . . . 60
border_inner . . . 61
border_inner_h . . . 61
border_inner_v . . . 62
border_outer . . . 63
border_remove . . . 64
colformat_char . . . 65
colformat_date . . . 66
colformat_datetime . . . 67
colformat_double . . . 68
colformat_image . . . 69
colformat_int . . . 70
colformat_lgl . . . 71
colformat_num . . . 72
color . . . 74
colorize . . . 75
compact_summary . . . 76
compose . . . 77
continuous_summary . . . 78
delete_columns . . . 79
delete_part . . . 80
delete_rows . . . 81
df_printer . . . 82
dim.flextable . . . 83
dim.flextableGrob . . . 84

dim_pretty . . . 84
empty_blanks . . . 85
fit_columns . . . 86
fit_to_width . . . 87
flextable . . . 88
flextable_dim . . . 90
flextable_to_rmd . . . 90
fmt_2stats . . . 91
fmt_avg_dev . . . 93
fmt_dbl . . . 93
fmt_header_n . . . 94
fmt_int . . . 95
fmt_n_percent . . . 96
fmt_pct . . . 97
fmt_signif_after_zeros . . . 98
font . . . 98
fontsize . . . 100
footnote . . . 101
fp_border_default . . . 103
fp_text_default . . . 104
gen_grob . . . 106
get_flextable_defaults . . . 108
gg_chunk . . . 109
grid_chunk . . . 110
height . . . 111
highlight . . . 113
hline . . . 114
hline_bottom . . . 115
hline_top . . . 116
hrule . . . 117
htmltools_value . . . 118
hyperlink_text . . . 119
indentation . . . 120
italic . . . 121
keep_with_next . . . 122
knit_print.flextable . . . 123
labelizor . . . 126
linerange . . . 127
line_spacing . . . 129
merge_at . . . 129
merge_h . . . 130
merge_h_range . . . 131
merge_none . . . 132
merge_v . . . 132
minibar . . . 134
ncol_keys . . . 135
nrow_part . . . 136
padding . . . 136

paginate . . . 138
ph_with.flextable . . . 140
plot.flextable . . . 141
plot.flextableGrob . . . 142
plot_chunk . . . 142
prepend_chunks . . . 144
print.flextable . . . 145
proc_freq . . . 146
repair_docx . . . 147
rotate . . . 148
rtf_add.flextable . . . 150
save_as_docx . . . 151
save_as_html . . . 152
save_as_image . . . 153
save_as_pptx . . . 154
save_as_rtf . . . 155
save_as_typst . . . 157
separate_header . . . 158
set_caption . . . 159
set_flextable_defaults . . . 163
set_formatter . . . 167
set_header_footer_df . . . 168
set_header_labels . . . 169
set_table_properties . . . 170
shift_table . . . 173
split_columns . . . 176
split_rows . . . 177
split_to_pages . . . 178
style . . . 179
summarizor . . . 181
surround . . . 182
tabulator . . . 184
tabulator_colnames . . . 186
tab_settings . . . 188
theme_alafoli . . . 190
theme_apa . . . 191
theme_booktabs . . . 192
theme_borderless . . . 193
theme_box . . . 194
theme_tron . . . 195
theme_tron_legacy . . . 196
theme_vader . . . 197
theme_vanilla . . . 198
theme_zebra . . . 199
to_html.flextable . . . 200
use_df_printer . . . 201
use_flextable_qmd . . . 201
use_model_printer . . . 202

valign 203
vline 204
vline_left 205
vline_right 206
void 207
width 208
wrap_flextable 209

Index **213**

`flextable-package` *flextable: Functions for Tabular Reporting*

Description

The flextable package facilitates access to and manipulation of tabular reporting elements from R.

The documentation of functions can be opened with command `help(package = "flextable")`.

`flextable()` function is producing flexible tables where each cell can contain several chunks of text with their own set of formatting properties (bold, font color, etc.). Function `mk_par()` lets customise text of cells.

Each cell holds a single paragraph composed of inline chunks (see `as_paragraph()`). This means cell content is strictly inline: bold, italic, links, images, equations, inline code, etc. Block-level structures (multiple paragraphs, bullet lists, headings or fenced code blocks) cannot be placed inside a cell. Soft line breaks (`\n`) are however supported.

The `as_flextable()` function is used to transform specific objects into flextable objects. For example, you can transform a crosstab produced with the 'tables' package into a flextable which can then be formatted, annotated or augmented with footnotes.

In order to reduce the homogenization efforts and the number of functions to be called, it is recommended to define formatting properties such as font, border color, number of decimals displayed which will then be applied by default. See `set_flextable_defaults()` for more details.

Table Structure

A flextable is composed of three distinct parts:

- `header`: By default, contains one row with the column names from the data.frame
- `body`: Contains the actual data from the data.frame
- `footer`: Empty by default, but can contain content (commonly used for footnotes or #' summary rows)

```
       HEADER         <- Column names, labels, spanning headers
--------------------
        BODY          <- Data rows
--------------------
       FOOTER         <- Summary rows, notes, footnotes (optional)
```

A basic flextable has:

- in the part 'header': 1 row with column names
- in the part 'body': as many rows as there are in the input data.frame
- no footer

Rows and columns can be added or removed to the basic flextable:

- Add new rows in header with `add_header()`, `add_header_row()`, `add_header_lines()` and `set_header_labels()`.
- Add new rows in footer with `add_footer()`, `add_footer_lines()`, `set_header_footer_df()` and `add_footer_row()`.
- Add new rows in body with `add_body()` and `add_body_row()`.
- Delete columns with `delete_columns()`.
- Delete a part with `delete_part()`.
- Use column names to separate a simple header row into multiple nested rows with `separate_header()`.

Selectors in flextable

Selectors are a core feature of flextable that allow you to specify which parts (`part`), rows (`i`) and columns (`j`) should be affected by formatting, styling, or content operations. See the corresponding manual: <`Selectors in flextable`>.

Author(s)

Maintainer: David Gohel <david.gohel@ardata.fr>

Authors:

- David Gohel <david.gohel@ardata.fr>
- Panagiotis Skintzos <panagiotis.skintzos@ardata.fr>

Other contributors:

- ArData [copyright holder]
- Clementine Jager [contributor]
- Eli Daniels [contributor]
- Quentin Fazilleau [contributor]
- Maxim Nazarov [contributor]
- Titouan Robert [contributor]
- Michael Barrowman [contributor]
- Atsushi Yasumoto [contributor]
- Paul Julian [contributor]
- Sean Browning [contributor]
- Rémi Thériault (ORCID) [contributor]
- Samuel Jobert [contributor]
- Keith Newman [contributor]

See Also

https://davidgohel.github.io/flextable/, https://ardata-fr.github.io/flextable-book/, flextable()

---

add_body *Add body rows with one value per column*

Description

Add new rows to the body where each value maps to a named column, preserving the original column data types. Unlike add_body_row() where labels can span multiple columns, here each value fills exactly one column.

If some columns are not provided, they will be replaced by NA and displayed as empty.

Usage

```
add_body(x, top = TRUE, ..., values = NULL)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
top	should the rows be inserted at the top or the bottom.
...	named arguments (names are data colnames) of values to add. It is important to insert data of the same type as the original data, otherwise it will be transformed (probably into strings if you add a character where a double is expected). This makes possible to still format cell contents with the colformat_* functions, for example colformat_num().
values	a list of name-value pairs of labels or values, names should be existing col_key values. This argument can be used instead of ... for programming purpose (If values is supplied argument ... is ignored).

See Also

flextable()

Other row and column operations: add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
ft <- flextable(head(iris),
  col_keys = c(
    "Species", "Sepal.Length", "Petal.Length",
    "Sepal.Width", "Petal.Width"
  )
)

ft <- add_body(
  x = ft, Sepal.Length = 1:5,
  Sepal.Width = 1:5 * 2, Petal.Length = 1:5 * 3,
  Petal.Width = 1:5 + 10, Species = "Blah", top = FALSE
)

ft <- theme_booktabs(ft)
ft
```

---

add_body_row *Add a body row with spanning labels*

Description

Add a single row to the body where labels can span multiple columns (merged cells) via the `colwidths` argument.

Labels are associated with a number of columns to merge that default to one if not specified. In this case, you have to make sure that the number of labels is equal to the number of columns displayed.

The function can add only one single row by call.

Labels can also be formatted with `as_paragraph()`.

Usage

```
add_body_row(x, top = TRUE, values = list(), colwidths = integer(0))
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`top`	should the row be inserted at the top or the bottom.
`values`	values to add. It can be a `list`, a `character()` vector or a call to `as_paragraph()`. If it is a list, it can be a named list with the names of the columns of the original data.frame or the `colkeys`; this is the recommended method because it allows to keep the original data types and therefore allows to perform conditional formatting. If a character, columns of the original data.frame stored in the flextable object are changed to `character()`; this is often not an issue with footer and header but can be inconvenient if adding rows into body as it will change data types to character and prevent efficient conditional formatting.
`colwidths`	the number of columns to merge in the row for each label

See Also

flextable(), set_caption()

Other row and column operations: add_body(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
library(flextable)

ft01 <- fp_text_default(color = "red")
ft02 <- fp_text_default(color = "orange")

pars <- as_paragraph(
  as_chunk(c("(1)", "(2)"), props = ft02), " ",
  as_chunk(
    c(
      "My tailor is rich",
      "My baker is rich"
    ),
    props = ft01
  )
)

ft_1 <- flextable(head(mtcars))
ft_1 <- add_body_row(ft_1,
  values = pars,
  colwidths = c(5, 6), top = FALSE
)
ft_1 <- add_body_row(ft_1,
  values = pars,
  colwidths = c(3, 8), top = TRUE
)
ft_1 <- theme_box(ft_1)
ft_1

ft_2 <- flextable(head(airquality))
ft_2 <- add_body_row(ft_2,
  values = c("blah", "bleeeh"),
  colwidths = c(4, 2), top = TRUE
)
ft_2 <- theme_box(ft_2)
ft_2
```

---

add_footer *Add footer rows with one value per column*

Description

Add new rows to the footer where each value maps to a named column. Unlike add_footer_row() where labels can span multiple columns, here each value fills exactly one column.

If some columns are not provided, they will be replaced by NA and displayed as empty.

Usage

```
add_footer(x, top = TRUE, ..., values = NULL)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
top	should the rows be inserted at the top or the bottom.
...	named arguments (names are data colnames) of values to add. It is important to insert data of the same type as the original data, otherwise it will be transformed (probably into strings if you add a character where a double is expected). This makes possible to still format cell contents with the colformat_* functions, for example colformat_num().
values	a list of name-value pairs of labels or values, names should be existing col_key values. This argument can be used instead of ... for programming purpose (If values is supplied argument ... is ignored).

See Also

Other row and column operations: add_body(), add_body_row(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
new_row <- as.list(colMeans(iris[, -5]))
new_row$Species <- "Means"

formatter <- function(x) sprintf("%.1f", x)

ft <- flextable(data = head(iris))
ft <- add_footer(ft, values = new_row)

# cosmetics
ft <- compose(
  x = ft, j = 1:4,
  value = as_paragraph(
    as_chunk(., formatter = formatter)
  ),
  part = "footer", use_dot = TRUE
)
ft <- align(ft, part = "footer", align = "right", j = 1:4)
ft
```

add_footer_lines *Add full-width rows to the footer*

Description

Add one or more rows to the footer where each label spans all columns (all cells merged into one). Useful for adding footnotes or source notes below the table.

Usage

```
add_footer_lines(x, values = character(0), top = FALSE)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
values	a character vector or a call to as_paragraph() to get formated content, each element will be added as a new row.
top	should the row be inserted at the top or the bottom. Default to TRUE.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- add_footer_lines(ft_1,
  values = c("blah 1", "blah 2")
)
ft_1
```

add_footer_row *Add a footer row with spanning labels*

Description

Add a single row to the footer where labels can span multiple columns (merged cells) via the colwidths argument.

Labels are associated with a number of columns to merge that default to one if not specified. In this case, you have to make sure that the number of labels is equal to the number of columns displayed.

The function can add only one single row by call.

Labels can be formatted with as_paragraph().

Usage

```
add_footer_row(x, top = TRUE, values = character(0), colwidths = integer(0))
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
top	should the row be inserted at the top or the bottom.
values	values to add. It can be a `list`, a `character()` vector or a call to `as_paragraph()`. If it is a list, it can be a named list with the names of the columns of the original data.frame or the `colkeys`; this is the recommended method because it allows to keep the original data types and therefore allows to perform conditional formatting. If a character, columns of the original data.frame stored in the flextable object are changed to `character()`; this is often not an issue with footer and header but can be inconvenient if adding rows into body as it will change data types to character and prevent efficient conditional formatting.
colwidths	the number of columns to merge in the row for each label

See Also

`flextable()`, `set_caption()`

Other row and column operations: `add_body()`, `add_body_row()`, `add_footer()`, `add_footer_lines()`, `add_header()`, `add_header_lines()`, `add_header_row()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `paginate()`, `separate_header()`, `set_header_footer_df`, `set_header_labels()`, `split_columns()`, `split_rows()`, `split_to_pages()`

Examples

```
library(flextable)

ft01 <- fp_text_default(color = "red")
ft02 <- fp_text_default(color = "orange")

pars <- as_paragraph(
  as_chunk(c("(1)", "(2)"), props = ft02), " ",
  as_chunk(
    c(
      "My tailor is rich",
      "My baker is rich"
    ),
    props = ft01
  )
)

ft_1 <- flextable(head(mtcars))
ft_1 <- add_footer_row(ft_1,
  values = pars,
  colwidths = c(5, 6), top = FALSE
)
```

```
ft_1 <- add_footer_row(ft_1,
  values = pars,
  colwidths = c(3, 8), top = TRUE
)
ft_1

ft_2 <- flextable(head(airquality))
ft_2 <- add_footer_row(ft_2,
  values = c("Measure", "Time"),
  colwidths = c(4, 2), top = TRUE
)
ft_2 <- theme_box(ft_2)
ft_2
```

---

add_header *Add header rows with one value per column*

Description

Add new rows to the header where each value maps to a named column. Unlike add_header_row() where labels can span multiple columns, here each value fills exactly one column.

If some columns are not provided, they will be replaced by NA and displayed as empty.

Usage

```
add_header(x, top = TRUE, ..., values = NULL)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
top	should the rows be inserted at the top or the bottom.
...	named arguments (names are data colnames) of values to add. It is important to insert data of the same type as the original data, otherwise it will be transformed (probably into strings if you add a character where a double is expected). This makes possible to still format cell contents with the colformat_* functions, for example colformat_num().
values	a list of name-value pairs of labels or values, names should be existing col_key values. This argument can be used instead of ... for programming purpose (If values is supplied argument ... is ignored).

Note

when repeating values, they can be merged together with function merge_h() and merge_v().

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
library(flextable)

fun <- function(x) {
  paste0(
    c("min: ", "max: "),
    formatC(range(x))
  )
}
new_row <- list(
  Sepal.Length = fun(iris$Sepal.Length),
  Sepal.Width =  fun(iris$Sepal.Width),
  Petal.Width =  fun(iris$Petal.Width),
  Petal.Length = fun(iris$Petal.Length)
)

ft_1 <- flextable(data = head(iris))
ft_1 <- add_header(ft_1, values = new_row, top = FALSE)
ft_1 <- append_chunks(ft_1, part = "header", i = 2, )
ft_1 <- theme_booktabs(ft_1, bold_header = TRUE)
ft_1 <- align(ft_1, align = "center", part = "all")
ft_1
```

---

add_header_lines *Add full-width rows to the header*

Description

Add one or more rows to the header where each label spans all columns (all cells merged into one). Useful for adding titles or subtitles above the column headers.

Usage

```
add_header_lines(x, values = character(0), top = TRUE)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
values	a character vector or a call to as_paragraph() to get formated content, each element will be added as a new row.
top	should the row be inserted at the top or the bottom. Default to TRUE.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
# ex 1----
ft_1 <- flextable(head(iris))
ft_1 <- add_header_lines(ft_1, values = "blah blah")
ft_1 <- add_header_lines(ft_1, values = c("blah 1", "blah 2"))
ft_1 <- autofit(ft_1)
ft_1

# ex 2----
ft01 <- fp_text_default(color = "red")
ft02 <- fp_text_default(color = "orange")
ref <- c("(1)", "(2)")
pars <- as_paragraph(
  as_chunk(ref, props = ft02), " ",
  as_chunk(rep("My tailor is rich", length(ref)), props = ft01)
)

ft_2 <- flextable(head(mtcars))
ft_2 <- add_header_lines(ft_2, values = pars, top = FALSE)
ft_2 <- add_header_lines(ft_2, values = ref, top = TRUE)
ft_2 <- add_footer_lines(ft_2, values = "blah", top = TRUE)
ft_2 <- add_footer_lines(ft_2, values = pars, top = TRUE)
ft_2 <- add_footer_lines(ft_2, values = ref, top = FALSE)
ft_2 <- autofit(ft_2)
ft_2
```

---

add_header_row *Add a header row with spanning labels*

Description

Add a single row to the header where labels can span multiple columns (merged cells) via the colwidths argument.

Labels are associated with a number of columns to merge that default to one if not specified. In this case, you have to make sure that the number of labels is equal to the number of columns displayed.

The function can add only one single row by call.

Labels can also be formatted with as_paragraph().

Usage

```
add_header_row(x, top = TRUE, values = character(0), colwidths = integer(0))
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
top	should the row be inserted at the top or the bottom. Default to TRUE.
values	values to add, a character vector (as header rows contains only character values/columns), a list or a call to as_paragraph().
colwidths	the number of columns used for each label

See Also

flextable(), set_caption()

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
library(flextable)

ft01 <- fp_text_default(color = "red")
ft02 <- fp_text_default(color = "orange")

pars <- as_paragraph(
  as_chunk(c("(1)", "(2)"), props = ft02), " ",
  as_chunk(c(
    "My tailor is rich",
    "My baker is rich"
  ), props = ft01)
)

ft_1 <- flextable(head(mtcars))
ft_1 <- add_header_row(ft_1,
  values = pars,
  colwidths = c(5, 6), top = FALSE
)
ft_1 <- add_header_row(ft_1,
  values = pars,
  colwidths = c(3, 8), top = TRUE
)
ft_1

ft_2 <- flextable(head(airquality))
ft_2 <- add_header_row(ft_2,
  values = c("Measure", "Time"),
  colwidths = c(4, 2), top = TRUE
)
ft_2 <- theme_box(ft_2)
ft_2
```

`align` *Set text alignment*

Description

Change the text alignment of selected rows and columns of a flextable.

Usage

```
align(
  x,
  i = NULL,
  j = NULL,
  align = "left",
  part = c("body", "header", "footer", "all")
)

align_text_col(x, align = "left", header = TRUE, footer = TRUE)

align_nottext_col(x, align = "right", header = TRUE, footer = TRUE)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`align`	text alignment - a single character value, or a vector of character values equal in length to the number of columns selected by `j`. Expected values must be from the set ('left', 'right', 'center', or 'justify'). If the number of columns is a multiple of the length of the `align` parameter, then the values in `align` will be recycled across the remaining columns.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.
`header`	should the header be aligned with the body
`footer`	should the footer be aligned with the body

See Also

Other formatting shortcuts: `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
# Table of 6 columns
ft_car <- flextable(head(mtcars)[, 2:7])

# All 6 columns right aligned
align(ft_car, align = "right", part = "all")

# Manually specify alignment of each column
align(
  ft_car,
  align = c("left", "right", "left", "center", "center", "right"),
  part = "all"
)

# Center-align column 2 and left-align column 5
align(ft_car, j = c(2, 5), align = c("center", "left"), part = "all")

# Alternate left and center alignment across columns 1-4 for header only
align(ft_car, j = 1:4, align = c("left", "center"), part = "header")
ftab <- flextable(mtcars)
ftab <- align_text_col(ftab, align = "left")
ftab <- align_nottext_col(ftab, align = "right")
ftab
```

---

append_chunks *Append chunks to flextable content*

Description

append chunks (for example chunk `as_chunk()`) in a flextable.

Usage

```
append_chunks(x, ..., i = NULL, j = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	chunks to be appened, see `as_chunk()`, `gg_chunk()` and other chunk elements for paragraph.
i	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
j	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
part	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' is not allowed by the function.

See Also

as_chunk(), as_sup(), as_sub(), colorize()

Other functions to compose cell content: as_paragraph(), compose(), footnote(), labelizor(), prepend_chunks(), void()

Examples

```
library(flextable)
img.file <- file.path(R.home("doc"), "html", "logo.jpg")

ft_1 <- flextable(head(cars))

ft_1 <- append_chunks(ft_1,
  # where to append
  i = c(1, 3, 5),
  j = 1,
  # what to append
  as_chunk(" "),
  as_image(src = img.file, width = .20, height = .15)
)
ft_1 <- set_table_properties(ft_1, layout = "autofit")
ft_1
```

---

as_b *Bold chunk*

Description

The function is producing a chunk with bold font.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_b(x)
```

Arguments

x	value, if a chunk, the chunk will be updated

See Also

Other chunk elements for paragraph: as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Sepal.Length", "dummy")
)

ft <- compose(ft,
  j = "dummy",
  value = as_paragraph(
    as_b(Sepal.Length)
  )
)

ft
```

`as_bracket` *Bracket chunk*

Description

The function is producing a chunk by pasting values and add the result in brackets.

It is used to add it to the content of a cell of the flextable with the functions `compose()`, `append_chunks()` or `prepend_chunks()`.

Usage

```
as_bracket(..., sep = ", ", p = "(", s = ")")
```

Arguments

`...`	text and column names
`sep`	separator
`p`	prefix, default to '('
`s`	suffix, default to ')'

See Also

Other chunk elements for paragraph: `as_b()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Species", "Sepal", "Petal")
)
ft <- set_header_labels(ft, Sepal = "Sepal", Petal = "Petal")
ft <- compose(ft,
```

```
  j = "Sepal",
  value = as_paragraph(as_bracket(Sepal.Length, Sepal.Width))
)
ft <- compose(ft,
  j = "Petal",
  value = as_paragraph(as_bracket(Petal.Length, Petal.Width))
)
ft
```

---

as_chunk *Text chunk*

Description

The function lets add formated text in flextable cells.

It is used to add it to the content of a cell of the flextable with the functions `compose()`, `append_chunks()` or `prepend_chunks()`.

It should be used inside a call to `as_paragraph()`.

Usage

```
as_chunk(x, props = NULL, formatter = format_fun, ...)
```

Arguments

x	text or any element that can be formatted as text with function provided in argument `formatter`.
props	an `fp_text_default()` or `officer::fp_text()` object to be used to format the text. If not specified, it will be the default value corresponding to the cell.
formatter	a function that will format x as a character vector.
...	additional arguments for `formatter` function.

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
library(officer)

ft <- flextable(head(iris))

ft <- compose(ft,
  j = "Sepal.Length",
  value = as_paragraph(
```

```
    "Sepal.Length value is ",
    as_chunk(Sepal.Length, props = fp_text(color = "red"))
  ),
  part = "body"
)
ft <- color(ft, color = "gray40", part = "all")
ft <- autofit(ft)
ft
```

`as_equation` *Equation chunk*

Description

This function is used to insert equations into flextable.

It is used to add it to the content of a cell of the flextable with the functions `compose()`, `append_chunks()` or `prepend_chunks()`.

To use this function, package 'equatags' is required.

Usage

```
as_equation(x, width = 1, height = 0.2, unit = "in", props = NULL)
```

Arguments

`x`	values containing the 'MathJax' equations
`width, height`	size of the resulting equation
`unit`	unit for width and height, one of "in", "cm", "mm".
`props`	an `fp_text_default()` or `officer::fp_text()` object to be used to format the text. If not specified, it will be the default value corresponding to the cell.

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
library(flextable)
if (require("equatags")) {
  eqs <- c(
    "(ax^2 + bx + c = 0)",
    "a \\ne 0",
    "x = {-b \\pm \\sqrt{b^2-4ac} \\over 2a}"
  )
  df <- data.frame(formula = eqs)
```

```
  df


  ft <- flextable(df)
  ft <- compose(
    x = ft, j = "formula",
    value = as_paragraph(as_equation(formula, width = 2, height = .5))
  )
  ft <- align(ft, align = "center", part = "all")
  ft <- width(ft, width = 2)
  ft
}
```

`as_flextable` *Method to transform objects into flextables*

Description

This is a convenient function to let users create flextable bindings from any objects. Users should consult documentation of corresponding method to understand the details and see what arguments can be used.

Usage

```
as_flextable(x, ...)
```

Arguments

`x`	object to be transformed as flextable
`...`	arguments for custom methods

See Also

Other flextable constructors: `flextable()`

`as_flextable.compact_summary`
Transform a 'compact_summary' object into a flextable

Description

`compact_summary` objects can be transformed into a flextable with method `as_flextable()`.

Numeric columns are formatted with `formatC()` using the `digits` value stored in the object and the current flextable defaults for `big.mark` and `decimal.mark`.

Usage

```
## S3 method for class 'compact_summary'
as_flextable(x, ...)
```

Arguments

x	A compact_summary object produced by compact_summary().
...	unused arguments.

Value

A flextable() object.

See Also

Other as_flextable methods: as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
z <- compact_summary(iris, show_type = TRUE, show_na = TRUE)
as_flextable(z)
```

---

as_flextable.data.frame

Transform and summarise a 'data.frame' into a flextable Simple summary of a data.frame as a flextable

---

Description

It displays the first rows and shows the column types. If there is only one row, a simplified vertical table is produced.

Usage

```
## S3 method for class 'data.frame'
as_flextable(
  x,
  max_row = 10,
  split_colnames = FALSE,
  short_strings = FALSE,
  short_size = 35,
  short_suffix = "...",
  do_autofit = TRUE,
  show_coltype = TRUE,
```

```
  color_coltype = "#999999",
  ...
)
```

Arguments

x	a data.frame
max_row	The number of rows to print. Default to 10.
split_colnames	Should the column names be split (with non alpha-numeric characters). Default to FALSE.
short_strings	Should the character column be shorten. Default to FALSE.
short_size	Maximum length of character column if short_strings is TRUE. Default to 35.
short_suffix	Suffix to add when character values are shorten. Default to "...".
do_autofit	Use autofit() before rendering the table. Default to TRUE.
show_coltype	Show column types. Default to TRUE.
color_coltype	Color to use for column types. Default to "#999999".
...	unused arguments

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
as_flextable(mtcars)
```

---

as_flextable.gam *Transform a 'gam' model into a flextable*

Description

produce a flextable describing a generalized additive model produced by function mgcv::gam.

Usage

```
## S3 method for class 'gam'
as_flextable(x, ...)
```

Arguments

x	gam model
...	unused argument

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
if (require("mgcv")) {
  set.seed(2)

  # Simulated data
  dat <- gamSim(1, n = 400, dist = "normal", scale = 2)

  # basic GAM model
  b <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat)

  ft <- as_flextable(b)
  ft
}
```

---

as_flextable.glm *Transform a 'glm' object into a flextable*

Description

produce a flextable describing a generalized linear model produced by function glm.

You can remove significance stars by setting options options(show.signif.stars = FALSE).

Usage

```
## S3 method for class 'glm'
as_flextable(x, ...)
```

Arguments

x	glm model
...	unused argument

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
if (require("broom")) {
  dat <- attitude
  dat$high.rating <- (dat$rating > 70)
  probit.model <- glm(high.rating ~ learning + critical +
    advance, data = dat, family = binomial(link = "probit"))
  ft <- as_flextable(probit.model)
  ft
}
```

---

as_flextable.grouped_data

Transform a 'grouped_data' object into a flextable

Description

Produce a flextable from a table produced by function as_grouped_data().

Usage

```
## S3 method for class 'grouped_data'
as_flextable(x, col_keys = NULL, hide_grouplabel = FALSE, ...)
```

Arguments

x	'grouped_data' object to be transformed into a "flextable"
col_keys	columns names/keys to display. If some column names are not in the dataset, they will be added as blank columns by default.
hide_grouplabel	if TRUE, group label will not be rendered, only level/value will be rendered.
...	unused argument

See Also

as_grouped_data()

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
library(data.table)
CO2 <- CO2
setDT(CO2)
CO2$conc <- as.integer(CO2$conc)

data_co2 <- dcast(CO2, Treatment + conc ~ Type,
  value.var = "uptake", fun.aggregate = mean
)
data_co2 <- as_grouped_data(x = data_co2, groups = c("Treatment"))

ft <- as_flextable(data_co2)
ft <- add_footer_lines(ft, "dataset CO2 has been used for this flextable")
ft <- add_header_lines(ft, "mean of carbon dioxide uptake in grass plants")
ft <- set_header_labels(ft, conc = "Concentration")
ft <- autofit(ft)
ft <- width(ft, width = c(1, 1, 1))
ft
```

---

as_flextable.htest *Transform a 'htest' object into a flextable*

Description

produce a flextable describing an object oof class `htest`.

Usage

```
## S3 method for class 'htest'
as_flextable(x, ...)
```

Arguments

x	htest object
...	unused argument

See Also

Other as_flextable methods: `as_flextable.compact_summary()`, `as_flextable.data.frame()`, `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.kmeans()`, `as_flextable.lm()`, `as_flextable.merMod()`, `as_flextable.pam()`, `as_flextable.summarizor()`, `as_flextable.table()`, `as_flextable.tabular()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `compact_summary()`

Examples

```
if (require("stats")) {
  M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
  dimnames(M) <- list(
    gender = c("F", "M"),
    party = c("Democrat", "Independent", "Republican")
  )
  ft_1 <- as_flextable(chisq.test(M))
  ft_1
}
```

---

as_flextable.kmeans *Transform a 'kmeans' object into a flextable*

Description

produce a flextable describing a kmeans object. The function is only using package 'broom' that provides the data presented in the resulting flextable.

Usage

```
## S3 method for class 'kmeans'
as_flextable(x, digits = 4, ...)
```

Arguments

x	a kmeans() object
digits	number of digits for the numeric columns
...	unused argument

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
if (require("stats")) {
  cl <- kmeans(scale(mtcars[1:7]), 5)
  ft <- as_flextable(cl)
  ft
}
```

as_flextable.lm *Transform a 'lm' object into a flextable*

Description

produce a flextable describing a linear model produced by function `lm`.

You can remove significance stars by setting options `options(show.signif.stars = FALSE)`.

Usage

```
## S3 method for class 'lm'
as_flextable(x, ...)
```

Arguments

`x`	lm model
`...`	unused argument

See Also

Other as_flextable methods: `as_flextable.compact_summary()`, `as_flextable.data.frame()`, `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.kmeans()`, `as_flextable.merMod()`, `as_flextable.pam()`, `as_flextable.summarizor()`, `as_flextable.table()`, `as_flextable.tabular()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `compact_summary()`

Examples

```
if (require("broom")) {
  lmod <- lm(rating ~ complaints + privileges +
    learning + raises + critical, data = attitude)
  ft <- as_flextable(lmod)
  ft
}
```

as_flextable.merMod *Transform a 'merMod' or 'lme' object into a flextable*

Description

produce a flextable describing a mixed model. The function is only using package 'broom.mixed' that provides the data presented in the resulting flextable.

You can remove significance stars by setting options `options(show.signif.stars = FALSE)`.

Usage

```
## S3 method for class 'merMod'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'lme'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'gls'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'nlme'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'brmsfit'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'glmmTMB'
as_flextable(x, add.random = TRUE, ...)

## S3 method for class 'glmmadmb'
as_flextable(x, add.random = TRUE, ...)
```

Arguments

x	a mixed model
add.random	TRUE or FALSE, if TRUE random effects are added to the table.
...	unused argument

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
if (require("broom.mixed") && require("nlme")) {
  m1 <- lme(distance ~ age, data = Orthodont)
  ft <- as_flextable(m1)
  ft
}
```

as_flextable.pam *Transform a 'pam' object into a flextable*

Description

produce a flextable describing a pam object. The function is only using package 'broom' that provides the data presented in the resulting flextable.

Usage

```
## S3 method for class 'pam'
as_flextable(x, digits = 4, ...)
```

Arguments

x	a cluster::pam() object
digits	number of digits for the numeric columns
...	unused argument

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
if (require("cluster")) {
  dat <- as.data.frame(scale(mtcars[1:7]))
  cl <- pam(dat, 3)
  ft <- as_flextable(cl)
  ft
}
```

as_flextable.summarizor *Transform a 'summarizor' object into a flextable*

Description

summarizor object should be transformed into a flextable with method as_flextable().

Usage

```
## S3 method for class 'summarizor'
as_flextable(x, ...)
```

Arguments

x	result from summarizor()
...	arguments for as_flextable.tabulator()

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
z <- summarizor(CO2[-c(1, 4)],
  by = "Treatment",
  overall_label = "Overall"
)
ft_1 <- as_flextable(z, spread_first_col = TRUE)
ft_1 <- prepend_chunks(ft_1,
  i = ~ is.na(variable), j = 1,
  as_chunk("\t")
)
ft_1 <- autofit(ft_1)
ft_1
```

---

as_flextable.table *Transform a 'table' object into a flextable*

Description

produce a flextable describing a count table produced by function table().

This function uses the proc_freq() function.

Usage

```
## S3 method for class 'table'
as_flextable(x, ...)
```

Arguments

x	table object
...	arguments used by proc_freq().

See Also

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), compact_summary()

Examples

```
tab <- with(warpbreaks, table(wool, tension))
ft <- as_flextable(tab)
ft
```

---

as_flextable.TableTree

Transform an rtables object into a flextable

Description

produce a flextable from a TableTree or ElementaryTable object produced with the 'rtables' package.

The conversion uses formatters::matrix_form() to extract the formatted content, column spans, alignments, indentation and footnotes, then maps them to flextable features.

Indentation of row labels is rendered with left padding.

Label rows (LabelRow) are displayed in bold in the first column. Content rows (ContentRow) are displayed entirely in bold.

When LabelRow groups exist, paginate() is applied so that the rows of the smallest labelled groups are kept together on the same page in Word and RTF output. Each group starts at a label row, and the labels of its parent groups are kept with the first of their child groups.

To paginate the resulting flextable into multiple pages, use split_to_pages(), split_rows(), or split_columns() after calling as_flextable().

Usage

```
## S3 method for class 'TableTree'
as_flextable(x, indent_padding = 4, ...)

## S3 method for class 'ElementaryTable'
as_flextable(x, indent_padding = 4, ...)
```

Arguments

x	a TableTree or ElementaryTable object produced by the 'rtables' package.
indent_padding	base left padding in points per indentation level. Default is 4.
...	unused arguments

Value

a flextable object.

See Also

split_to_pages(), split_rows(), split_columns()

Examples

```
library(rtables)

lyt <- basic_table(title = "Demographic Summary")
lyt <- split_cols_by(lyt, "ARM")
lyt <- split_rows_by(lyt, "SEX")
lyt <- analyze(lyt, "AGE", afun = mean, format = "xx.x")

tbl <- build_table(lyt, DM)
as_flextable(tbl)
```

---

as_flextable.tabular *Transform a 'tables::tabular' object into a flextable*

Description

Produce a flextable from a 'tabular' object produced with function tables::tabular().

When as_flextable.tabular=TRUE, the first column is used as row separator acting as a row title. It can be formated with arguments fp_p (the formatting properties of the paragraph) and row_title that specifies the content and eventually formattings of the content.

Two hidden columns can be used for conditional formatting after the creation of the flextable (use only when spread_first_col=TRUE):

- The column .row_title that contains the title label
- The column .type that can contain the following values:
  - "one_row": Indicates that there is only one row for this group. In this case, the row is not expanded with a title above.
  - "list_title": Indicates a row that serves as a title for the data that are displayed after it.
  - "list_data": Indicates rows that follow a title and contain data to be displayed.

The result is paginated (see paginate()).

Usage

```
## S3 method for class 'tabular'
as_flextable(
  x,
  spread_first_col = FALSE,
  fp_p = fp_par(text.align = "center", padding.top = 4),
  row_title = as_paragraph(as_chunk(.row_title)),
  add_tab = FALSE,
  ...
)
```

Arguments

x	object produced by `tables::tabular()`.
spread_first_col	if TRUE, first row is spread as a new line separator instead of being a column. This helps to reduce the width and allows for clear divisions.
fp_p	paragraph formatting properties associated with row titles, see `officer::fp_par()`.
row_title	a call to `as_paragraph()` - it will be applied to the row titles if any when `spread_first_col=TRUE`.
add_tab	adds a tab in front of "list_data" label lines (located in column `.type`).
...	unused argument

See Also

Other as_flextable methods: `as_flextable.compact_summary()`, `as_flextable.data.frame()`, `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.kmeans()`, `as_flextable.lm()`, `as_flextable.merMod()`, `as_flextable.pam()`, `as_flextable.summarizor()`, `as_flextable.table()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `compact_summary()`

Examples

```
if (require("tables")) {
  set.seed(42)
  genders <- c("Male", "Female")
  status <- c("low", "medium", "high")
  Sex <- factor(sample(genders, 100, rep = TRUE))
  Status <- factor(sample(status, 100, rep = TRUE))
  z <- rnorm(100) + 5
  fmt <- function(x) {
    s <- format(x, digits = 2)
    even <- ((1:length(s)) %% 2) == 0
    s[even] <- sprintf("(%s)", s[even])
    s
  }
  tab <- tabular(
    Justify(c) * Heading() * z *
```

```
      Sex * Heading(Statistic) *
      Format(fmt()) *
      (mean + sd) ~ Status
  )
  as_flextable(tab)
}

if (require("tables")) {
  tab <- tabular(
    (Species + 1) ~ (n = 1) + Format(digits = 2) *
      (Sepal.Length + Sepal.Width) * (mean + sd),
    data = iris
  )
  as_flextable(tab)
}

if (require("tables")) {
  x <- tabular((Factor(gear, "Gears") + 1)
  * ((n = 1) + Percent()
      + (RowPct = Percent("row"))
      + (ColPct = Percent("col")))
  ~ (Factor(carb, "Carburetors") + 1)
    * Format(digits = 1), data = mtcars)

  ft <- as_flextable(
    x,
    spread_first_col = TRUE,
    row_title = as_paragraph(
      colorize("Gears: ", color = "#666666"),
      colorize(as_b(.row_title), color = "red")
    )
  )
  ft
}

if (require("tables")) {
  tab <- tabular(
    (mean + mean) * (Sepal.Length + Sepal.Width) ~ 1,
    data = iris
  )
  as_flextable(tab)
}
```

`as_flextable.tabulator` *Transform a 'tabulator' object into a flextable*

Description

`tabulator()` object can be transformed as a flextable with method `as_flextable()`.

Usage

```
## S3 method for class 'tabulator'
as_flextable(
  x,
  separate_with = character(0),
  big_border = fp_border_default(width = 1.5),
  small_border = fp_border_default(width = 0.75),
  rows_alignment = "left",
  columns_alignment = "center",
  label_rows = x$rows,
  spread_first_col = FALSE,
  expand_single = FALSE,
  sep_w = 0.05,
  unit = "in",
  ...
)
```

Arguments

x	result from tabulator()
separate_with	columns used to sepatate the groups with an horizontal line.
big_border, small_border	big and small border properties defined by a call to fp_border_default() or officer::fp_border().
rows_alignment, columns_alignment	alignments to apply to columns corresponding to rows and columns; see arguments rows and columns in tabulator().
label_rows	labels to use for the first column names, i.e. the *row* column names. It must be a named vector, the values will be matched based on the names.
spread_first_col	if TRUE, first row is spread as a new line separator instead of being a column. This helps to reduce the width and allows for clear divisions.
expand_single	if FALSE (the default), groups with only one row will not be expanded with a title row. If TRUE, single row groups and multi-row groups are all restructured.
sep_w	blank column separators'width to be used. If 0, blank column separators will not be used.
unit	unit of argument sep_w, one of "in", "cm", "mm".
...	unused argument

See Also

summarizor(), as_grouped_data()

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.xtable(), compact_summary()

Examples

```
## Not run:
library(flextable)

set_flextable_defaults(digits = 2, border.color = "gray")

if (require("stats")) {
  dat <- aggregate(breaks ~ wool + tension,
    data = warpbreaks, mean
  )

  cft_1 <- tabulator(
    x = dat,
    rows = "wool",
    columns = "tension",
    `mean` = as_paragraph(as_chunk(breaks)),
    `(N)` = as_paragraph(
      as_chunk(length(breaks))
    )
  )

  ft_1 <- as_flextable(cft_1, sep_w = .1)
  ft_1
}

if (require("stats")) {
  set_flextable_defaults(
    padding = 1, font.size = 9,
    border.color = "orange"
  )

  ft_2 <- as_flextable(cft_1, sep_w = 0)
  ft_2
}

if (require("stats")) {
  set_flextable_defaults(
    padding = 6, font.size = 11,
    border.color = "white",
    font.color = "white",
    background.color = "#333333"
  )

  ft_3 <- as_flextable(
    x = cft_1, sep_w = 0,
    rows_alignment = "center",
    columns_alignment = "right"
  )
  ft_3
}

init_flextable_defaults()
```

```
## End(Not run)
```

`as_flextable.xtable` *Transform a 'xtable' object into a flextable*

Description

Get a `flextable` object from a `xtable` object.

Usage

```
## S3 method for class 'xtable'
as_flextable(
  x,
  text.properties = fp_text_default(),
  format.args = getOption("xtable.format.args", NULL),
  rowname_col = "rowname",
  hline.after = getOption("xtable.hline.after", c(-1, 0, nrow(x))),
  NA.string = getOption("xtable.NA.string", ""),
  include.rownames = TRUE,
  rotate.colnames = getOption("xtable.rotate.colnames", FALSE),
  ...
)
```

Arguments

`x`	`xtable` object
`text.properties`	default text formatting properties
`format.args`	List of arguments for the formatC function. See argument `format.args` of `print.xtable`. Not yet implemented.
`rowname_col`	colname used for row names column
`hline.after`	see `?print.xtable`.
`NA.string`	see `?print.xtable`.
`include.rownames`	see `?print.xtable`.
`rotate.colnames`	see `?print.xtable`.
`...`	unused arguments

See Also

Other as_flextable methods: `as_flextable.compact_summary()`, `as_flextable.data.frame()`, `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.kmeans()`, `as_flextable.lm()`, `as_flextable.merMod()`, `as_flextable.pam()`, `as_flextable.summarizor()`, `as_flextable.table()`, `as_flextable.tabular()`, `as_flextable.tabulator()`, `compact_summary()`

Examples

```
library(officer)
if( require("xtable") ){

  data(tli)
  tli.table <- xtable(tli[1:10, ])
  align(tli.table) <- rep("r", 6)
  align(tli.table) <- "|r|r|clr|r|"
  ft_1 <- as_flextable(
    tli.table,
    rotate.colnames = TRUE,
    include.rownames = FALSE)
  ft_1 <- height(ft_1, i = 1, part = "header", height = 1)
  ft_1


  Grade3 <- c("A","B","B","A","B","C","C","D","A","B",
    "C","C","C","D","B","B","D","C","C","D")
  Grade6 <- c("A","A","A","B","B","B","B","B","C","C",
    "A","C","C","C","D","D","D","D","D","D")
  Cohort <- table(Grade3, Grade6)
  ft_2 <- as_flextable(xtable(Cohort))
  ft_2 <- set_header_labels(ft_2, rowname = "Grade 3")
  ft_2 <- autofit(ft_2)
  ft_2 <- add_header(ft_2, A = "Grade 6")
  ft_2 <- merge_at(ft_2, i = 1, j = seq_len( ncol(Cohort) ) + 1,
    part = "header" )
  ft_2 <- bold(ft_2, j = 1, bold = TRUE, part = "body")
  ft_2 <- height_all(ft_2, part = "header", height = .4)
  ft_2

  temp.ts <- ts(cumsum(1 + round(rnorm(100), 0)),
    start = c(1954, 7), frequency = 12)
  ft_3 <- as_flextable(x = xtable(temp.ts, digits = 0),
    NA.string = "-")
  ft_3

  detach("package:xtable", unload = TRUE)
}
```

---

as_grouped_data *Insert group-label rows into a data frame*

Description

Repeated consecutive values of group columns will be used to define the title of the groups and will be added as a row title.

Usage

```
as_grouped_data(x, groups, columns = NULL, expand_single = TRUE)
```

Arguments

x	dataset
groups	columns names to be used as row separators.
columns	columns names to keep
expand_single	if FALSE, groups with only one row will not be expanded with a title row. If TRUE (the default), single row groups and multi-row groups are all restructured.

See Also

as_flextable.grouped_data()

Examples

```
# as_grouped_data -----
library(data.table)
CO2 <- CO2
setDT(CO2)
CO2$conc <- as.integer(CO2$conc)

data_co2 <- dcast(CO2, Treatment + conc ~ Type,
  value.var = "uptake", fun.aggregate = mean
)
data_co2
data_co2 <- as_grouped_data(x = data_co2, groups = c("Treatment"))
data_co2
```

---

as_highlight *Highlight chunk*

Description

The function is producing a chunk with an highlight chunk.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_highlight(x, color)
```

Arguments

x	value, if a chunk, the chunk will be updated
color	color to use as text highlighting color as character vector.

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_i(), as_image(), as_qmd(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Sepal.Length", "dummy")
)

ft <- compose(ft,
  j = "dummy",
  value = as_paragraph(as_highlight(Sepal.Length, color = "yellow"))
)

ft
```

---

as_i *Italic chunk*

Description

The function is producing a chunk with italic font.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_i(x)
```

Arguments

x	value, if a chunk, the chunk will be updated

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_image(), as_qmd(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Sepal.Length", "dummy")
)

ft <- compose(ft,
```

```
  j = "dummy",
  value = as_paragraph(as_i(Sepal.Length))
)

ft
```

---

as_image *Image chunk*

Description

The function lets add images within flextable objects with functions:

- compose() and as_paragraph(),
- append_chunks(),
- prepend_chunks()

Usage

```
as_image(
  src,
  width = NULL,
  height = NULL,
  unit = "in",
  guess_size = TRUE,
  alt = "",
  ...
)
```

Arguments

src	image filename
width, height	size of the image file. It can be ignored if parameter guess_size=TRUE, see parameter guess_size.
unit	unit for width and height, one of "in", "cm", "mm".
guess_size	If TRUE and width or height is not provided, the size of the image is read from the file, one pixel being 1/72 inch. The size of 'png', 'jpeg' and 'gif' files is read from their header, which is fast; other formats, such as 'svg', require package 'magick'.
alt	alternative text for the image (used for accessibility)
...	unused argument

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

`compose()`, `as_paragraph()`

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
img.file <- file.path(
  R.home("doc"),
  "html", "logo.jpg"
)
myft <- flextable(head(iris))
myft <- compose(myft,
  i = 1:3, j = 1,
  value = as_paragraph(
    as_image(src = img.file),
    " ",
    as_chunk(Sepal.Length,
      props = fp_text_default(color = "red")
    )
  ),
  part = "body"
)
ft <- autofit(myft)
ft
```

---

as_paragraph *Build a paragraph from chunks*

Description

`as_paragraph()` assembles one or more chunks into a single paragraph that defines the content of a flextable cell. Each cell in a flextable contains exactly one paragraph; a paragraph is an ordered sequence of chunks.

Chunks are the smallest content units and can be created with `as_chunk()` (formatted text), `as_b()` / `as_i()` (bold / italic shortcuts), `minibar()` (inline bar), `as_image()` (image), `gg_chunk()` (ggplot), `as_equation()` (equation) or `hyperlink_text()` (link). Plain character strings passed to `as_paragraph()` are automatically converted to chunks via `as_chunk()`.

The resulting paragraph is passed to the value argument of compose(), mk_par(), add_header_lines(), add_footer_lines() or footnote() to set cell content.

Usage

```
as_paragraph(..., list_values = NULL)
```

Arguments

...	chunk elements that are defining the paragraph content. If a character is used, it is transformed to a chunk object with function as_chunk().
list_values	a list of chunk elements that are defining the paragraph content. If specified argument ... is unused.

See Also

as_chunk(), minibar(), as_image(), hyperlink_text()

Other functions to compose cell content: append_chunks(), compose(), footnote(), labelizor(), prepend_chunks(), void()

Examples

```
library(flextable)
ft <- flextable(airquality[sample.int(150, size = 10), ])
ft <- compose(ft,
  j = "Wind",
  value = as_paragraph(
    as_chunk(Wind, props = fp_text_default(color = "orange")),
    " ",
   minibar(value = Wind, max = max(airquality$Wind), barcol = "orange", bg = "black", height = .15)
  ),
  part = "body"
)
ft <- autofit(ft)
ft
```

---

as_qmd *Quarto inline markdown chunk*

Description

as_qmd() creates a chunk for inline Quarto markdown content (text-level) that fits within a table cell paragraph. This enables cross-references (@fig-xxx, @tbl-xxx), links, bold/italic, math, inline code, shortcodes and other inline Quarto markdown features inside flextable cells.

It is not designed for block-level elements such as headings, bullet lists or fenced code blocks.

The chunk is used with compose(), append_chunks() or prepend_chunks(). It requires the flextable-qmd Lua filter extension (see use_flextable_qmd()) and works with HTML, PDF, Typst and Word (docx) Quarto output formats.

Usage

```
as_qmd(x, display = x)
```

Arguments

x	character vector of Quarto markdown content.
display	character vector of display text used as fallback when the Lua filter is not active. Defaults to x.

Setup

1. Install the extension once per project:

```
flextable::use_flextable_qmd()
```

1. Add the filter to your Quarto document YAML. For HTML and PDF, a single line is enough:

```
filters:
  - flextable-qmd
```

For Word (docx), an additional post-render filter removes the wrapper table that Quarto adds around labelled flextables:

```
filters:
  - flextable-qmd
  - at: post-render
    path: _extensions/flextable-qmd/unwrap-float.lua
```

Supported markdown

- Cross-references: `@fig-xxx`, `@tbl-xxx`
- Bold / italic: `**bold**`, `*italic*`
- Inline code: `` `code` ``
- Links: `[text](url)` (internal and external)
- Math: `$\\alpha + \\beta$`
- Shortcodes and other Quarto markdown constructs

Limitations

Each table cell in a flextable contains a single paragraph built from inline chunks (see `as_paragraph()`). There is no mechanism to insert block-level structures (multiple paragraphs, lists, headings, fenced code blocks, callouts, etc.) inside a cell. Because `as_qmd()` produces one of these inline chunks, only inline markdown is supported.

See Also

use_flextable_qmd() to install the Lua filter extension, knit_print.flextable() for rendering options in knitr documents.

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
library(flextable)

dat <- data.frame(
  label = c("Bold", "Link", "Code"),
  content = c(
    "This is **bold** text",
    "Visit [Quarto](https://quarto.org)",
    "Use `print()` here"
  )
)
ft <- flextable(dat)
ft <- mk_par(ft, j = "content",
  value = as_paragraph(as_qmd(content)))
ft
```

---

as_strike *Strikethrough chunk*

Description

The function is producing a chunk with strikethrough font.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_strike(x)
```

Arguments

x	value, if a chunk, the chunk will be updated

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Sepal.Length", "dummy")
)

ft <- compose(ft,
  j = "dummy",
  value = as_paragraph(
    as_strike(Sepal.Length)
  )
)

ft
```

---

as_sub *Subscript chunk*

Description

The function is producing a chunk with subscript vertical alignment.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_sub(x)
```

Arguments

x	value, if a chunk, the chunk will be updated

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_strike(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris), col_keys = c("dummy"))

ft <- compose(ft,
  i = 1, j = "dummy", part = "header",
  value = as_paragraph(
    as_sub("Sepal.Length"),
    " anything "
  )
)
```

```
ft <- autofit(ft)
ft
```

---

as_sup *Superscript chunk*

Description

The function is producing a chunk with superscript vertical alignment.

It is used to add it to the content of a cell of the flextable with the functions compose(), append_chunks() or prepend_chunks().

Usage

```
as_sup(x)
```

Arguments

x	value, if a chunk, the chunk will be updated

Note

This is a sugar function that ease the composition of complex labels made of different formattings. It should be used inside a call to as_paragraph().

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_strike(), as_sub(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()

Examples

```
ft <- flextable(head(iris), col_keys = c("dummy"))

ft <- compose(ft,
  i = 1, j = "dummy", part = "header",
  value = as_paragraph(
    " anything ",
    as_sup("Sepal.Width")
  )
)

ft <- autofit(ft)
ft
```

as_word_field *Word dynamic field chunk*

Description

`as_word_field()` inserts a Word field code (e.g. page numbers, dates, cross-references) as a chunk inside a flextable cell. Field codes are Word's mechanism for auto-computed values; see Microsoft's field-code reference for the available codes.

The chunk is used with `compose()`, `append_chunks()` or `prepend_chunks()`. It only has an effect in Word (docx) output; other formats ignore it. To apply it conditionally, use the post-processing step (see `set_flextable_defaults(post_process_docx = ...)`).

Important: fields are inserted but not computed. After opening the document in Word, select all text and press `F9` (on macOS: `Fn` + `F9`) to refresh the field values.

Usage

```
as_word_field(x, props = NULL, width = 0.1, height = 0.15, unit = "in")
```

Arguments

`x`	computed field strings
`props`	text properties (see `fp_text_default()` or `officer::fp_text()`) object to be used to format the text. If not specified, it will use the default text properties of the cell(s).
`width, height`	size computed field
`unit`	unit for width and height, one of "in", "cm", "mm".

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
library(flextable)

# define some default values ----
set_flextable_defaults(font.size = 22, border.color = "gray")

# an example with append_chunks ----
pp_docx <- function(x) {
  x <- add_header_lines(x, "Page ")
  x <- append_chunks(
    x = x, i = 1, part = "header", j = 1,
    as_word_field(x = "Page")
  )
```

```
  align(x, part = "header", align = "left")
}
ft_1 <- flextable(cars)
ft_1 <- autofit(ft_1)
ft_1 <- pp_docx(ft_1)

## or:
# set_flextable_defaults(post_process_docx = pp_docx)
## to prevent this line addition when output is not docx

# print(ft_1, preview = "docx")

# an example with compose ----

library(officer)
ft_2 <- flextable(head(cars))
ft_2 <- add_footer_lines(ft_2, "temp text")
ft_2 <- compose(
  x = ft_2, part = "footer", i = 1, j = 1,
  as_paragraph(
    "p. ",
    as_word_field(x = "Page", width = .05),
    " on ", as_word_field(x = "NumPages", width = .05)
  )
)
ft_2 <- autofit(ft_2, part = c("header", "body"))

doc <- read_docx()
doc <- body_add_flextable(doc, ft_2)
doc <- body_add_break(doc)
doc <- body_add_flextable(doc, ft_2)
outfile <- print(doc, target = tempfile(fileext = ".docx"))

# reset default values ----
init_flextable_defaults()
```

`autofit` *Adjust columns to their content size*

Description

Compute and apply the minimum widths and heights needed to display each cell's content on a single line, with an optional extra margin (`add_w`, `add_h`).

This function sizes columns to fit their content. It does not constrain the table to a given total width. To enforce a maximum width, use `fit_columns()` (wraps text) or `fit_to_width()` (shrinks font size).

Note that this function is not related to 'Microsoft Word' *Autofit* feature.

There is an alternative to fixed-width layouts that works well with HTML and Word output that can be set with `set_table_properties(layout = "autofit")`, see `set_table_properties()`.

Usage

```
autofit(
  x,
  add_w = 0.1,
  add_h = 0.1,
  part = c("body", "header"),
  unit = "in",
  hspans = "none"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
add_w	extra width to add in inches
add_h	extra height to add in inches
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' can be used.
unit	unit for add_h and add_w, one of "in", "cm", "mm".
hspans	specifies how cells that are horizontally are included in the calculation. It must be one of the following values "none", "divided" or "included". If "none", widths of horizontally spanned cells is set to 0 (then do not affect the widths); if "divided", widths of horizontally spanned cells is divided by the number of spanned cells; if "included", all widths (included horizontally spanned cells) will be used in the calculation.

See Also

Other functions for flextable size management: `dim.flextable()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

```
ft_1 <- flextable(head(mtcars))
ft_1
ft_2 <- autofit(ft_1)
ft_2
```

---

before *Detect rows before a given value*

Description

Returns a logical vector indicating which elements of x appear before the first occurrence of any of the `entries` values. Useful as a row selector in `hline()` to insert a border above a summary row such as `"Total"`.

Usage

```
before(x, entries)
```

Arguments

x	an atomic vector of values to be tested
entries	a sequence of items to be searched in x.

See Also

hline()

Examples

```
library(flextable)
library(officer)

dat <- data.frame(
  stringsAsFactors = FALSE,
  check.names = FALSE,
  Level = c("setosa", "versicolor", "virginica", "<NA>", "Total"),
  Freq = as.integer(c(50, 50, 50, 0, 150)),
  `% Valid` = c(
    100 / 3,
    100 / 3, 100 / 3, NA, 100
  ),
  `% Valid Cum.` = c(100 / 3, 100 * 2 / 3, 100, NA, 100),
  `% Total` = c(
    100 / 3,
    100 / 3, 100 / 3, 0, 100
  ),
  `% Total Cum.` = c(
    100 / 3,
    100 * 2 / 3, 100, 100, 100
  )
)

ft <- flextable(dat)
ft <- hline(ft,
  i = ~ before(Level, "Total"),
  border = fp_border_default(width = 2)
)
ft
```

---

bg *Set background color*

Description

Change the background color of selected rows and columns of a flextable. A function can be used instead of fixed colors.

When bg is a function, it is possible to color cells based on values located in other columns; using hidden columns (those not used by argument `colkeys`) is a common use case. The argument `source` must be used to define the columns to be used for the color definition, and the argument `j` must be used to define where to apply the colors and only accepts values from `colkeys`.

Usage

```
bg(x, i = NULL, j = NULL, bg, part = "body", source = j)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* `i` *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* `j` *parameter* in <Selectors in flextable>.
bg	color to use as background color. If a function, the function must return a character vector of colors.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' can be used.
source	if bg is a function, source specifies the dataset column to be used as an argument to bg. This is only useful when j is colored with values contained in other columns.

Note

Word does not allow you to apply transparency to table cells or paragraph shading.

See Also

Other formatting shortcuts: `align()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
ft_1 <- flextable(head(mtcars))
ft_1 <- bg(ft_1, bg = "wheat", part = "header")
ft_1 <- bg(ft_1, i = ~ qsec < 18, bg = "#EFEFEF", part = "body")
ft_1 <- bg(ft_1, j = "drat", bg = "#606060", part = "all")
ft_1 <- color(ft_1, j = "drat", color = "white", part = "all")
ft_1

if (require("scales")) {
```

```
  ft_2 <- flextable(head(iris))
  colourer <- col_numeric(
    palette = c("wheat", "red"),
    domain = c(0, 7)
  )
  ft_2 <- bg(ft_2,
    j = c(
      "Sepal.Length", "Sepal.Width",
      "Petal.Length", "Petal.Width"
    ),
    bg = colourer, part = "body"
  )
  ft_2
}
```

`body_add_flextable` *Add flextable into a Word document*

Description

Add a flextable into a Word document created with 'officer'.

Usage

```
body_add_flextable(
  x,
  value,
  align = NULL,
  pos = "after",
  split = NULL,
  topcaption = TRUE,
  keepnext = NULL
)
```

Arguments

x	an rdocx object
value	flextable object
align	left, center (default) or right. The align parameter is still supported for the time being, but we recommend using set_flextable_defaults(table_align = "center") instead that will set this default alignment for all flextables during the R session, or to define alignement for each table with set_table_properties(align = "center").
pos	where to add the flextable relative to the cursor, one of "after", "before", "on" (end of line).

split	set to TRUE if you want to activate Word option 'Allow row to break across pages'. This argument is still supported for the time being, but we recommend using `set_flextable_defaults(split = TRUE)` instead that will set this as default setting for all flextables during the R session, or to define alignement for each table with `set_table_properties()` with argument `opts_word=list(split = TRUE)` instead.
topcaption	if TRUE caption is added before the table, if FALSE, caption is added after the table.
keepnext	Defunct in favor of `paginate()`. The default value used for `keep_with_next` is set with `set_flextable_defaults(keep_with_next = TRUE)`.

Details

Use the `paginate()` function to define whether the table should be displayed on one or more pages, and whether the header should be displayed with the first lines of the table body on the same page.

Use the `set_caption()` function to define formatted captions (with `as_paragraph()`) or simple captions (with a string). `topcaption` can be used to insert the caption before the table (default) or after the table (use `FALSE`).

See Also

`knit_print.flextable()`, `save_as_docx()`

Other functions for officer integration: `body_replace_flextable_at_bkm()`, `ph_with.flextable()`, `rtf_add.flextable()`

Examples

```
library(officer)

# define global settings
set_flextable_defaults(
  split = TRUE,
  table_align = "center",
  table.layout = "autofit"
)

# produce 3 flextable
ft_1 <- flextable(head(airquality, n = 20))
ft_1 <- color(ft_1, i = ~ Temp > 70, color = "red", j = "Temp")
ft_1 <- highlight(ft_1, i = ~ Wind < 8, color = "yellow", j = "Wind")
ft_1 <- set_caption(
  x = ft_1,
  autonum = run_autonum(seq_id = "tab"),
  caption = "Daily air quality measurements"
)
ft_1 <- paginate(ft_1, init = TRUE, hdr_ftr = TRUE)

ft_2 <- proc_freq(mtcars, "vs", "gear")
ft_2 <- set_caption(
  x = ft_2,
```

```
  autonum = run_autonum(seq_id = "tab", bkm = "mtcars"),
  caption = as_paragraph(
    as_b("mtcars"), " ",
    colorize("table", color = "orange")
  ),
  fp_p = fp_par(keep_with_next = TRUE)
)
ft_2 <- paginate(ft_2, init = TRUE, hdr_ftr = TRUE)

ft_3 <- summarizor(iris, by = "Species")
ft_3 <- as_flextable(ft_3, spread_first_col = TRUE)
ft_3 <- set_caption(
  x = ft_3,
  autonum = run_autonum(seq_id = "tab"),
  caption = "iris summary"
)
ft_3 <- paginate(ft_3, init = TRUE, hdr_ftr = TRUE)

# add the 3 flextable in a new Word document
doc <- read_docx()
doc <- body_add_flextable(doc, value = ft_1)
doc <- body_add_par(doc, value = "")
doc <- body_add_flextable(doc, value = ft_2)
doc <- body_add_par(doc, value = "")
doc <- body_add_flextable(doc, value = ft_3)

fileout <- tempfile(fileext = ".docx")
print(doc, target = fileout)
```

`body_replace_flextable_at_bkm` *Add flextable at bookmark location in a Word document*

Description

Use this function if you want to replace a paragraph containing a bookmark with a flextable. As a side effect, the bookmark will be lost.

Usage

```
body_replace_flextable_at_bkm(
  x,
  bookmark,
  value,
  align = "center",
  split = FALSE
)
```

Arguments

x	an rdocx object
bookmark	bookmark id
value	flextable object
align	left, center (default) or right.
split	set to TRUE if you want to activate Word option 'Allow row to break across pages'.

See Also

Other functions for officer integration: body_add_flextable(), ph_with.flextable(), rtf_add.flextable()

---

bold *Set bold font*

Description

Change the font weight of selected rows and columns of a flextable.

Usage

```
bold(x, i = NULL, j = NULL, bold = TRUE, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
bold	boolean value
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), color(), empty_blanks(), font(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
ft <- flextable(head(iris))
ft <- bold(ft, bold = TRUE, part = "header")
```

border_inner *Set all inner borders*

Description

The function is applying a vertical and horizontal borders to inner content of one or all parts of a flextable.

Usage

```
border_inner(x, border = NULL, part = "all")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
border	border properties defined by a call to `officer::fp_border()`
part	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline()`, `vline_left()`, `vline_right()`

Examples

```
library(officer)
std_border <- fp_border(color = "orange", width = 1)

dat <- iris[c(1:5, 51:55, 101:105), ]
ft <- flextable(dat)
ft <- border_remove(x = ft)

# add inner vertical borders
ft <- border_inner(ft, border = std_border)
ft
```

border_inner_h *Set inner horizontal borders*

Description

The function is applying a border to inner content of one or all parts of a flextable.

Usage

```
border_inner_h(x, border = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
border	border properties defined by a call to officer::fp_border()
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other borders management: border_inner(), border_inner_v(), border_outer(), border_remove(), hline(), hline_bottom(), hline_top(), surround(), vline(), vline_left(), vline_right()

Examples

```
library(officer)
std_border <- fp_border(color = "orange", width = 1)

dat <- iris[c(1:5, 51:55, 101:105), ]
ft <- flextable(dat)
ft <- border_remove(x = ft)

# add inner horizontal borders
ft <- border_inner_h(ft, border = std_border)
ft
```

---

border_inner_v *Set inner vertical borders*

Description

The function is applying a vertical border to inner content of one or all parts of a flextable.

Usage

```
border_inner_v(x, border = NULL, part = "all")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
border	border properties defined by a call to officer::fp_border()
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other borders management: border_inner(), border_inner_h(), border_outer(), border_remove(), hline(), hline_bottom(), hline_top(), surround(), vline(), vline_left(), vline_right()

Examples

```
library(officer)
std_border <- fp_border(color = "orange", width = 1)

dat <- iris[c(1:5, 51:55, 101:105), ]
ft <- flextable(dat)
ft <- border_remove(x = ft)

# add inner vertical borders
ft <- border_inner_v(ft, border = std_border)
ft
```

---

border_outer *Set outer borders*

Description

The function is applying a border to outer cells of one or all parts of a flextable.

Usage

```
border_outer(x, border = NULL, part = "all")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
border	border properties defined by a call to officer::fp_border()
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other borders management: border_inner(), border_inner_h(), border_inner_v(), border_remove(), hline(), hline_bottom(), hline_top(), surround(), vline(), vline_left(), vline_right()

Examples

```
library(officer)
big_border <- fp_border(color = "red", width = 2)

dat <- iris[c(1:5, 51:55, 101:105), ]
ft <- flextable(dat)
ft <- border_remove(x = ft)

# add outer borders
ft <- border_outer(ft, part = "all", border = big_border)
ft
```

`border_remove` *Remove borders*

Description

The function is deleting all borders of the flextable object.

Usage

```
border_remove(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `hline()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline()`, `vline_left()`, `vline_right()`

Examples

```
dat <- iris[c(1:5, 51:55, 101:105), ]
ft_1 <- flextable(dat)
ft_1 <- theme_box(ft_1)
ft_1

# remove all borders
ft_2 <- border_remove(x = ft_1)
ft_2
```

colformat_char *Format character cells*

Description

Format character cells in a flextable.

Usage

```
colformat_char(
  x,
  i = NULL,
  j = NULL,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
na_str, nan_str	string to be used for NA and NaN values
prefix, suffix	string to be used as prefix or suffix

See Also

Other cells formatters: colformat_date(), colformat_datetime(), colformat_double(), colformat_image(), colformat_int(), colformat_lgl(), colformat_num(), set_formatter()

Examples

```
dat <- iris
z <- flextable(head(dat))
ft <- colformat_char(
  x = z, j = "Species", suffix = "!"
)
z <- autofit(z)
z
```

`colformat_date` *Format date cells*

Description

Format date cells in a flextable.

Usage

```
colformat_date(
  x,
  i = NULL,
  j = NULL,
  fmt_date = get_flextable_defaults()$fmt_date,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`fmt_date`	see `strptime()`
`na_str, nan_str`	string to be used for NA and NaN values
`prefix, suffix`	string to be used as prefix or suffix

See Also

Other cells formatters: `colformat_char()`, `colformat_datetime()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

```
dat <- data.frame(
  z = Sys.Date() + 1:3,
  w = Sys.Date() - 1:3
)
ft <- flextable(dat)
ft <- colformat_date(x = ft)
ft <- autofit(ft)
ft
```

`colformat_datetime` *Format datetime cells*

Description

Format datetime cells in a flextable.

Usage

```
colformat_datetime(
  x,
  i = NULL,
  j = NULL,
  fmt_datetime = get_flextable_defaults()$fmt_datetime,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`fmt_datetime`	see `strptime()`
`na_str, nan_str`	string to be used for NA and NaN values
`prefix, suffix`	string to be used as prefix or suffix

See Also

Other cells formatters: `colformat_char()`, `colformat_date()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

```
dat <- data.frame(
  z = Sys.time() + (1:3) * 24,
  w = Sys.Date() - (1:3) * 24
)
ft <- flextable(dat)
ft <- colformat_datetime(x = ft)
ft <- autofit(ft)
ft
```

colformat_double *Format double cells*

Description

Format numeric cells in a flextable using `formatC()` with explicit control over digits and decimal mark.

Usage

```
colformat_double(
  x,
  i = NULL,
  j = NULL,
  big.mark = get_flextable_defaults()$big.mark,
  decimal.mark = get_flextable_defaults()$decimal.mark,
  digits = get_flextable_defaults()$digits,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
j	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
big.mark, digits, decimal.mark	see `formatC()`
na_str, nan_str	string to be used for NA and NaN values
prefix, suffix	string to be used as prefix or suffix

See Also

Other cells formatters: `colformat_char()`, `colformat_date()`, `colformat_datetime()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

```
dat <- mtcars
ft <- flextable(head(dat))
ft <- colformat_double(
```

```
  x = ft,
  big.mark = ",", digits = 2, na_str = "N/A"
)
autofit(ft)
```

`colformat_image` *Format cells as images*

Description

Format image paths as images in a flextable.

Usage

```
colformat_image(
  x,
  i = NULL,
  j = NULL,
  width,
  height,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = "",
  alt = ""
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`width, height`	size of the png file in inches
`na_str, nan_str`	string to be used for NA and NaN values
`prefix, suffix`	string to be used as prefix or suffix
`alt`	alternative text for the images (used for accessibility)

See Also

Other cells formatters: `colformat_char()`, `colformat_date()`, `colformat_datetime()`, `colformat_double()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

```
img.file <- file.path(R.home("doc"), "html", "logo.jpg")

dat <- head(iris)
dat$Species <- as.character(dat$Species)
dat[c(1, 3, 5), "Species"] <- img.file

myft <- flextable(dat)
myft <- colformat_image(
  myft,
  i = c(1, 3, 5),
  j = "Species", width = .20, height = .15
)
ft <- autofit(myft)
ft
```

---

colformat_int *Format integer cells*

Description

Format integer cells in a flextable.

Usage

```
colformat_int(
  x,
  i = NULL,
  j = NULL,
  big.mark = get_flextable_defaults()$big.mark,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
big.mark	see format()
na_str, nan_str	string to be used for NA and NaN values
prefix, suffix	string to be used as prefix or suffix

See Also

Other cells formatters: colformat_char(), colformat_date(), colformat_datetime(), colformat_double(), colformat_image(), colformat_lgl(), colformat_num(), set_formatter()

Examples

```
z <- flextable(head(mtcars))
j <- c("vs", "am", "gear", "carb")
z <- colformat_int(x = z, j = j, prefix = "# ")
z
```

---

colformat_lgl *Format logical cells*

Description

Format logical cells in a flextable.

Usage

```
colformat_lgl(
  x,
  i = NULL,
  j = NULL,
  true = "true",
  false = "false",
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = ""
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
false, true	string to be used for logical
na_str, nan_str	string to be used for NA and NaN values
prefix, suffix	string to be used as prefix or suffix

See Also

Other cells formatters: colformat_char(), colformat_date(), colformat_datetime(), colformat_double(), colformat_image(), colformat_int(), colformat_num(), set_formatter()

Examples

```
dat <- data.frame(a = c(TRUE, FALSE), b = c(FALSE, TRUE))

z <- flextable(dat)
z <- colformat_lgl(x = z, j = c("a", "b"))
autofit(z)
```

---

colformat_num *Format numeric cells with format()*

Description

Format numeric cells in a flextable using R's format() function.

The function is different from colformat_double() on numeric type columns. The function uses the format() function of R on numeric type columns. So this is normally what you see on the R console most of the time (but scientific mode is disabled and NA are replaced).

Usage

```
colformat_num(
  x,
  i = NULL,
  j = NULL,
  big.mark = get_flextable_defaults()$big.mark,
  decimal.mark = get_flextable_defaults()$decimal.mark,
  na_str = get_flextable_defaults()$na_str,
  nan_str = get_flextable_defaults()$nan_str,
  prefix = "",
  suffix = "",
  ...
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.

big.mark, decimal.mark
: see format()

na_str, nan_str
: string to be used for NA and NaN values

prefix, suffix
: string to be used as prefix or suffix

...
: additional argument for function format(), scientific and digits can not be used.

format call

Function format() is called with the following values:

- trim is set to TRUE,
- scientific is set to FALSE,
- big.mark is set to the value of big.mark argument,
- decimal.mark is set to the value of decimal.mark argument,
- other arguments are passed 'as is' to the format function.

argument digits is ignored as it is not the same digits that users want, this one will be used by format() and not formatC(). To change the digit argument use options(digits=4) instead.

This argument will not be changed because colformat_num() is supposed to format things roughly as what you see on the R console.

If these functions does not fit your needs, use set_formatter() that lets you use any format function.

See Also

Other cells formatters: colformat_char(), colformat_date(), colformat_datetime(), colformat_double(), colformat_image(), colformat_int(), colformat_lgl(), set_formatter()

Examples

```
dat <- mtcars
dat[2, 1] <- NA
ft <- flextable(head(dat))
ft <- colformat_num(
  x = ft,
  big.mark = " ", decimal.mark = ",",
  na_str = "N/A"
)
ft <- autofit(ft)
ft
```

color *Set font color*

Description

Change the text color of selected rows and columns of a flextable. A function can be used instead of fixed colors.

When `color` is a function, it is possible to color cells based on values located in other columns; using hidden columns (those not used by argument `colkeys`) is a common use case. The argument `source` must be used to define the columns to be used for the color definition, and the argument `j` must be used to define where to apply the colors and only accepts values from `colkeys`.

Usage

```
color(x, i = NULL, j = NULL, color, part = "body", source = j)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`color`	color to use as font color. If a function, the function must return a character vector of colors.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.
`source`	if color is a function, source specifies the dataset column to be used as an argument to `color`. This is only useful when j is colored with values contained in other columns.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
ft <- flextable(head(mtcars))
ft <- color(ft, color = "orange", part = "header")
ft <- color(ft,
  color = "red",
  i = ~ qsec < 18 & vs < 1
)
```

```
ft

if (require("scales")) {
  scale <- scales::col_numeric(domain = c(-1, 1), palette = "RdBu")
  x <- as.data.frame(cor(iris[-5]))
  x <- cbind(
    data.frame(
      colname = colnames(x),
      stringsAsFactors = FALSE
    ),
    x
  )
}
```

`colorize` *Colorize chunk*

Description

The function is producing a chunk with a font in color.

It is used to add it to the content of a cell of the flextable with the functions `compose()`, `append_chunks()` or `prepend_chunks()`.

Usage

```
colorize(x, color)
```

Arguments

`x`	value, if a chunk, the chunk will be updated
`color`	color to use as text highlighting color as character vector.

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
ft <- flextable(head(iris),
  col_keys = c("Sepal.Length", "dummy")
)

ft <- compose(ft,
  j = "dummy",
  value = as_paragraph(colorize(Sepal.Length, color = "red"))
)

ft
```

compact_summary *Compact Summary of a Dataset*

Description

Produces a compact summary of a data.frame. Each row of the result describes one column of the input with a type-specific synopsis.

Supported types and their summaries:

- **numeric / integer**: count of non-NA values, min and max.
- **character**: number of unique values, first values listed.
- **factor**: number of levels, levels listed.
- **logical**: count of non-NA values, counts of TRUE and FALSE.
- **Date**: count of non-NA values, date range.
- **POSIXct / POSIXlt**: count of non-NA values, datetime range.
- **hms / difftime**: count of non-NA values, time range.

Character and factor columns share the same summary layout but report a different type label.

The result has class "compact_summary" and can be converted into a flextable with as_flextable().

Usage

```
compact_summary(x, show_type = FALSE, show_na = FALSE, max_levels = 10L)
```

Arguments

x	A data.frame.
show_type	If TRUE, a *Type* column is added when the object is rendered as a flextable.
show_na	If TRUE, a *NA* column showing the count of missing values per column is added when rendered as a flextable.
max_levels	Maximum number of levels or unique values displayed for factor and character columns. Additional values are replaced by ", ...".

Value

A data.frame with additional class "compact_summary".

See Also

as_flextable.compact_summary()

Other as_flextable methods: as_flextable.compact_summary(), as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable()

Examples

```
z <- compact_summary(iris)
as_flextable(z)

z <- compact_summary(iris, show_type = TRUE, show_na = TRUE)
as_flextable(z)
```

---

compose *Set cell content from paragraph chunks*

Description

Modify flextable displayed values with eventually mixed content paragraphs.

Function is handling complex formatting as image insertion with `as_image()`, superscript with `as_sup()`, formated text with `as_chunk()` and several other *chunk* functions.

Function `mk_par` is another name for `compose` as there is an unwanted **conflict with package 'purrr'**.

If you only need to add some content at the end or the beginning of paragraphs and keep existing content as it is, functions `append_chunks()` and `prepend_chunks()` should be prefered.

Usage

```
compose(x, i = NULL, j = NULL, value, part = "body", use_dot = FALSE)

mk_par(x, i = NULL, j = NULL, value, part = "body", use_dot = FALSE)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
j	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
value	a call to function `as_paragraph()`.
part	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.
use_dot	by default `use_dot=FALSE`; if `use_dot=TRUE`, `value` is evaluated within a data.frame augmented of a column named `.` containing the `j`th column.

See Also

`fp_text_default()`, `as_chunk()`, `as_b()`, `as_word_field()`, `labelizor()`

Other functions to compose cell content: `append_chunks()`, `as_paragraph()`, `footnote()`, `labelizor()`, `prepend_chunks()`, `void()`

Examples

```
ft_1 <- flextable(head(cars, n = 5), col_keys = c("speed", "dist", "comment"))
ft_1 <- mk_par(
  x = ft_1, j = "comment",
  i = ~ dist > 9,
  value = as_paragraph(
    colorize(as_i("speed: "), color = "gray"),
    as_sup(sprintf("%.0f", speed))
  )
)
ft_1 <- set_table_properties(ft_1, layout = "autofit")
ft_1

# using `use_dot = TRUE` ----
set.seed(8)
dat <- iris[sample.int(n = 150, size = 10), ]
dat <- dat[order(dat$Species), ]


ft_2 <- flextable(dat)
ft_2 <- mk_par(ft_2,
  j = ~ . - Species,
  value = as_paragraph(
    minibar(.,
      barcol = "white",
      height = .1
    )
  ), use_dot = TRUE
)
ft_2 <- theme_vader(ft_2)
ft_2 <- autofit(ft_2)
ft_2
```

`continuous_summary` *Summarize continuous variables as a flextable*

Description

create a data.frame summary for continuous variables

Usage

```
continuous_summary(
  dat,
  columns = NULL,
  by = character(0),
  hide_grouplabel = TRUE,
  digits = 3
)
```

Arguments

dat	a data.frame
columns	continuous variables to be summarized. If NULL all continuous variables are summarized.
by	discrete variables to use as groups when summarizing.
hide_grouplabel	if TRUE, group label will not be rendered, only level/value will be rendered.
digits	the desired number of digits after the decimal point

Examples

```
ft_1 <- continuous_summary(iris, names(iris)[1:4],
  by = "Species",
  hide_grouplabel = FALSE
)
ft_1
```

---

delete_columns *Delete flextable columns*

Description

The function removes one or more columns from a 'flextable'.

Usage

```
delete_columns(x, j = NULL)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.

Details

Deleting one or more columns will result in the deletion of any span parameters that may have been set previously. They will have to be redone after this operation or performed only after this deletion.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
ft <- flextable(head(iris))
ft <- delete_columns(ft, j = "Species")
ft
```

`delete_part` *Delete flextable part*

Description

indicate to not print a part of the flextable, i.e. an header, footer or the body.

Usage

```
delete_part(x, part = "header")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.

See Also

Other row and column operations: `add_body()`, `add_body_row()`, `add_footer()`, `add_footer_lines()`, `add_footer_row()`, `add_header()`, `add_header_lines()`, `add_header_row()`, `delete_columns()`, `delete_rows()`, `paginate()`, `separate_header()`, `set_header_footer_df`, `set_header_labels()`, `split_columns()`, `split_rows()`, `split_to_pages()`

Examples

```
ft <- flextable(head(iris))
ft <- delete_part(x = ft, part = "header")
ft
```

delete_rows *Delete flextable rows*

Description

The function removes one or more rows from a 'flextable'.

Usage

```
delete_rows(x, i = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

Details

Deleting one or more rows will result in the deletion of any span parameters that may have been set previously. They will have to be redone after this operation or performed only after this deletion.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
ft <- flextable(head(iris))
ft <- delete_rows(ft, i = 1:5, part = "body")
ft
```

`df_printer` *data.frame automatic printing as a flextable*

Description

Create a summary from a data.frame as a flextable. This function is to be used in an R Markdown document.

To use that function, you must declare it in the part `df_print` of the 'YAML' header of your R Markdown document:

```
---
df_print: !expr function(x) flextable::df_printer(x)
---
```

We notice an unexpected behavior with bookdown. When using bookdown it is necessary to use `use_df_printer()` instead in a setup run chunk:

```
use_df_printer()
```

Usage

```
df_printer(dat, ...)
```

Arguments

`dat`	the data.frame
`...`	unused argument

Details

'knitr' chunk options are available to customize the output:

- `ft_max_row`: The number of rows to print. Default to 10.
- `ft_split_colnames`: Should the column names be split (with non alpha-numeric characters). Default to FALSE.
- `ft_short_strings`: Should the character column be shorten. Default to FALSE.
- `ft_short_size`: Maximum length of character column if `ft_short_strings` is TRUE. Default to 35.
- `ft_short_suffix`: Suffix to add when character values are shorten. Default to "...".
- `ft_do_autofit`: Use autofit() before rendering the table. Default to TRUE.
- `ft_show_coltype`: Show column types. Default to TRUE.
- `ft_color_coltype`: Color to use for column types. Default to "#999999".

See Also

Other functions for flextable output and export: `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`, `wrap_flextable()`

Examples

```
df_printer(head(mtcars))
```

---

dim.flextable *Get column widths and row heights of a flextable*

Description

returns widths and heights for each table columns and rows. Values are expressed in inches.

Usage

```
## S3 method for class 'flextable'
dim(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

See Also

Other functions for flextable size management: `autofit()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

```
ftab <- flextable(head(iris))
dim(ftab)
```

dim.flextableGrob *Get optimal width and height of a flextable grob*

Description

returns the optimal width and height for the grob, according to the grob generation parameters.

Usage

```
## S3 method for class 'flextableGrob'
dim(x)
```

Arguments

x	a flextableGrob object

Value

a named list with two elements, `width` and `height`. Values are expressed in inches.

Examples

```
ftab <- flextable(head(iris))
gr <- gen_grob(ftab)
dim(gr)
```

dim_pretty *Calculate optimal column widths and row heights*

Description

return minimum estimated widths and heights for each table columns and rows in inches.

Usage

```
dim_pretty(x, part = "all", unit = "in", hspans = "none")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' can be used.
unit	unit for returned values, one of "in", "cm", "mm".

hspans	specifies how cells that are horizontally are included in the calculation. It must be one of the following values "none", "divided" or "included". If "none", widths of horizontally spanned cells is set to 0 (then do not affect the widths); if "divided", widths of horizontally spanned cells is divided by the number of spanned cells; if "included", all widths (included horizontally spanned cells) will be used in the calculation.

See Also

Other functions for flextable size management: autofit(), dim.flextable(), fit_columns(), fit_to_width(), flextable_dim(), height(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()

Examples

```
ftab <- flextable(head(mtcars))
dim_pretty(ftab)
```

---

empty_blanks *Make blank columns transparent*

Description

Blank columns are set as transparent. This is a shortcut function that deletes top and bottom borders, changes the background color to transparent, displays empty content, and sets blank column widths.

Usage

```
empty_blanks(x, width = 0.05, unit = "in", part = "all")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
width	width of blank columns (.1 inch by default).
unit	unit for width, one of "in", "cm", "mm".
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), font(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
typology <- data.frame(
  col_keys = c(
    "Sepal.Length", "Sepal.Width", "Petal.Length",
    "Petal.Width", "Species"
  ),
  what = c("Sepal", "Sepal", "Petal", "Petal", " "),
  measure = c("Length", "Width", "Length", "Width", "Species"),
  stringsAsFactors = FALSE
)
typology

ftab <- flextable(head(iris), col_keys = c(
  "Species",
  "break1", "Sepal.Length", "Sepal.Width",
  "break2", "Petal.Length", "Petal.Width"
))
ftab <- set_header_df(ftab, mapping = typology, key = "col_keys")
ftab <- merge_h(ftab, part = "header")
ftab <- theme_vanilla(ftab)
ftab <- empty_blanks(ftab)
ftab <- width(ftab, j = c(2, 5), width = .1)
ftab
```

---

fit_columns *Constrain table width by wrapping text*

Description

Shrink column widths so that the total table width does not exceed `max_width`. Font sizes are unchanged; text wraps inside the narrower cells.

Columns that cannot shrink below their longest word (plus padding) are clamped at that floor; remaining space is iteratively redistributed among unclamped columns.

Columns listed in `no_wrap` keep their optimal width (as computed by `dim_pretty()`) and are not compressed.

To constrain width by reducing font size instead, see `fit_to_width()`. To size columns to their content without a width constraint, see `autofit()`.

Usage

```
fit_columns(x, max_width, no_wrap = NULL, unit = "in")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`max_width`	maximum total table width (in inches by default)

no_wrap	column names or indices that must keep their optimal width (no compression, no text wrapping).
unit	unit for max_width, one of "in", "cm", "mm".

See Also

Other functions for flextable size management: autofit(), dim.flextable(), dim_pretty(), fit_to_width(), flextable_dim(), height(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()

Examples

```
ft <- qflextable(head(mtcars))
ft <- fit_columns(ft, max_width = 5)
flextable_dim(ft)$widths
ft
```

---

fit_to_width *Constrain table width by shrinking font size*

Description

Decrease font size incrementally until the table fits within max_width.

To constrain width by wrapping text instead, see fit_columns(). To size columns to their content without a width constraint, see autofit().

Usage

```
fit_to_width(x, max_width, inc = 1L, max_iter = 20, unit = "in")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
max_width	maximum width to fit in inches
inc	the font size decrease for each step
max_iter	maximum iterations
unit	unit for max_width, one of "in", "cm", "mm".

See Also

Other functions for flextable size management: autofit(), dim.flextable(), dim_pretty(), fit_columns(), flextable_dim(), height(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()

Examples

```
ft_1 <- qflextable(head(mtcars))
ft_1 <- width(ft_1, width = 1)
ft_1

ft_2 <- fit_to_width(ft_1, max_width = 4)
ft_2
```

`flextable` *Create a flextable from a data frame*

Description

Create a flextable object with function `flextable`.

`flextable` are designed to make tabular reporting easier for R users. Functions are available to let you format text, paragraphs and cells; table cells can be merge vertically or horizontally, row headers can easily be defined, rows heights and columns widths can be manually set or automatically computed.

If working with 'R Markdown' documents, you should read about knitr chunk options in `knit_print.flextable()` and about setting default values with `set_flextable_defaults()`.

Usage

```
flextable(
  data,
  col_keys = names(data),
  cwidth = 0.75,
  cheight = 0.25,
  defaults = list(),
  theme_fun = theme_booktabs,
  use_labels = TRUE
)

qflextable(data)
```

Arguments

`data`	dataset
`col_keys`	columns names/keys to display. If some column names are not in the dataset, they will be added as blank columns by default.
`cwidth, cheight`	initial width and height to use for cell sizes in inches.
`defaults, theme_fun`	deprecated, use `set_flextable_defaults()` instead.
`use_labels`	Logical; if TRUE, any column labels or value labels present in the dataset will be used for display purposes. Defaults to TRUE.

Reuse frequently used parameters

Some default formatting properties are automatically applied to every flextable you produce.

It is highly recommended to use this function because its use will minimize the code. For example, instead of calling the `fontsize()` function over and over again for each new flextable, set the font size default value by calling (before creating the flextables) `set_flextable_defaults(font.size = 11)`. This is also a simple way to have homogeneous arrays and make the documents containing them easier to read.

You can change these default values with function `set_flextable_defaults()`. You can reset them with function `init_flextable_defaults()`. You can access these values by calling `get_flextable_defaults()`.

new lines and tabulations

The 'flextable' package will translate for you the new lines expressed in the form `\n` and the tabs expressed in the form `\t`.

The new lines will be transformed into "soft-return", that is to say a simple carriage return and not a new paragraph.

Tabs are different depending on the output format:

- HTML is using entity *em space*
- Word - a Word 'tab' element
- PowerPoint - a PowerPoint 'tab' element
- latex - tag "\quad "
- Typst - horizontal space `#h(1em)`

flextable parts

A `flextable` is made of 3 parts: header, body and footer.

Most functions have an argument named `part` that will be used to specify what part of of the table should be modified.

qflextable

`qflextable` is a convenient tool to produce quickly a flextable for reporting where layout is fixed (see `set_table_properties()`) and columns widths are adjusted with `autofit()`.

See Also

`style()`, `autofit()`, `theme_booktabs()`, `knit_print.flextable()`, `compose()`, `footnote()`, `set_caption()`

Other flextable constructors: `as_flextable()`

Examples

```
ft <- flextable(head(mtcars))
ft
```

flextable_dim *Get overall width and height of a flextable*

Description

Returns the width, height and aspect ratio of a flextable in a named list. The aspect ratio is the ratio corresponding to `height/width`.

Names of the list are `widths`, `heights` and `aspect_ratio`.

Usage

```
flextable_dim(x, unit = "in")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`unit`	unit for returned values, one of "in", "cm", "mm".

See Also

Other functions for flextable size management: `autofit()`, `dim.flextable()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

```
ftab <- flextable(head(iris))
flextable_dim(ftab)
ftab <- autofit(ftab)
flextable_dim(ftab)
```

flextable_to_rmd *Print a flextable inside knitr loops and conditionals*

Description

Print flextable in R Markdown or Quarto documents within `for` loop or `if` statement.

The function is particularly useful when you want to generate flextable in a loop from a R Markdown document.

Inside R Markdown document, chunk option `results` must be set to 'asis'.

See knit_print.flextable for more details.

Usage

```
flextable_to_rmd(x, ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	unused argument

See Also

Other functions for flextable output and export: df_printer(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
## Not run:
library(rmarkdown)
if (pandoc_available() &&
  pandoc_version() > numeric_version("2")) {
  demo_loop <- system.file(
    package = "flextable",
    "examples/rmd",
    "loop_with_flextable.Rmd"
  )
  rmd_file <- tempfile(fileext = ".Rmd")
  file.copy(demo_loop, to = rmd_file, overwrite = TRUE)
  render(
    input = rmd_file, output_format = "html_document",
    output_file = "loop_with_flextable.html"
  )
}

## End(Not run)
```

---

fmt_2stats *Format summarizor statistics as text*

Description

Format the output of summarizor() into display strings: quantitative variables are shown as mean (sd), median (IQR), or range; qualitative variables are shown as count (percentage).

Usage

```
fmt_2stats(stat, num1, num2, cts, pcts, ...)

fmt_summarizor(stat, num1, num2, cts, pcts, ...)
```

Arguments

stat	a character column containing the name of statictics
num1	a numeric statistic to display such as a mean or a median
num2	a numeric statistic to display such as a standard deviation or a median absolute deviation.
cts	a count to display
pcts	a percentage to display
...	unused arguments

See Also

summarizor(), tabulator(), mk_par()

Other value formatters: fmt_avg_dev(), fmt_dbl(), fmt_header_n(), fmt_int(), fmt_n_percent(), fmt_pct(), fmt_signif_after_zeros()

Examples

```
library(flextable)
z <- summarizor(iris, by = "Species")

tab_1 <- tabulator(
  x = z,
  rows = c("variable", "stat"),
  columns = "Species",
  blah = as_paragraph(
    as_chunk(
      fmt_summarizor(
        stat = stat,
        num1 = value1, num2 = value2,
        cts = cts, pcts = percent
      )
    )
  )
)

ft_1 <- as_flextable(x = tab_1, separate_with = "variable")
ft_1 <- labelizor(
  x = ft_1, j = "stat",
  labels = c(
    mean_sd = "Moyenne (ecart-type)",
    median_iqr = "Mediane (IQR)",
    range = "Etendue",
    missing = "Valeurs manquantes"
  )
)
ft_1 <- autofit(ft_1)
ft_1
```

fmt_avg_dev *Format mean and standard deviation as text*

Description

Format means and standard deviations as `mean (sd)`.

Usage

```
fmt_avg_dev(avg, dev, digit1 = 1, digit2 = 1)
```

Arguments

avg, dev	mean and sd values
digit1, digit2	number of digits to show when printing 'mean' and 'sd'.

See Also

`tabulator()`, `mk_par()`

Other value formatters: `fmt_2stats()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_n_percent()`, `fmt_pct()`, `fmt_signif_after_zeros()`

Examples

```
library(flextable)

df <- data.frame(avg = 1:3 * 3, sd = 1:3)

ft_1 <- flextable(df, col_keys = "avg")
ft_1 <- mk_par(
  x = ft_1, j = 1, part = "body",
  value = as_paragraph(fmt_avg_dev(avg = avg, dev = sd))
)
ft_1 <- autofit(ft_1)
ft_1
```

fmt_dbl *Format numbers as doubles*

Description

Format numeric values with decimal digits, using the flextable default settings for separators and precision.

Usage

```
fmt_dbl(x)
```

Arguments

x	numeric values

See Also

tabulator(), mk_par()

Other value formatters: fmt_2stats(), fmt_avg_dev(), fmt_header_n(), fmt_int(), fmt_n_percent(), fmt_pct(), fmt_signif_after_zeros()

Examples

```
library(flextable)

df <- data.frame(zz = .45)

ft_1 <- flextable(df)
ft_1 <- mk_par(
  x = ft_1, j = 1, part = "body",
  value = as_paragraph(as_chunk(zz, formatter = fmt_dbl))
)
ft_1 <- autofit(ft_1)
ft_1
```

---

fmt_header_n *Format count as '(N=XX)' for column headers*

Description

Format counts as \n(N=XX) for appending sample sizes to column titles.

Usage

```
fmt_header_n(n, newline = TRUE)
```

Arguments

n	count values
newline	indicates to prefix the text with a new line (sof return).

See Also

tabulator(), mk_par()

Other value formatters: fmt_2stats(), fmt_avg_dev(), fmt_dbl(), fmt_int(), fmt_n_percent(), fmt_pct(), fmt_signif_after_zeros()

Examples

```
library(flextable)

df <- data.frame(zz = 1)

ft_1 <- flextable(df)
ft_1 <- append_chunks(
  x = ft_1, j = 1, part = "header",
  value = as_chunk(fmt_header_n(200))
)
ft_1 <- autofit(ft_1)
ft_1
```

---

fmt_int *Format numbers as integers*

Description

Format numeric values as integers (no decimals).

Usage

```
fmt_int(x)
```

Arguments

x	numeric values

See Also

tabulator(), mk_par()

Other value formatters: fmt_2stats(), fmt_avg_dev(), fmt_dbl(), fmt_header_n(), fmt_n_percent(), fmt_pct(), fmt_signif_after_zeros()

Examples

```
library(flextable)

df <- data.frame(zz = 1.23)

ft_1 <- flextable(df)
ft_1 <- mk_par(
  x = ft_1, j = 1, part = "body",
  value = as_paragraph(as_chunk(zz, formatter = fmt_int))
)
ft_1 <- autofit(ft_1)
ft_1
```

`fmt_n_percent` *Format count and percentage as text*

Description

Format counts and percentages as `n (xx.x%)`. If percentages are missing, only the count is shown.

Usage

```
fmt_n_percent(n, pct, digit = 1)
```

Arguments

`n`	count values
`pct`	percent values
`digit`	number of digits for the percentages

See Also

`tabulator()`, `mk_par()`

Other value formatters: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_pct()`, `fmt_signif_after_zeros()`

Examples

```
library(flextable)

df <- structure(
  list(
    cut = structure(
      .Data = 1:5, levels = c(
        "Fair", "Good", "Very Good", "Premium", "Ideal"
      ),
      class = c("ordered", "factor")
    ),
    n = c(1610L, 4906L, 12082L, 13791L, 21551L),
    pct = c(0.0299, 0.0909, 0.2239, 0.2557, 0.3995)
  ),
  row.names = c(NA, -5L),
  class = "data.frame"
)

ft_1 <- flextable(df, col_keys = c("cut", "txt"))
ft_1 <- mk_par(
  x = ft_1, j = "txt",
  value = as_paragraph(fmt_n_percent(n, pct))
)
ft_1 <- align(ft_1, j = "txt", part = "all", align = "right")
```

```
ft_1 <- autofit(ft_1)
ft_1
```

`fmt_pct` *Format numbers as percentages*

Description

Format numeric values as percentages (e.g. `"45.0%"`).

Usage

```
fmt_pct(x)
```

Arguments

x	numeric values

See Also

`tabulator()`, `mk_par()`

Other value formatters: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_n_percent()`, `fmt_signif_after_zeros()`

Examples

```
library(flextable)

df <- data.frame(zz = .45)

ft_1 <- flextable(df)
ft_1 <- mk_par(
  x = ft_1, j = 1, part = "body",
  value = as_paragraph(as_chunk(zz, formatter = fmt_pct))
)
ft_1 <- autofit(ft_1)
ft_1
```

fmt_signif_after_zeros *Format with significant figures after zeros*

Description

Rounds significant figures after zeros in numeric vectors. The number of digits displayed after the leading zeros is customizable using the `digits` parameter.

Usage

```
fmt_signif_after_zeros(x, digits = 3)
```

Arguments

`x`	numeric values
`digits`	number of digits displayed after the leading zeros

See Also

`tabulator()`, `mk_par()`

Other value formatters: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_n_percent()`, `fmt_pct()`

Examples

```
x <- data.frame(
  x = c(0.00000004567, 2.000003456, 3, pi)
)
ft_1 <- flextable(x)
ft_1 <- align(x = ft_1, j = 1, align = "left")
mk_par(ft_1, value = as_paragraph(
  fmt_signif_after_zeros(x)))
```

font *Set font*

Description

Change the font of selected rows and columns of a flextable.

Fonts impact the readability and aesthetics of the table. Font families refer to a set of typefaces that share common design features, such as 'Arial' and 'Open Sans'.

'Google Fonts' is a popular library of free web fonts that can be easily integrated into flextable with the `gdtools::register_gfont()` function. When the output is HTML, the font will be automatically added to the HTML document.

Usage

```
font(
  x,
  i = NULL,
  j = NULL,
  fontname,
  part = "body",
  cs.family = fontname,
  hansi.family = fontname,
  eastasia.family = fontname
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
fontname	single character value, the font family name. With Word and PowerPoint output, this value specifies the font to be used for formatting characters in the Unicode range (U+0000-U+007F).
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.
cs.family	Optional font to be used for formatting characters in a complex script Unicode range. For example, Arabic text might be displayed using the "Arial Unicode MS" font. Used only with Word and PowerPoint outputs. The default value is the value of fontname.
hansi.family	Optional font to be used for formatting characters in a Unicode range that does not fall into one of the other categories. Used only with Word and PowerPoint outputs. The default value is the value of fontname.
eastasia.family	Optional font to be used for formatting characters in an East Asian Unicode range. For example, Japanese text might be displayed using the "MS Mincho" font. Used only with Word and PowerPoint outputs. The default value is the value of fontname.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
library(gdtools)
fontname <- "Brush Script MT"

if (font_family_exists(fontname)) {
  ft_1 <- flextable(head(iris))
  ft_2 <- font(ft_1, fontname = fontname, part = "header")
  ft_2 <- font(ft_2, fontname = fontname, j = 5)
  ft_2
}
```

`fontsize` *Set font size*

Description

Change the font size of selected rows and columns of a flextable.

Usage

```
fontsize(x, i = NULL, j = NULL, size = 11, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`size`	integer value (points)
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
ft <- flextable(head(iris))
ft <- fontsize(ft, size = 14, part = "header")
ft <- fontsize(ft, size = 14, j = 2)
ft <- fontsize(ft, size = 7, j = 3)
ft
```

footnote *Add footnotes to flextable*

Description

The function let add footnotes to a flextable object by adding some symbols in the flextable and associated notes in the footer of the flextable.

Symbols are added to the cells designated by the selection `i` and `j`. If you use i = c(1,3) and j = c(2,5), then you will add the symbols (or the repeated symbol) to cells `[1,2]` and `[3,5]`.

Usage

```
footnote(
  x,
  i = NULL,
  j = NULL,
  value,
  ref_symbols = NULL,
  part = "body",
  inline = FALSE,
  sep = "; ",
  symbol_sep = ""
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`value`	a call to function `as_paragraph()`.
`ref_symbols`	character value, symbols to append that will be used as references to notes.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' is not allowed by the function.
`inline`	whether to add footnote on same line as previous footnote or not
`sep`	used only when inline = TRUE, character string to use as a separator between footnotes.
`symbol_sep`	separator to insert between multiple footnote symbols in the same cell (e.g. `","` to produce `1,2` instead of `12`). Default is `""` (no separator, backward compatible).

See Also

Other functions to compose cell content: append_chunks(), as_paragraph(), compose(), labelizor(), prepend_chunks(), void()

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- footnote(ft_1,
  i = 1, j = 1:3,
  value = as_paragraph(
    c(
      "This is footnote one",
      "This is footnote two",
      "This is footnote three"
    )
  ),
  ref_symbols = c("a", "b", "c"),
  part = "header"
)
ft_1 <- valign(ft_1, valign = "bottom", part = "header")
ft_1 <- autofit(ft_1)

ft_2 <- flextable(head(iris))
ft_2 <- autofit(ft_2)
ft_2 <- footnote(ft_2,
  i = 1, j = 1:2,
  value = as_paragraph(
    c(
      "This is footnote one",
      "This is footnote two"
    )
  ),
  ref_symbols = c("a", "b"),
  part = "header", inline = TRUE
)
ft_2 <- footnote(ft_2,
  i = 1, j = 3:4,
  value = as_paragraph(
    c(
      "This is footnote three",
      "This is footnote four"
    )
  ),
  ref_symbols = c("c", "d"),
  part = "header", inline = TRUE
)
ft_2

ft_3 <- flextable(head(iris))
ft_3 <- autofit(ft_3)
ft_3 <- footnote(
  x = ft_3, i = 1:3, j = 1:3,
```

```
  ref_symbols = "a",
  value = as_paragraph("This is footnote one")
)
ft_3
```

`fp_border_default` *Create border formatting with flextable defaults*

Description

Create a `officer::fp_border()` object whose unspecified arguments inherit from `set_flextable_defaults()` (border color, width).

Use `fp_border_default()` instead of `officer::fp_border()` so that borders automatically match the table's default style.

Usage

```
fp_border_default(
  color = flextable_global$defaults$border.color,
  style = "solid",
  width = flextable_global$defaults$border.width
)
```

Arguments

`color`	border color - single character value (e.g. "#000000" or "black")
`style`	border style - single character value : See Details for supported border styles.
`width`	border width - an integer value : 0>= value

See Also

`hline()`, `vline()`

Other default formatting properties: `fp_text_default()`

Examples

```
library(flextable)

set_flextable_defaults(
  border.color = "orange"
)

z <- flextable(head(cars))
z <- theme_vanilla(z)
z <- vline(
  z,
  j = 1, part = "all",
  border = officer::fp_border()
```

```
)
z <- vline(
  z,
  j = 2, part = "all",
  border = fp_border_default()
)
z

init_flextable_defaults()
```

`fp_text_default` *Create text formatting with flextable defaults*

Description

Create a `officer::fp_text()` object whose unspecified arguments inherit from `set_flextable_defaults()` (font family, size, color).

Use `fp_text_default()` instead of `officer::fp_text()` when building chunks with `as_chunk()`: only override the properties you need, and the rest will match the table's default style.

See also `set_flextable_defaults()` to modify the inherited values.

Usage

```
fp_text_default(
  color = flextable_global$defaults$font.color,
  font.size = flextable_global$defaults$font.size,
  bold = FALSE,
  italic = FALSE,
  underlined = FALSE,
  strike = FALSE,
  font.family = flextable_global$defaults$font.family,
  cs.family = NULL,
  eastasia.family = NULL,
  hansi.family = NULL,
  vertical.align = "baseline",
  shading.color = "transparent"
)
```

Arguments

`color`	font color - a single character value specifying a valid color (e.g. "#000000" or "black").
`font.size`	font size (in point) - 0 or positive integer value.
`bold`	is bold
`italic`	is italic

underlined	is underlined
strike	is strikethrough
font.family	single character value. Specifies the font to be used to format characters in the Unicode range (U+0000-U+007F).
cs.family	optional font to be used to format characters in a complex script Unicode range. For example, Arabic text might be displayed using the "Arial Unicode MS" font.
eastasia.family	optional font to be used to format characters in an East Asian Unicode range. For example, Japanese text might be displayed using the "MS Mincho" font.
hansi.family	optional. Specifies the font to be used to format characters in a Unicode range which does not fall into one of the other categories.
vertical.align	single character value specifying font vertical alignments. Expected value is one of the following : default 'baseline' or 'subscript' or 'superscript'
shading.color	shading color - a single character value specifying a valid color (e.g. "#000000" or "black").

See Also

as_chunk(), compose(), append_chunks(), prepend_chunks()

Other default formatting properties: fp_border_default()

Examples

```
library(flextable)

set_flextable_defaults(
  font.size = 11, font.color = "#303030",
  padding = 3, table.layout = "autofit"
)
z <- flextable(head(cars))

z <- compose(
  x = z,
  i = ~ speed < 6,
  j = "speed",
  value = as_paragraph(
    as_chunk("slow... ", props = fp_text_default(color = "red")),
    as_chunk(speed, props = fp_text_default(italic = TRUE))
  )
)
z

init_flextable_defaults()
```

gen_grob *Render a flextable as a graphic object*

Description

gen_grob() converts a flextable into a Grid Graphics object (grob) that can be drawn on any R graphic device. This is the function behind save_as_image() and the patchwork integration (wrap_flextable()).

Typical uses:

- embed a flextable in a ggplot2 plot (via wrap_flextable() or cowplot)
- export a flextable as a PNG or SVG image (via save_as_image())

Text wrapping and scaling are supported. The fit argument controls how the table adapts to the available space (fixed size, auto-fit width, or fill the device).

Not recommended for very large tables because the grid calculations can be slow.

Limitations: equations (as_equation()) and hyperlinks (officer::hyperlink_ftext()) are not rendered.

Use a 'ragg', 'svglite' or 'ggiraph' device for correct rendering.

Usage

```
gen_grob(
  x,
  ...,
  fit = c("auto", "width", "fixed"),
  scaling = c("min", "full", "fixed"),
  wrapping = TRUE,
  autowidths = TRUE,
  just = NULL
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	Reserved for extra arguments
fit	Determines the fitting/scaling of the grob on its parent viewport. One of auto, width, fixed, TRUE, FALSE: • auto or TRUE (default): The grob is resized to fit in the parent viewport. The table row heights and column widths are resized proportionally. • width: The grob is resized horizontally to fit the width of the parent viewport. The column widths are resized proportionally. The row heights are unaffected and the table height may be smaller or larger than the height of the parent viewport.

- `fixed` or `FALSE`: The grob will have fixed dimensions, as determined by the column widths and the row heights.

`scaling` Determines the scaling of the table contents. One of `min`, `full`, `fixed`, `TRUE`, `FALSE`:

- `min` or `TRUE` (default): When the parent viewport is smaller than the necessary, the various content sizes (text font size, line width and image dimensions) will decrease accordingly so that the content can still fit. When the parent viewport is larger than the necessary, the content sizes will remain the same, they will not increase.
- `full`: Same as `min`, except that the content sizes are scaled fully, they will increase or decrease, according to the size of the drawing surface.
- `fixed` or `FALSE`: The content sizes will not be scaled.

`wrapping` Determines the soft wrapping (line breaking) method for the table cell contents. One of `TRUE`, `FALSE`:

- `TRUE`: Text content may wrap into separate lines at normal word break points (such as a space or tab character between two words) or at newline characters anywhere in the text content. If a word does not fit in the available cell width, then the text content may wrap at any character. Non-text content (such as images) is also wrapped into new lines, according to the available cell width.
- `FALSE`: Text content may wrap only with a newline character. Non-text content is not wrapped.

Superscript and subscript chunks do not wrap. Newline and tab characters are removed from these chunk types.

`autowidths` If `TRUE` (default) the column widths are adjusted in order to fit the contents of the cells (taking into account the `wrapping` setting).

`just` Justification of viewport layout, same as `just` argument in `grid::grid.layout()`. When set to `NULL` (default), it is determined according to the `fit` argument.

Value

a grob (gTree) object made with package `grid`

size

The size of the flextable can be known by using the method dim on the grob.

caption

It's important to note that captions are not part of the table itself. This means when exporting a table to PNG or SVG formats (image formats), the caption won't be included. Captions are intended for document outputs like Word, HTML, or PDF, where tables are embedded within the document itself.

See Also

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
library(ragg)
library(gdtools)
register_liberationsans()

set_flextable_defaults(font.family = "Liberation Sans")

ft <- flextable(head(mtcars))
gr <- gen_grob(ft)

plot(gr)
```

---

get_flextable_defaults

Get flextable defaults formatting properties

Description

The current formatting properties are automatically applied to every flextable you produce. These default values are returned by this function.

Usage

```
get_flextable_defaults()
```

Value

a list containing default values.

See Also

Other themes and defaults: set_flextable_defaults(), theme_alafoli(), theme_apa(), theme_booktabs(), theme_borderless(), theme_box(), theme_tron(), theme_tron_legacy(), theme_vader(), theme_vanilla(), theme_zebra()

Examples

```
get_flextable_defaults()
```

gg_chunk *ggplot chunk*

Description

This function is used to insert mini gg plots into flextable with functions:

- `compose()` and `as_paragraph()`,
- `append_chunks()`,
- `prepend_chunks()`.

Usage

```
gg_chunk(value, width = 1, height = 0.2, unit = "in", res = 300, alt = "")
```

Arguments

`value`	gg objects, stored in a list column; or a list of 'ggplot' objects.
`width, height`	size of the resulting png file.
`unit`	unit for width and height, one of "in", "cm", "mm".
`res`	resolution of the png image in ppi
`alt`	alternative text for the image (used for accessibility)

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
library(data.table)
library(flextable)
if (require("ggplot2")) {
  my_cor_plot <- function(x) {
    cols <- colnames(x)[sapply(x, is.numeric)]
    x <- x[, .SD, .SDcols = cols]
    cormat <- as.data.table(cor(x))
```

```
    cormat$var1 <- colnames(cormat)
    cormat <- melt(cormat,
      id.vars = "var1", measure.vars = cormat$var1,
      variable.name = "var2", value.name = "correlation"
    )
    ggplot(data = cormat, aes(x = var1, y = var2, fill = correlation)) +
      geom_tile() +
      coord_equal() +
      scale_fill_gradient2(
        low = "blue",
        mid = "white", high = "red", limits = c(-1, 1),
        guide = "none"
      ) +
      theme_void()
  }
  z <- as.data.table(iris)
  z <- z[, list(gg = list(my_cor_plot(.SD))), by = "Species"]
  ft <- flextable(z)
  ft <- mk_par(ft,
    j = "gg",
    value = as_paragraph(
      gg_chunk(value = gg, width = 1, height = 1)
    )
  )
  ft
}
```

---

grid_chunk *Grid Graphics chunk*

Description

This function is used to insert grid objects into flextable with functions:

- compose() and as_paragraph(),
- append_chunks(),
- prepend_chunks().

Usage

```
grid_chunk(value, width = 1, height = 0.2, unit = "in", res = 300, alt = "")
```

Arguments

value	grid objects, stored in a list column; or a list of grid objects.
width, height	size of the resulting png file
unit	unit for width and height, one of "in", "cm", "mm".
res	resolution of the png image in ppi
alt	alternative text for the image (used for accessibility)

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
library(flextable)
ft_1 <- flextable(head(cars))
if (require("grid")) {
  ft_1 <- prepend_chunks(
    x = ft_1, i = 2, j = 2,
    grid_chunk(
      list(
        circleGrob(gp = gpar(
          fill = "#ec11c2",
          col = "transparent"
        ))
      ),
      width = .15, height = .15
    )
  )
}
ft_1
```

---

height *Set flextable rows height*

Description

control rows height for a part of the flextable when the line height adjustment is "atleast" or "exact" (see `hrule()`).

Usage

```
height(x, i = NULL, height, part = "body", unit = "in")

height_all(x, height, part = "all", unit = "in")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
height	height in inches
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.
unit	unit for height, one of "in", "cm", "mm".

height_all

height_all is a convenient function for setting the same height to all rows (selected with argument part).

Note

This function has no effect when the rule for line height is set to "auto" (see hrule()), which is the default case, except with PowerPoint which does not support this automatic line height adjustment feature.

See Also

Other functions for flextable size management: autofit(), dim.flextable(), dim_pretty(), fit_columns(), fit_to_width(), flextable_dim(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- height(ft_1, height = .5)
ft_1 <- hrule(ft_1, rule = "exact")
ft_1


ft_2 <- flextable(head(iris))
ft_2 <- height_all(ft_2, height = 1)
ft_2 <- hrule(ft_2, rule = "exact")
ft_2
```

`highlight` *Set text highlight color*

Description

Change the text highlight color of selected rows and columns of a flextable. A function can be used instead of fixed colors.

When `color` is a function, it is possible to color cells based on values located in other columns; using hidden columns (those not used by argument `colkeys`) is a common use case. The argument `source` must be used to define the columns to be used for the color definition, and the argument `j` must be used to define where to apply the colors and only accepts values from `colkeys`.

Usage

```
highlight(x, i = NULL, j = NULL, color = "yellow", part = "body", source = j)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`color`	color to use as text highlighting color. If a function, the function must return a character vector of colors.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.
`source`	if color is a function, source specifies the dataset column to be used as an argument to `color`. This is only useful when j is colored with values contained in other columns.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
my_color_fun <- function(x) {
  out <- rep("yellow", length(x))
  out[x < quantile(x, .75)] <- "pink"
  out[x < quantile(x, .50)] <- "wheat"
  out[x < quantile(x, .25)] <- "gray90"
  out
```

```
}
ft <- flextable(head(mtcars, n = 10))
ft <- highlight(ft, j = "disp", i = ~ disp > 200, color = "yellow")
ft <- highlight(ft, j = ~ drat + wt + qsec, color = my_color_fun)
ft
```

`hline` *Set horizontal borders below selected rows*

Description

`hline()` draws a horizontal line **below** each selected row by setting the bottom border of cells at row `i` (and the top border of cells at row `i + 1` so that the line renders consistently across output formats).

Use the `i` selector to target specific rows (e.g. a formula such as `~ before(col, "Total")`). When `i` is `NULL` (the default) the border is added below every row, which produces a full grid of inner horizontal lines.

For the **outer** edges of the table, use `hline_top()` and `hline_bottom()` instead; those always target the very first or very last row of a part.

Usage

```
hline(x, i = NULL, j = NULL, border = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline()`, `vline_left()`, `vline_right()`

Examples

```
library(officer)
std_border <- fp_border(color = "gray")

ft <- flextable(head(iris))
ft <- border_remove(x = ft)

# add horizontal borders below every row
ft <- hline(ft, part = "all", border = std_border)
ft
```

`hline_bottom` *Set the bottom border of a table part*

Description

`hline_bottom()` draws a horizontal line at the **very bottom** of a table part. It does not accept a row selector `i` because it always targets the last row.

Unlike `hline()`, which adds inner lines below arbitrary rows, `hline_bottom()` is meant for the outer bottom edge of a part.

Usage

```
hline_bottom(x, j = NULL, border = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_top()`, `surround()`, `vline()`, `vline_left()`, `vline_right()`

Examples

```
library(officer)
big_border <- fp_border(color = "orange", width = 3)

ft <- flextable(head(iris))
ft <- border_remove(x = ft)
```

```
# add a thick line at the bottom of the body
ft <- hline_bottom(ft, part = "body", border = big_border)
ft
```

`hline_top` *Set the top border of a table part*

Description

`hline_top()` draws a horizontal line at the **very top** of a table part. It does not accept a row selector `i` because it always targets the first row.

When the part above exists (e.g. header above body), the line is stored as the bottom border of that adjacent part so that it renders seamlessly.

Unlike `hline()`, which adds inner lines below arbitrary rows, `hline_top()` is meant for the outer top edge of a part.

Usage

```
hline_top(x, j = NULL, border = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_bottom()`, `surround()`, `vline()`, `vline_left()`, `vline_right()`

Examples

```
library(officer)
big_border <- fp_border(color = "orange", width = 3)

ft <- flextable(head(iris))
ft <- border_remove(x = ft)

# add a thick line on top of each part
ft <- hline_top(ft, part = "all", border = big_border)
ft
```

hrule *Set how row heights are determined*

Description

`hrule()` controls whether row heights are automatic, minimum or fixed. This only affects Word and PowerPoint outputs; it has no effect on HTML or PDF.

- `"auto"` (default): the row height adjusts to fit the content; any value set by `height()` is ignored.
- `"atleast"`: the row is at least as tall as the value set by `height()`, but can grow if the content is taller.
- `"exact"`: the row is exactly the height set by `height()`; content that overflows is clipped.

For PDF see the `ft.arraystretch` chunk option.

Usage

```
hrule(x, i = NULL, rule = "auto", part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`rule`	specify the meaning of the height. Possible values are "atleast" (height should be at least the value specified), "exact" (height should be exactly the value specified), or the default value "auto" (height is determined based on the height of the contents, so the value is ignored).
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other functions for flextable size management: `autofit()`, `dim.flextable()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `flextable_dim()`, `height()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- width(ft_1, width = 1.5)
ft_1 <- height(ft_1, height = 0.75, part = "header")
ft_1 <- hrule(ft_1, rule = "exact", part = "header")
ft_1

ft_2 <- hrule(ft_1, rule = "auto", part = "header")
ft_2
```

`htmltools_value` *Convert a flextable to an HTML object*

Description

get a `htmltools::div()` from a flextable object. This can be used in a shiny application. For an output within "R Markdown" document, use knit_print.flextable.

Usage

```
htmltools_value(
  x,
  ft.align = NULL,
  ft.shadow = NULL,
  extra_dependencies = NULL
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`ft.align`	flextable alignment, supported values are 'left', 'center' and 'right'.
`ft.shadow`	deprecated.
`extra_dependencies`	a list of HTML dependencies to add in the HTML output.

Value

an object marked as htmltools::HTML ready to be used within a call to `shiny::renderUI` for example.

See Also

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`, `wrap_flextable()`

Examples

```
htmltools_value(flextable(iris[1:5, ]))
```

hyperlink_text *Hyperlink chunk*

Description

The function lets add hyperlinks within flextable objects.

It is used to add it to the content of a cell of the flextable with the functions `compose()`, `append_chunks()` or `prepend_chunks()`.

URL are not encoded, they are preserved 'as is'.

Usage

```
hyperlink_text(x, props = NULL, formatter = format_fun, url, ...)
```

Arguments

x	text or any element that can be formatted as text with function provided in argument `formatter`.
props	an `fp_text_default()` or `officer::fp_text()` object to be used to format the text. If not specified, it will be the default value corresponding to the cell.
formatter	a function that will format x as a character vector.
url	url to be used
...	additional arguments for `formatter` function.

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

See Also

`compose()`

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `linerange()`, `minibar()`, `plot_chunk()`

Examples

```
dat <- data.frame(
  col = "Google it",
  href = "https://www.google.fr/search?source=hp&q=flextable+R+package",
  stringsAsFactors = FALSE
)

ftab <- flextable(dat)
```

```
ftab <- compose(
  x = ftab, j = "col",
  value = as_paragraph(
    "This is a link: ",
    hyperlink_text(x = col, url = href)
  )
)
ftab
```

`indentation` *Set paragraph indents*

Description

Change the first-line or hanging indent of paragraphs in selected rows and columns of a flextable.

A *first-line* indent moves only the first line of the paragraph to the right. A *hanging* indent moves all lines **but** the first one to the right; the indented lines start at the paragraph left padding plus the hanging value, so `hanging` should not exceed the left padding (see `padding()`).

Usage

```
indentation(
  x,
  i = NULL,
  j = NULL,
  first_line = NULL,
  hanging = NULL,
  part = "body"
)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`first_line`	first-line indent in pts (points). Use NA to unset.
`hanging`	hanging indent in pts (points). Use NA to unset. If both `first_line` and `hanging` are provided, `hanging` wins.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), font(), fontsize(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- mk_par(
  x = ft_1, i = 1, j = 1, part = "header",
  value = as_paragraph("A rather long header that will wrap on two lines")
)
ft_1 <- padding(ft_1, i = 1, j = 1, padding.left = 30, part = "header")
ft_1 <- indentation(ft_1, i = 1, j = 1, hanging = 20, part = "header")
ft_1 <- width(ft_1, j = 1, width = 1.2)
ft_1
```

---

italic *Set italic font*

Description

Change the font decoration of selected rows and columns of a flextable.

Usage

```
italic(x, i = NULL, j = NULL, italic = TRUE, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
italic	boolean value
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), font(), fontsize(), highlight(), indentation(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
ft <- flextable(head(mtcars))
ft <- italic(ft, italic = TRUE, part = "header")
```

---

keep_with_next *Set Word 'Keep with next' instructions*

Description

The 'Keep with next' functionality in 'Word', applied to the rows of a table, ensures that rows with that attribute stay together and do not break across multiple pages.

This function provides better control of page breaks than the global `keep_with_next` parameter.

Usage

```
keep_with_next(x, i = NULL, value = TRUE, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`value`	TRUE or FALSE. When applied to a group, all rows except the last one should be flagged with the 'Keep with next' attribute.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' is not allowed by the function.

See Also

`paginate()`

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `tab_settings()`, `valign()`

Examples

```
library(flextable)
dat <- iris[c(1:25, 51:75, 101:125), ]
ft <- qflextable(dat)
ft <- keep_with_next(
  x = ft,
  i = c(1:24, 26:49, 51:74),
  value = TRUE
)

save_as_docx(ft, path = tempfile(fileext = ".docx"))
```

`knit_print.flextable` *Render flextable in knitr documents*

Description

This function is called automatically by knitr to display a flextable in R Markdown and Quarto documents. You do not need to call it directly.

Supported output formats: HTML, Word (docx), PDF, PowerPoint (pptx) and Typst (`format: typst`, Quarto only). For other formats (e.g., `github_document`, `beamer_presentation`), the table is rendered as a PNG image.

Usage

```
## S3 method for class 'flextable'
knit_print(x, ...)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`...`	unused.

Getting started

No special setup is needed: place a flextable object in a code chunk and it will be rendered in the output document.

Add a caption with `set_caption()`:

```
ft <- set_caption(ft, caption = "My table caption")
```

In Quarto documents, use chunk options `tbl-cap` and `label` instead:

````
```{r}
#| label: tbl-mytable
#| tbl-cap: "My table caption"
ft
```
````

Captions

Recommended method: use `set_caption()` to define the caption directly on the flextable object. When `set_caption()` is used, chunk options related to captions are ignored.

Alternative (R Markdown only): use knitr chunk options `tab.cap` and `tab.id`:

Description	Chunk option	Default

Caption text	tab.cap	NULL
Caption id/bookmark	tab.id	NULL
Caption on top of the table	tab.topcaption	TRUE
Caption sequence identifier	tab.lp	"tab:"
Word style for captions	tab.cap.style	NULL

Bookdown: cross-references use the pattern `\@ref(tab:chunk_label)`. The usual bookdown numbering applies.

Quarto: cross-references use `@tbl-chunk_label`. To embed cross-references or other Quarto markdown inside flextable cells, use `as_qmd()` with the `flextable-qmd` extension (see `use_flextable_qmd()`).

Chunk options

Use `knitr::opts_chunk$set(...)` to set defaults for the whole document.

All formats:

- `ft.align`: table alignment, one of `'left'`, `'center'` (default) or `'right'`.

HTML:

- `ft.htmlscroll`: `TRUE` or `FALSE` (default) to enable horizontal scrolling. See `set_table_properties()` for finer control.

Word:

- `ft.split`: allow rows to break across pages (`TRUE` by default).

PDF:

- `ft.tabcolsep`: space between text and cell borders (default 0 pt).
- `ft.arraystretch`: row height multiplier (default 1.5).
- `ft.latex.float`: floating placement. One of `'none'` (default), `'float'`, `'wrap-r'`, `'wrap-l'`, `'wrap-i'`, `'wrap-o'`.

PowerPoint:

- `ft.left`, `ft.top`: top-left coordinates of the table placeholder in inches (defaults: 1 and 2).

Word with officedown

When using `officedown::rdocx_document()`, additional caption options are available:

Description	Chunk option	Default
Numbering prefix	tab.cap.pre	"Table "
Numbering suffix	tab.cap.sep	": "
Title number depth	tab.cap.tnd	0
Caption prefix formatting	tab.cap.fp_text	`fp_text_lite(bold=TRUE)`
Title number / table number separator	tab.cap.tns	"-"

Quarto

flextable works natively in Quarto documents for HTML, PDF, Word and Typst.

The `flextable-qmd` Lua filter extension enables Quarto markdown inside flextable cells: cross-references (`@tbl-xxx`, `@fig-xxx`), bold/italic, links, math, inline code and shortcodes. See `as_qmd()` and `use_flextable_qmd()` for setup instructions.

PDF accessibility (PDF/UA-2)

Quarto's `pdf-standard: ua-2` injects `\DocumentMetadata{tagging=on}` in the LaTeX preamble, activating 'tagpdf'. This code patches LaTeX commands at compile time to insert PDF structure tags. Neither Quarto nor flextable control this process.

The tagging code does not yet support `longtable` + `colortbl`, `booktabs` rules, and `cline`, which flextable relies on. Compilation fails with `\ERRORtbl/row`. When these upstream issues are resolved, flextable PDF output will support tagging without changes. Other formats are not affected.

PDF limitations

The following properties are not supported in PDF output: `padding.top`, `padding.bottom`, `line_spacing` and row `height`. Justified text is converted to left-aligned.

To use system fonts, set `latex_engine: xelatex` in the YAML header (the default `pdflatex` engine does not support them).

See `add_latex_dep()` when caching flextable results.

PowerPoint limitations

PowerPoint only supports fixed table layout. Use `autofit()` to adjust column widths. Images inside table cells are not supported (this is a PowerPoint limitation).

HTML note

HTML output uses Shadow DOM to isolate table styles from the rest of the page.

See Also

`set_caption()`, `as_qmd()`, `use_flextable_qmd()`, `paginate()`

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `plot.flextable()`, `print.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`, `wrap_flextable()`

Examples

```
## Not run:
library(rmarkdown)
if (pandoc_available() &&
  pandoc_version() > numeric_version("2")) {
  demo_loop <- system.file(
    package = "flextable",
```

```
    "examples/rmd",
    "demo.Rmd"
  )
  rmd_file <- tempfile(fileext = ".Rmd")
  file.copy(demo_loop, to = rmd_file, overwrite = TRUE)
  render(
    input = rmd_file, output_format = "html_document",
    output_file = "demo.html"
  )
}

## End(Not run)
```

`labelizor` *Replace displayed text with labels*

Description

`labelizor()` substitutes text values shown in a flextable with human-readable labels. This is useful to turn column values such as variable names, factor levels or coded strings into presentation-ready wording (e.g. `"Sepal.Length"` to `"Sepal Length"`).

`labels` can be either a **named character vector** (names identify values to find, values are the replacement labels) or a **function** applied to every text chunk (e.g. `toupper()`).

Only the displayed content is affected; the underlying data of the flextable is unchanged.

Usage

```
labelizor(x, j = NULL, labels, part = "all")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`labels`	a named vector whose names will be used to identify values to replace and values will be used as labels.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

`mk_par()`, `append_chunks()`, `prepend_chunks()`

Other functions to compose cell content: `append_chunks()`, `as_paragraph()`, `compose()`, `footnote()`, `prepend_chunks()`, `void()`

Examples

```
z <- summarizor(
  x = CO2[-c(1, 4)],
  by = "Treatment",
  overall_label = "Overall"
)

ft_1 <- as_flextable(z, separate_with = "variable")

ft_1 <- labelizor(
  x = ft_1, j = c("stat"),
  labels = c(Missing = "Kouign amann")
)

ft_1 <- labelizor(
  x = ft_1, j = c("variable"),
  labels = toupper
)

ft_1
```

---

linerange *Mini linerange chunk*

Description

This function is used to insert lineranges into flextable with functions:

- compose() and as_paragraph(),
- append_chunks(),
- prepend_chunks().

Usage

```
linerange(
  value,
  min = NULL,
  max = NULL,
  rangecol = "#CCCCCC",
  stickcol = "#FF0000",
  bg = "transparent",
  width = 1,
  height = 0.2,
  raster_width = 30,
  unit = "in",
  alt = ""
)
```

Arguments

value	values containing the bar size
min	min bar size. Default min of value
max	max bar size. Default max of value
rangecol	bar color
stickcol	jauge color
bg	background color
width, height	size of the resulting png file in inches
raster_width	number of pixels used as width when interpolating value.
unit	unit for width and height, one of "in", "cm", "mm".
alt	alternative text for the image (used for accessibility)

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

`compose()`, `as_paragraph()`

Other chunk elements for paragraph: `as_b()`, `as_bracket()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_i()`, `as_image()`, `as_qmd()`, `as_strike()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `minibar()`, `plot_chunk()`

Examples

```
myft <- flextable(head(iris, n = 10))

myft <- compose(myft,
  j = 1,
  value = as_paragraph(
    linerange(value = Sepal.Length)
  ),
  part = "body"
)

autofit(myft)
```

line_spacing *Set line spacing*

Description

Change the line spacing of selected rows and columns of a flextable.

Usage

```
line_spacing(x, i = NULL, j = NULL, space = 1, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
space	space between lines of text; 1 is single line spacing, 2 is double line spacing.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), font(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), padding(), rotate(), style(), tab_settings(), valign()

Examples

```
ft <- flextable(head(mtcars)[, 3:6])
ft <- line_spacing(ft, space = 1.6, part = "all")
ft <- set_table_properties(ft, layout = "autofit")
ft
```

merge_at *Merge flextable cells into a single one*

Description

Merge flextable cells into a single one. All rows and columns must be consecutive.

Usage

```
merge_at(x, i = NULL, j = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.

See Also

Other cell merging functions: merge_h(), merge_h_range(), merge_none(), merge_v()

Examples

```
ft_merge <- flextable(head(mtcars), cwidth = .5)
ft_merge <- merge_at(ft_merge, i = 1:2, j = 1:2)
ft_merge
```

---

merge_h *Merge flextable cells horizontally*

Description

Merge flextable cells horizontally when consecutive cells have identical values. Text of formatted values are used to compare values.

Usage

```
merge_h(x, i = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.

See Also

Other cell merging functions: merge_at(), merge_h_range(), merge_none(), merge_v()

Examples

```
library(flextable)

schedule <- data.frame(
  time = c("9h", "10h", "11h", "14h", "15h", "16h"),
  monday = c("Math", "Math", "French", "History", "Science", "French"),
  tuesday = c("English", "Math", "Art", "Math", "Math", "French"),
  wednesday = c("Science", "Math", "Science", "English", "English", "French"),
  stringsAsFactors = FALSE
)

ft <- flextable(schedule)
ft <- theme_box(ft)
ft <- merge_h(ft)
ft
```

---

merge_h_range *Rowwise merge of a range of columns*

Description

Merge flextable columns into a single one for each selected rows. All columns must be consecutive.

Usage

```
merge_h_range(x, i = NULL, j1 = NULL, j2 = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j1, j2	selected columns that will define the range of columns to merge.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.

See Also

Other cell merging functions: merge_at(), merge_h(), merge_none(), merge_v()

Examples

```
ft <- flextable(head(mtcars), cwidth = .5)
ft <- theme_box(ft)
ft <- merge_h_range(ft, i = ~ cyl == 6, j1 = "am", j2 = "carb")
ft <- flextable::align(ft, i = ~ cyl == 6, align = "center")
ft
```

merge_none *Delete flextable merging information*

Description

Delete all merging information from a flextable.

Usage

```
merge_none(x, part = "all")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other cell merging functions: `merge_at()`, `merge_h()`, `merge_h_range()`, `merge_v()`

Examples

```
typology <- data.frame(
  col_keys = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"),
  what = c("Sepal", "Sepal", "Petal", "Petal", "Species"),
  measure = c("Length", "Width", "Length", "Width", "Species"),
  stringsAsFactors = FALSE
)

ft <- flextable(head(iris))
ft <- set_header_df(ft, mapping = typology, key = "col_keys")
ft <- merge_v(ft, j = c("Species"))

ft <- theme_tron_legacy(merge_none(ft))
ft
```

merge_v *Merge flextable cells vertically*

Description

Merge flextable cells vertically when consecutive cells have identical values. Text of formatted values are used to compare values if available.

Two options are available, either a column-by-column algorithm or an algorithm where the combinations of these columns are used once for all target columns.

Usage

```
merge_v(x, j = NULL, target = NULL, part = "body", combine = FALSE)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
j	column selector, see section *Column selection with the* `j` *parameter* in <Selectors in flextable>.
target	columns names where cells have to be merged.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.
combine	If the value is TRUE, the columns defined by `j` will be combined into a single column/value and the consecutive values of this result will be used. Otherwise, the columns are inspected one by one to perform cell merges.

See Also

Other cell merging functions: `merge_at()`, `merge_h()`, `merge_h_range()`, `merge_none()`

Examples

```
ft_merge <- flextable(mtcars)
ft_merge <- merge_v(ft_merge, j = c("gear", "carb"))
ft_merge

data_ex <- structure(list(srdr_id = c(
  "175124", "175124", "172525", "172525",
  "172545", "172545", "172609", "172609", "172609"
), substances = c(
  "alcohol",
  "alcohol", "alcohol", "alcohol", "cannabis",
  "cannabis", "alcohol\n cannabis\n other drugs",
  "alcohol\n cannabis\n other drugs",
  "alcohol\n cannabis\n other drugs"
), full_name = c(
  "TAU", "MI", "TAU", "MI (parent)", "TAU", "MI",
  "TAU", "MI", "MI"
), article_arm_name = c(
  "Control", "WISEteens",
  "Treatment as usual", "Brief MI (b-MI)", "Assessed control",
  "Intervention", "Control", "Computer BI", "Therapist BI"
)), row.names = c(
  NA,
  -9L
), class = c("tbl_df", "tbl", "data.frame"))


ft_1 <- flextable(data_ex)
ft_1 <- theme_box(ft_1)
```

```
ft_2 <- merge_v(ft_1,
  j = "srdr_id",
  target = c("srdr_id", "substances")
)
ft_2
```

`minibar` *Mini barplot chunk*

Description

This function is used to insert bars into flextable with functions:

- `compose()` and `as_paragraph()`,
- `append_chunks()`,
- `prepend_chunks()`.

Usage

```
minibar(
  value,
  max = NULL,
  barcol = "#CCCCCC",
  bg = "transparent",
  width = 1,
  height = 0.2,
  unit = "in",
  alt = ""
)
```

Arguments

`value`	values containing the bar size
`max`	max bar size
`barcol`	bar color
`bg`	background color
`width, height`	size of the resulting png file in inches
`unit`	unit for width and height, one of "in", "cm", "mm".
`alt`	alternative text for the image (used for accessibility)

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (`format: docx`) or `rmarkdown::word_document()`, the resulting file must be repaired with `repair_docx()`.

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

compose(), as_paragraph()

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), plot_chunk()

Examples

```
ft <- flextable(head(iris, n = 10))

ft <- compose(ft,
  j = 1,
  value = as_paragraph(
    minibar(value = Sepal.Length, max = max(Sepal.Length))
  ),
  part = "body"
)

ft <- autofit(ft)
ft
```

---

ncol_keys *Number of columns*

Description

returns the number of columns displayed

Usage

```
ncol_keys(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

See Also

Other functions for flextable size management: autofit(), dim.flextable(), dim_pretty(), fit_columns(), fit_to_width(), flextable_dim(), height(), hrule(), nrow_part(), set_table_properties(), width()

Examples

```
library(flextable)
ft <- qflextable(head(cars))
ncol_keys(ft)
```

nrow_part *Number of rows of a part*

Description

returns the number of lines in a part of flextable.

Usage

```
nrow_part(x, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other functions for flextable size management: autofit(), dim.flextable(), dim_pretty(), fit_columns(), fit_to_width(), flextable_dim(), height(), hrule(), ncol_keys(), set_table_properties(), width()

Examples

```
library(flextable)
ft <- qflextable(head(cars))
nrow_part(ft, part = "body")
```

padding *Set paragraph paddings*

Description

Change the padding of selected rows and columns of a flextable.

Usage

```
padding(
  x,
  i = NULL,
  j = NULL,
  padding = NULL,
  padding.top = NULL,
  padding.bottom = NULL,
```

```
  padding.left = NULL,
  padding.right = NULL,
  part = "body"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
padding	padding (shortcut for top, bottom, left and right), unit is pts (points).
padding.top	padding top, unit is pts (points).
padding.bottom	padding bottom, unit is pts (points).
padding.left	padding left, unit is pts (points).
padding.right	padding right, unit is pts (points).
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

Note

In PDF output, padding.top and padding.bottom are ignored by default: for a long time they were not supported, and row heights were only driven by arraystretch. To keep existing PDF tables unchanged, applying them is opt-in with the padding_vertical option (see set_flextable_defaults() or opts_pdf in set_table_properties()), preferably with arraystretch = 1: set_flextable_defaults(padding_vert = TRUE, arraystretch = 1). They are then supported with the "fixed" layout (the default) but not with the "autofit" layout.

Global horizontal spacing can also be set with set_table_properties(opts_pdf = list(tabcolsep = 1)).

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), font(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), line_spacing(), rotate(), style(), tab_settings(), valign()

Examples

```
ft_1 <- flextable(head(iris))
ft_1 <- theme_vader(ft_1)
ft_1 <- padding(ft_1, padding.top = 4, part = "all")
ft_1 <- padding(ft_1, j = 1, padding.right = 40)
ft_1 <- padding(ft_1, i = 3, padding.top = 40)
ft_1 <- padding(ft_1, padding.top = 10, part = "header")
ft_1 <- padding(ft_1, padding.bottom = 10, part = "header")
```

```
ft_1 <- autofit(ft_1)
ft_1
```

---

paginate *Prevent page breaks inside a flextable*

Description

Prevents breaks between tables rows you want to stay together. This feature only applies to Word and RTF output.

Usage

```
paginate(
  x,
  init = NULL,
  hdr_ftr = TRUE,
  group = character(),
  group_def = c("rle", "nonempty", "starts")
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
init	init value for keep_with_next property, it default value is `get_flextable_defaults()$keep_with_next`
hdr_ftr	if TRUE (default), prevent breaks between table body and header and between table body and footer.
group	name of a column to use for finding groups (when `group_def` is `"rle"` or `"nonempty"`) or an integer vector of row indices that start new groups (when `group_def` is `"starts"`).
group_def	algorithm to be used to identify groups that should not be split into two pages, one of 'rle', 'nonempty', 'starts': • 'rle': runs of equal values are used to define the groups, to be used with `tabulator()`. • 'nonempty': non empty value start a new group, to be used with `as_flextable.tabular()`. • 'starts': `group` is a numeric vector of body row indices where new groups begin. Page breaks are allowed before these rows.

Details

The pagination of tables allows you to control their position in relation to page breaks.

For small tables, a simple setting is usually used that indicates that all rows should be displayed together:

```
paginate(x, init = TRUE, hdr_ftr = TRUE)
```

For large tables, it is recommended to use a setting that indicates that all rows of the header should be bound to the first row of the table to avoid the case where the header is displayed alone at the bottom of the page and then repeated on the next one:

```
paginate(x, init = FALSE, hdr_ftr = TRUE)
```

For tables that present groups that you don't want to be presented on two pages, you must use a parameterization involving the notion of group and an algorithm for determining the groups.

```
paginate(x, group = "grp", group_def = "rle")
```

Value

updated flextable object

See Also

knit_print.flextable()

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
library(data.table)
library(flextable)

init_flextable_defaults()

multi_fun <- function(x) {
  list(mean = mean(x), sd = sd(x))
}

dat <- as.data.table(ggplot2::diamonds)
dat <- dat[clarity %in% c("I1", "SI1", "VS2")]

dat <- dat[, unlist(lapply(.SD, multi_fun),
  recursive = FALSE
),
.SDcols = c("z", "y"),
by = c("cut", "color", "clarity")
]

tab <- tabulator(
  x = dat, rows = c("cut", "color"),
  columns = "clarity",
  `z stats` = as_paragraph(as_chunk(fmt_avg_dev(z.mean, z.sd, digit2 = 2))),
```

```
  `y stats` = as_paragraph(as_chunk(fmt_avg_dev(y.mean, y.sd, digit2 = 2)))
)

ft_1 <- as_flextable(tab)
ft_1 <- autofit(x = ft_1, add_w = .05)
ft_1 <- paginate(ft_1, group = "cut", group_def = "rle")

save_as_docx(ft_1, path = tempfile(fileext = ".docx"))
save_as_rtf(ft_1, path = tempfile(fileext = ".rtf"))
```

`ph_with.flextable` *Add a flextable into a PowerPoint slide*

Description

Add a flextable in a PowerPoint document object produced by `officer::read_pptx()`.

This function will create a native PowerPoint table from the flextable and the result can be eventually edited.

Usage

```
## S3 method for class 'flextable'
ph_with(x, value, location, ...)
```

Arguments

`x`	a pptx device
`value`	flextable object
`location`	a location for a placeholder. See `officer::ph_location_type()` for example.
`...`	unused arguments.

caption

Captions are not printed in PowerPoint slides.

While captions are useful for document formats like Word, RTF, HTML, or PDF, they aren't directly supported in PowerPoint slides. Unlike documents with a defined layout, PowerPoint slides lack a structured document flow. They don't function like HTML documents or paginated formats (RTF, Word, PDF). This makes it technically challenging to determine the ideal placement for a caption within a slide. Additionally, including a caption within the table itself isn't feasible.

Note

The width and height of the table can not be set with `location`. Use functions `width()`, `height()`, `autofit()` and `dim_pretty()` instead. The overall size is resulting from cells, paragraphs and text properties (i.e. padding, font size, border widths).

See Also

Other functions for officer integration: body_add_flextable(), body_replace_flextable_at_bkm(), rtf_add.flextable()

Examples

```
library(officer)

ft <- flextable(head(iris))

doc <- read_pptx()
doc <- add_slide(doc, "Title and Content", "Office Theme")
doc <- ph_with(doc, ft, location = ph_location_left())

fileout <- tempfile(fileext = ".pptx")
print(doc, target = fileout)
```

---

plot.flextable *Plot a flextable*

Description

plots a flextable as a grid grob object and display the result in a new graphics window. 'ragg' or 'svglite' or 'ggiraph' graphical device drivers should be used to ensure a correct rendering.

Usage

```
## S3 method for class 'flextable'
plot(x, ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	additional arguments passed to gen_grob().

caption

It's important to note that captions are not part of the table itself. This means when exporting a table to PNG or SVG formats (image formats), the caption won't be included. Captions are intended for document outputs like Word, HTML, or PDF, where tables are embedded within the document itself.

See Also

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
library(gdtools)
library(ragg)
register_liberationsans()
set_flextable_defaults(font.family = "Liberation Sans")
ftab <- as_flextable(cars)


plot(ftab)
```

---

plot.flextableGrob *plot a flextable grob*

Description

plot a flextable grob

Usage

```
## S3 method for class 'flextableGrob'
plot(x, ...)
```

Arguments

x	a flextableGrob object
...	additional arguments passed to other functions

---

plot_chunk *Mini plot chunk*

Description

This function is used to insert mini plots into flextable with functions:

- `compose()` and `as_paragraph()`,
- `append_chunks()`,
- `prepend_chunks()`.

Available plots are 'box', 'line', 'points', 'density'.

Usage

```
plot_chunk(
  value,
  width = 1,
  height = 0.2,
  type = "box",
  free_scale = FALSE,
  unit = "in",
  alt = "",
  ...
)
```

Arguments

value	a numeric vector, stored in a list column.
width, height	size of the resulting png file in inches
type	type of the plot: 'box', 'line', 'points' or 'density'.
free_scale	Should scales be free (TRUE or FALSE, the default value).
unit	unit for width and height, one of "in", "cm", "mm".
alt	alternative text for the image (used for accessibility)
...	arguments sent to plot functions (see par())

Note

This chunk option requires package officedown in a R Markdown context with Word output format. With Quarto (format: docx) or rmarkdown::word_document(), the resulting file must be repaired with repair_docx().

PowerPoint cannot mix images and text in a paragraph, images are removed when outputing to PowerPoint format.

See Also

Other chunk elements for paragraph: as_b(), as_bracket(), as_chunk(), as_equation(), as_highlight(), as_i(), as_image(), as_qmd(), as_strike(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), minibar()

Examples

```
library(data.table)
library(flextable)

z <- as.data.table(iris)
z <- z[, list(
  Sepal.Length = mean(Sepal.Length, na.rm = TRUE),
  z = list(.SD$Sepal.Length)
), by = "Species"]
```

```
ft <- flextable(z,
  col_keys = c("Species", "Sepal.Length", "box", "density")
)
ft <- mk_par(ft, j = "box", value = as_paragraph(
  plot_chunk(
    value = z, type = "box",
    border = "red", col = "transparent"
  )
))
ft <- mk_par(ft, j = "density", value = as_paragraph(
  plot_chunk(value = z, type = "dens", col = "red")
))
ft <- set_table_properties(ft, layout = "autofit", width = .6)
ft <- set_header_labels(ft, box = "boxplot", density = "density")
theme_vanilla(ft)
```

---

prepend_chunks *Prepend chunks to flextable content*

Description

prepend chunks (for example chunk as_chunk()) in a flextable.

Usage

```
prepend_chunks(x, ..., i = NULL, j = NULL, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	chunks to be prepended, see as_chunk(), gg_chunk() and other chunk elements for paragraph.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' is not allowed by the function.

See Also

Other functions to compose cell content: append_chunks(), as_paragraph(), compose(), footnote(), labelizor(), void()

Examples

```
x <- flextable(head(iris))
x <- prepend_chunks(
  x,
  i = 1, j = 1,
  colorize(as_b("Hello "), color = "red"),
  colorize(as_i("World"), color = "magenta")
)
x
```

`print.flextable` *Print a flextable*

Description

print a flextable object to format `html`, `docx`, `pptx` or as text (not for display but for informative purpose). This function is to be used in an interactive context.

Usage

```
## S3 method for class 'flextable'
print(x, preview = "html", align = "center", ...)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`preview`	preview type, one of c("html", "pptx", "docx", "rtf", "pdf, "log"). When `"log"` is used, a description of the flextable is printed.
`align`	left, center (default) or right. Only for docx/html/pdf.
`...`	arguments for 'pdf_document' call when preview is "pdf".

Note

When argument `preview` is set to `"docx"` or `"pptx"`, an external client linked to these formats (Office is installed) is used to edit a document. The document is saved in the temporary directory of the R session and will be removed when R session will be ended.

When argument `preview` is set to `"html"`, an external client linked to these HTML format is used to display the table. If RStudio is used, the Viewer is used to display the table.

Note also that a print method is used when flextable are used within R markdown documents. See `knit_print.flextable()`.

See Also

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`, `wrap_flextable()`

proc_freq *Frequency table*

Description

This function computes a one or two way contingency table and creates a flextable from the result.

The function is largely inspired by "PROC FREQ" from "SAS" and was written with the intent to make it as compact as possible.

Usage

```
proc_freq(
  x,
  row = character(),
  col = character(),
  include.row_percent = TRUE,
  include.column_percent = TRUE,
  include.table_percent = TRUE,
  include.table_count = TRUE,
  weight = character(),
  count_format_fun = fmt_int,
  ...
)
```

Arguments

x	a data.frame object containing variable(s) to use for counts.
row	characer column names for row
col	characer column names for column
include.row_percent	boolean whether to include the row percents; defaults to TRUE
include.column_percent	boolean whether to include the column percents; defaults to TRUE
include.table_percent	boolean whether to include the table percents; defaults to TRUE
include.table_count	boolean whether to include the table counts; defaults to TRUE
weight	character column name for weight
count_format_fun	a function to format the count values, defaults to fmt_int.
...	unused arguments

Examples

```
proc_freq(mtcars, "vs", "gear")
proc_freq(mtcars, "gear", "vs", weight = "wt")
```

repair_docx *Repair flextables in a 'Word' document produced by 'pandoc'*

Description

Word documents produced by Quarto (`format: docx`) or by `rmarkdown::word_document()` are written by 'pandoc', which copies the raw OOXML of flextable objects verbatim: images and hyperlinks are left as unresolved references (file paths and URLs instead of relationship identifiers) and **Word refuses to open the file**.

This function reads such a document and rewrites it with the package 'officer', whose writer resolves these references: images and SVG files are embedded, hyperlinks are registered, and the other officer-flavored contents are processed as well (sections, footnotes, comments, custom style names, list markers, documents poured with `officer::block_pour_docx()`).

It is not needed when the document is produced with `officedown::rdocx_document()` or `save_as_docx()`, where the file is already written by 'officer'.

Usage

```
repair_docx(path, target = path)
```

Arguments

`path`	path of the 'docx' file to repair. The file is modified in place unless `target` is provided.
`target`	path of the resulting 'docx' file. Defaults to `path`.

Value

the path of the repaired file, invisibly.

What to do with Quarto

The repair must run after `quarto render`. Quarto can do it for you with a *post-render script*, but this can **only** be declared in a `_quarto.yml` file - the `project:` key is ignored in the YAML header of a `.qmd` file.

1. Next to your `.qmd` file, create a file `_quarto.yml` containing:

```
project:
  post-render: repair.R
```

1. In the same directory, create the file `repair.R` containing:

```
files <- strsplit(Sys.getenv("QUARTO_PROJECT_OUTPUT_FILES"), "\n")[[1]]
for (f in files[grepl("\\.docx$", files, ignore.case = TRUE)]) {
  flextable::repair_docx(f)
}
```

1. Render as usual, either the whole project (`quarto render`) or a single file (`quarto render doc.qmd`); the script runs in both cases and only processes the files that were rendered.

Without a `_quarto.yml`, call `repair_docx("doc.docx")` manually after each render.

What to do with R Markdown

Prefer `officedown::rdocx_document()` as output format: the file is post-processed by 'officer' and no repair is needed. With `rmarkdown::word_document()`, call `repair_docx()` on the output file after `rmarkdown::render()`.

Requirements and limits

- The images referenced in the document must still exist, at paths resolvable **from the directory where `repair_docx()` runs**: a Quarto post-render script runs at the project root, so images referenced with paths relative to a sub-directory document will not be found - use absolute paths (e.g. `here::here()`) in that case.
- Mini graphics generated by `minibar()`, `linerange()`, `gg_chunk()`, `plot_chunk()` and `grid_chunk()` are written to temporary 'PNG' files that no longer exist once rendering has completed: **these can not be repaired afterwards**. With Quarto or `rmarkdown::word_document()`, use image files you manage yourself (`as_image()`, `colformat_image()`); mini graphics require `officedown::rdocx_document()` or `save_as_docx()`.
- When a referenced image can not be found, the rewrite fails with an error `cannot open the connection` and a warning naming the missing file.

See Also

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`, `wrap_flextable()`

Examples

```
## Not run:
repair_docx("document.docx")

## End(Not run)
```

---

rotate *Rotate cell text*

Description

It can be useful to change the text direction when table headers are large. For example, header labels can be rendered as "tbrl" (top to bottom and right to left), corresponding to a 90-degree rotation, or "btlr", corresponding to a 270-degree rotation. This function changes cell text direction. By default, it is "lrtb", which means from left to right and top to bottom.

'Word' and 'PowerPoint' do not handle automatic height with rotated headers. Therefore, you need to set header heights (with the `height()` function) and set the rule to "exact" for row heights (with the `hrule()` function); otherwise, Word and PowerPoint outputs will have insufficient height to properly display the text.

flextable does not rotate text by arbitrary angles. It only rotates by right angles (90-degree increments). This choice ensures consistent rendering across Word, PowerPoint (limited to angles 0, 270, and 90), HTML, and PDF.

Usage

```
rotate(x, i = NULL, j = NULL, rotation, align = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`rotation`	one of "lrtb", "tbrl", "btlr".
`align`	vertical alignment of paragraph within cell, one of "center" or "top" or "bottom".
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

Details

When the `autofit()` function is used, rotation will be ignored. In that case, use dim_pretty and width instead of `autofit()`.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `style()`, `tab_settings()`, `valign()`

Examples

```
library(flextable)

ft_1 <- flextable(head(iris))
```

```
ft_1 <- rotate(ft_1, j = 1:4, align = "bottom", rotation = "tbrl", part = "header")
ft_1 <- rotate(ft_1, j = 5, align = "bottom", rotation = "btlr", part = "header")

# if output is docx or pptx, think about (1) set header heights
# and (2) set rule "exact" for rows heights because Word
# and PowerPoint don't handle auto height with rotated headers
ft_1 <- height(ft_1, height = 1.2, part = "header")
ft_1 <- hrule(ft_1, i = 1, rule = "exact", part = "header")

ft_1

dat <- data.frame(
  a = c("left-top", "left-middle", "left-bottom"),
  b = c("center-top", "center-middle", "center-bottom"),
  c = c("right-top", "right-middle", "right-bottom")
)

ft_2 <- flextable(dat)
ft_2 <- theme_box(ft_2)
ft_2 <- height_all(x = ft_2, height = 1.3, part = "body")
ft_2 <- hrule(ft_2, rule = "exact")
ft_2 <- rotate(ft_2, rotation = "tbrl")
ft_2 <- width(ft_2, width = 1.3)

ft_2 <- align(ft_2, j = 1, align = "left")
ft_2 <- align(ft_2, j = 2, align = "center")
ft_2 <- align(ft_2, j = 3, align = "right")

ft_2 <- valign(ft_2, i = 1, valign = "top")
ft_2 <- valign(ft_2, i = 2, valign = "center")
ft_2 <- valign(ft_2, i = 3, valign = "bottom")

ft_2
```

`rtf_add.flextable` *Add a 'flextable' into an RTF document*

Description

`officer::rtf_add()` method for adding flextable objects into 'RTF' documents.

Usage

```
## S3 method for class 'flextable'
rtf_add(x, value, ...)
```

Arguments

`x`	rtf object, created by `officer::rtf_doc()`.
`value`	a flextable object
`...`	unused arguments

See Also

Other functions for officer integration: body_add_flextable(), body_replace_flextable_at_bkm(), ph_with.flextable()

Examples

```
library(flextable)
library(officer)

ft <- flextable(head(iris))
ft <- autofit(ft)

z <- rtf_doc()
z <- rtf_add(z, ft)

print(z, target = tempfile(fileext = ".rtf"))
```

---

save_as_docx *Save flextable objects in a 'Word' file*

Description

sugar function to save flextable objects in an Word file.

Usage

```
save_as_docx(..., values = NULL, path, pr_section = NULL, align = "center")
```

Arguments

...	flextable objects, objects, possibly named. If named objects, names are used as titles.
values	a list (possibly named), each element is a flextable object. If named objects, names are used as titles. If provided, argument ... will be ignored.
path	Word file to be created
pr_section	a officer::prop_section object that can be used to define page layout such as orientation, width and height.
align	left, center (default) or right.

Value

a string containing the full name of the generated file

See Also

paginate()

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
tf <- tempfile(fileext = ".docx")

library(officer)
ft1 <- flextable(head(iris))
save_as_docx(ft1, path = tf)


ft2 <- flextable(head(mtcars))
sect_properties <- prop_section(
  page_size = page_size(
    orient = "landscape",
    width = 8.3, height = 11.7
  ),
  type = "continuous",
  page_margins = page_mar()
)
save_as_docx(
  `iris table` = ft1, `mtcars table` = ft2,
  path = tf, pr_section = sect_properties
)
```

---

save_as_html *Save flextable objects in an 'HTML' file*

Description

save a flextable in an 'HTML' file. This function is useful to save the flextable in 'HTML' file without using R Markdown (it is highly recommanded to use R Markdown instead).

Usage

```
save_as_html(..., values = NULL, path, lang = "en", title = "&#32;")
```

Arguments

...	flextable objects, objects, possibly named. If named objects, names are used as titles.
values	a list (possibly named), each element is a flextable object. If named objects, names are used as titles. If provided, argument ... will be ignored.

path	HTML file to be created
lang	language of the document using IETF language tags
title	page title

Value

a string containing the full name of the generated file

See Also

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
ft1 <- flextable(head(iris))
tf1 <- tempfile(fileext = ".html")
if (rmarkdown::pandoc_available()) {
  save_as_html(ft1, path = tf1)
  # browseURL(tf1)
}

ft2 <- flextable(head(mtcars))
tf2 <- tempfile(fileext = ".html")
if (rmarkdown::pandoc_available()) {
  save_as_html(
    `iris table` = ft1,
    `mtcars table` = ft2,
    path = tf2,
    title = "rhoooo"
  )
  # browseURL(tf2)
}
```

---

save_as_image *Save a flextable in a 'png' or 'svg' file*

Description

Save a flextable as a png or svg image. This function uses R graphic system to create an image from the flextable, allowing for high-quality image output. See gen_grob() for more options.

Usage

```
save_as_image(x, path, expand = 10, res = 200, ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
path	image file to be created. It should end with '.png' or '.svg'.
expand	space in pixels to add around the table.
res	The resolution of the device
...	unused arguments

Value

a string containing the full name of the generated file

caption

It's important to note that captions are not part of the table itself. This means when exporting a table to PNG or SVG formats (image formats), the caption won't be included. Captions are intended for document outputs like Word, HTML, or PDF, where tables are embedded within the document itself.

See Also

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_pptx(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
library(gdtools)
register_liberationsans()
set_flextable_defaults(font.family = "Liberation Sans")

ft <- flextable(head(mtcars))
ft <- autofit(ft)
tf <- tempfile(fileext = ".png")
save_as_image(x = ft, path = tf)

init_flextable_defaults()
```

---

save_as_pptx *Save flextable objects in a 'PowerPoint' file*

Description

sugar function to save flextable objects in an PowerPoint file.

This feature is available to simplify the work of users by avoiding the need to use the 'officer' package. If it doesn't suit your needs, then use the API offered by 'officer' which allows simple and complicated things.

Usage

```
save_as_pptx(..., values = NULL, path)
```

Arguments

...	flextable objects, objects, possibly named. If named objects, names are used as slide titles.
values	a list (possibly named), each element is a flextable object. If named objects, names are used as slide titles. If provided, argument ... will be ignored.
path	PowerPoint file to be created

Value

a string containing the full name of the generated file

Note

The PowerPoint format ignores captions (see set_caption()).

See Also

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_image(), save_as_rtf(), to_html.flextable(), wrap_flextable()

Examples

```
ft1 <- flextable(head(iris))
tf <- tempfile(fileext = ".pptx")
save_as_pptx(ft1, path = tf)

ft2 <- flextable(head(mtcars))
tf <- tempfile(fileext = ".pptx")
save_as_pptx(`iris table` = ft1, `mtcars table` = ft2, path = tf)
```

---

save_as_rtf *Save flextable objects in an 'RTF' file*

Description

sugar function to save flextable objects in an 'RTF' file.

Usage

```
save_as_rtf(..., values = NULL, path, pr_section = NULL)
```

Arguments

...	flextable objects, objects, possibly named. If named objects, names are used as titles.
values	a list (possibly named), each element is a flextable object. If named objects, names are used as titles. If provided, argument ... will be ignored.
path	Word file to be created
pr_section	a officer::prop_section object that can be used to define page layout such as orientation, width and height.

Value

a string containing the full name of the generated file

See Also

paginate()

Other functions for flextable output and export: df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), repair_docx(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), to_html.flextable(), wrap_flextable()

Examples

```
tf <- tempfile(fileext = ".rtf")

library(officer)
ft1 <- flextable(head(iris))
save_as_rtf(ft1, path = tf)


ft2 <- flextable(head(mtcars))
sect_properties <- prop_section(
  page_size = page_size(
    orient = "landscape",
    width = 8.3, height = 11.7
  ),
  type = "continuous",
  page_margins = page_mar(),
  header_default = block_list(
    fpar(ftext("text for default page header")),
    qflextable(data.frame(a = 1L))
  )
)
tf <- tempfile(fileext = ".rtf")
save_as_rtf(
  `iris table` = ft1, `mtcars table` = ft2,
  path = tf, pr_section = sect_properties
)
```

save_as_typst *Save flextable objects in a Typst file*

Description

Writes one or more flextables to a standalone Typst (`.typ`) document. Images are embedded as base64 (decoded inline at compile time), so the file is self-contained and compiles offline with `typst compile`. Equations require the `mitex` Typst package, which is fetched from the Typst package registry on first compilation.

Usage

```
save_as_typst(
  ...,
  values = NULL,
  path,
  page = "#set page(width: auto, height: auto, margin: 12pt)"
)
```

Arguments

`...`	flextable objects, possibly named.
`values`	a list of flextable objects. If provided, `...` is ignored.
`path`	Typst file to be created.
`page`	a Typst `#set page(...)` rule used as document setup. The default produces a tightly cropped page fitting the table(s).

Value

the path to the generated file, invisibly.

Examples

```
ft <- flextable(head(iris))
tf <- tempfile(fileext = ".typ")
save_as_typst(ft, path = tf)
```

separate_header *Split column names using a separator into multiple rows*

Description

This function is used to separate and place individual labels in their own rows if your variable names contain multiple delimited labels.

Usage

```
separate_header(
  x,
  opts = c("span-top", "center-hspan", "bottom-vspan", "default-theme"),
  split = "[_\\.]",
  fixed = FALSE
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
opts	Optional treatments to apply to the resulting header part. This should be a character vector with support for multiple values. Supported values include: • "span-top": This operation spans empty cells with the first non-empty cell, applied column by column. • "center-hspan": Center the cells that are horizontally spanned. • "bottom-vspan": Aligns to the bottom the cells treated at the when "span-top" is applied. • "default-theme": Applies the theme set in `set_flextable_defaults(theme_fun = ...)` to the new header part.
split	a regular expression (unless `fixed = TRUE`) to use for splitting.
fixed	logical. If TRUE match `split` exactly, otherwise use regular expressions.

See Also

Other row and column operations: `add_body()`, `add_body_row()`, `add_footer()`, `add_footer_lines()`, `add_footer_row()`, `add_header()`, `add_header_lines()`, `add_header_row()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `paginate()`, `set_header_footer_df`, `set_header_labels()`, `split_columns()`, `split_rows()`, `split_to_pages()`

Examples

```
library(flextable)

x <- data.frame(
  Species = as.factor(c("setosa", "versicolor", "virginica")),
  Sepal.Length_mean = c(5.006, 5.936, 6.588),
  Sepal.Length_sd = c(0.35249, 0.51617, 0.63588),
  Sepal.Width_mean = c(3.428, 2.77, 2.974),
  Sepal.Width_sd = c(0.37906, 0.3138, 0.3225),
  Petal.Length_mean = c(1.462, 4.26, 5.552),
  Petal.Length_sd = c(0.17366, 0.46991, 0.55189),
  Petal.Width_mean = c(0.246, 1.326, 2.026),
  Petal.Width_sd = c(0.10539, 0.19775, 0.27465)
)

ft_1 <- flextable(x)
ft_1 <- colformat_double(ft_1, digits = 2)
ft_1 <- theme_box(ft_1)
ft_1 <- separate_header(
  x = ft_1,
  opts = c("span-top", "bottom-vspan")
)
ft_1
```

---

set_caption *Set flextable caption*

Description

Set caption value in a flextable. The function can also be used to define formattings that will be applied if possible to Word and HTML outputs.

- The caption will be associated with a paragraph style when the output is Word. It can also be numbered as a auto-numbered Word computed value.
- The PowerPoint format ignores captions. PowerPoint documents are not structured and do not behave as HTML documents and paginated documents (word, pdf), and it's not possible to know where we should create a shape to contain the caption (technically it can't be in the PowerPoint shape containing the table).

When working with 'R Markdown' or 'Quarto', the caption settings defined with `set_caption()` will be prioritized over knitr chunk options.

Caption value can be a single string or the result to a call to `as_paragraph()`. With the latter, the caption is made of formatted chunks whereas with the former, caption will not be associated with any formatting.

Usage

```
set_caption(
  x,
  caption = NULL,
  autonum = NULL,
  word_stylename = "Table Caption",
  style = word_stylename,
  fp_p = fp_par(padding = 3),
  align_with_table = TRUE,
  html_classes = NULL,
  html_escape = TRUE
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
caption	caption value. The caption can be either a string either a call to as_paragraph(). In the latter case, users are free to format the caption with colors, italic fonts, also mixed with images or equations. Note that Quarto does not allow the use of this feature. Caption as a string does not support 'Markdown' syntax. If you want to add a bold text in the caption, use as_paragraph('a ', as_b('bold'), ' text') when providing caption.
autonum	an autonum representation. See officer::run_autonum(). This has an effect only when the output is "Word" (in which case the object is used to define the Word auto-numbering), "html" and "pdf" (in which case only the bookmark identifier will be used). If used, the caption is preceded by an auto-number sequence.
word_stylename, style	'Word' style name to associate with caption paragraph. These names are available with function officer::styles_info() when output is Word. Argument style is deprecated in favor of word_stylename. If the caption is defined with as_paragraph(), some of the formattings of the paragraph style will be replaced by the formattings associated with the chunks (such as the font).
fp_p	paragraph formatting properties associated with the caption, see officer::fp_par(). It applies when possible, i.e. in HTML and 'Word' but not with bookdown.
align_with_table	if TRUE, caption is aligned as the flextable, if FALSE, fp_p will not be updated and alignement is as defined with fp_p. It applies when possible, i.e. in HTML and 'Word' but not with bookdown.
html_classes	css class(es) to apply to associate with caption paragraph when output is 'Word'.
html_escape	should HTML entities be escaped so that it can be safely included as text or an attribute value within an HTML document.

Details

The behavior of captions in the 'flextable' package varies depending on the formats and technologies used.

The values set by the `set_caption()` function will be prioritized whenever possible, including the caption ID and associated paragraph style. However, it's important to note that the behavior may differ across different tools. Here's what we have observed and attempted to respect, but please inform us if you believe our observations are incorrect:

- In Word and HTML documents created with 'rmarkdown' `rmarkdown::word_document()` and `rmarkdown::html_document()`, numbered and cross-referenced captions are not typically expected.
- In PDF documents created with 'rmarkdown' `rmarkdown::pdf_document()`, numbers are automatically added before the caption.
- In Word and HTML documents created with 'bookdown', numbered and cross-referenced captions are expected. 'bookdown' handles this functionality, but due to technical reasons, the caption should not be defined within an HTML or XML block. Therefore, when using 'flextable', the ability to format the caption content is lost (this limitation does not apply to PDF documents).
- HTML and PDF documents created with Quarto handle captions and cross-references differently. Quarto replaces captions with 'tbl-cap' and 'label' values.
- Word documents created with Quarto present another specific case. Currently, Quarto does not inject captions using the 'tbl-cap' and label values. However, this is a temporary situation that is expected to change in the future. The 'flextable' package will adapt accordingly as Quarto evolves.
- When using the `body_add_flextable()` function, all the options specified with `set_caption()` will be enabled.

Using `body_add_flextable()` enable all options specified with `set_caption()`.

R Markdown

flextable captions can be defined from R Markdown documents by using `knitr::opts_chunk$set()`. User don't always have to call `set_caption()` to set a caption, he can use knitr chunk options instead. A typical call would be:

```
```{r}
#| tab.id: bookmark_id
#| tab.cap: caption text
flextable(head(cars))
```
```

`tab.id` is the caption id or bookmark, `tab.cap` is the caption text. There are many options that can replace `set_caption()` features. The following knitr chunk options are available:

label	name	value
Word stylename to use for table captions.	tab.cap.style	NULL

caption id/bookmark	tab.id	NULL
caption	tab.cap	NULL
display table caption on top of the table or not	tab.topcaption	TRUE
caption table sequence identifier.	tab.lp	"tab:"
prefix for numbering chunk (default to "Table ").	tab.cap.pre	Table
suffix for numbering chunk (default to ": ").	tab.cap.sep	" :"
title number depth	tab.cap.tnd	0
separator to use between title number and table number.	tab.cap.tns	"-"
caption prefix formatting properties	tab.cap.fp_text	fp_text_lite(bold = TRUE)

See knit_print.flextable for more details.

Formatting the caption

To create captions in R Markdown using the 'flextable' package and 'officer' package, you can utilize the `as_paragraph()` function. This approach is recommended when your captions require complex content, such as a combination of different text styles or the inclusion of images and equations.

The caption is constructed as a paragraph consisting of multiple chunks. Each chunk represents a specific portion of the caption with its desired formatting, such as red bold text or Arial italic text.

By default, if no specific formatting is specified (using either "a string" or `as_chunk("a string")`), the `fp_text_default()` function sets the font settings for the caption, including the font family, boldness, italics, color, etc. The default values can be modified using the `set_flextable_defaults()` function. However, it is recommended to explicitly use `as_chunk()` to define the desired formatting.

It's important to note that the style properties of the caption will not override the formatting of the individual elements within it. Therefore, you need to explicitly specify the font to be used for the caption.

Here's an example of how to set a caption for a flextable in R Markdown using the 'officer' package:

```
library(flextable)
library(officer)

ftab <- flextable(head(cars)) |>
  set_caption(
    as_paragraph(
      as_chunk("caption", props = fp_text_default(font.family = "Cambria"))
    ), word_stylename = "Table Caption"
  )

print(ftab, preview = "docx")
```

In this example, the `set_caption()` function sets the caption for the flextable. The caption is created using `as_paragraph()` with a single chunk created using `as_chunk("caption", props = fp_text_default(font.family = "Cambria"))`. The `word_stylename` parameter is used to

specify the table caption style in the resulting Word document. Finally, the `print()` function generates the flextable with the caption, and `preview = "docx"` displays a preview of the resulting Word document.

Using 'Quarto'

In 'Quarto', captions and cross-references are handled differently compared to 'R Markdown', where flextable takes care of the job. In Quarto, the responsibility for managing captions lies with the Quarto framework itself. Consequently, the `set_caption()` function in 'flextable' is not as useful in a 'Quarto' document. The formatting and numbering of captions are determined by Quarto rather than flextable. Please refer to the Quarto documentation for more information on how to work with captions in Quarto.

See Also

`flextable()`, `knit_print.flextable()`

Examples

```
ftab <- flextable(head(iris))
ftab <- set_caption(ftab, "my caption")
ftab

library(officer)
autonum <- run_autonum(seq_id = "tab", bkm = "mtcars")
ftab <- flextable(head(mtcars))
ftab <- set_caption(ftab, caption = "mtcars data", autonum = autonum)
ftab
```

---

set_flextable_defaults

Modify flextable defaults formatting properties

Description

The current formatting properties (see `get_flextable_defaults()`) are automatically applied to every flextable you produce. Use `set_flextable_defaults()` to override them. Use `init_flextable_defaults()` to re-init all values with the package defaults.

Usage

```
set_flextable_defaults(
  font.family = NULL,
  font.size = NULL,
  font.color = NULL,
  text.align = NULL,
  padding = NULL,
  padding.bottom = NULL,
```

```
  padding.top = NULL,
  padding.left = NULL,
  padding.right = NULL,
  border.color = NULL,
  border.width = NULL,
  background.color = NULL,
  line_spacing = NULL,
  table.layout = NULL,
  cs.family = NULL,
  eastasia.family = NULL,
  hansi.family = NULL,
  decimal.mark = NULL,
  big.mark = NULL,
  digits = NULL,
  pct_digits = NULL,
  na_str = NULL,
  nan_str = NULL,
  fmt_date = NULL,
  fmt_datetime = NULL,
  extra_css = NULL,
  scroll = NULL,
  table_align = "center",
  split = NULL,
  keep_with_next = NULL,
  tabcolsep = NULL,
  arraystretch = NULL,
  padding_vertical = NULL,
  float = NULL,
  fonts_ignore = NULL,
  theme_fun = NULL,
  post_process_all = NULL,
  post_process_pdf = NULL,
  post_process_docx = NULL,
  post_process_html = NULL,
  post_process_pptx = NULL,
  ...
)

init_flextable_defaults()
```

Arguments

font.family	single character value. When format is Word, it specifies the font to be used to format characters in the Unicode range (U+0000-U+007F). If you want to use non ascii characters in Word, you should also set hansi.family to the same family name.
font.size	font size (in point) - 0 or positive integer value.
font.color	font color - a single character value specifying a valid color (e.g. "#000000" or

"black").

`text.align` text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.

`padding` padding (shortcut for top, bottom, left and right padding)

`padding.bottom`, `padding.top`, `padding.left`, `padding.right`
paragraph paddings - 0 or positive integer value.

`border.color` border color - single character value (e.g. "#000000" or "black").

`border.width` border width in points.

`background.color`
cell background color - a single character value specifying a valid color (e.g. "#000000" or "black").

`line_spacing` space between lines of text, 1 is single line spacing, 2 is double line spacing.

`table.layout` 'autofit' or 'fixed' algorithm. Default to 'autofit'.

`cs.family` optional and only for Word. Font to be used to format characters in a complex script Unicode range. For example, Arabic text might be displayed using the "Arial Unicode MS" font.

`eastasia.family`
optional and only for Word. Font to be used to format characters in an East Asian Unicode range. For example, Japanese text might be displayed using the "MS Mincho" font.

`hansi.family` optional and only for Word. Font to be used to format characters in a Unicode range which does not fall into one of the other categories.

`decimal.mark`, `big.mark`, `na_str`, `nan_str`
formatC arguments used by `colformat_num()`, `colformat_double()`, and `colformat_int()`.

`digits` formatC argument used by `colformat_double()`.

`pct_digits` number of digits for percentages.

`fmt_date`, `fmt_datetime`
formats for date and datetime columns as documented in `strptime()`. Default to '%Y-%m-%d' and '%Y-%m-%d %H:%M:%S'.

`extra_css` css instructions to be integrated with the table.

`scroll` NULL or a list if you want to add a scroll-box. See **scroll** element of argument `opts_html` in function `set_table_properties()`.

`table_align` default flextable alignment, supported values are 'left', 'center' and 'right'.

`split` Word option 'Allow row to break across pages' can be activated when TRUE.

`keep_with_next` default initialization value used by the `paginate()` function corresponding to the Word option "keep rows together" that will be defined in the array.

`tabcolsep` space between the text and the left/right border of its containing cell.

`arraystretch` height of each row relative to its default height, the default value is 1.5.

`padding_vertical`
if TRUE, PDF output applies `padding.top` and `padding.bottom` (see `padding()`) with the "fixed" layout. The default value is FALSE for backward compatibility: rows height then only depends on `arraystretch`, as in previous versions. When activated, `arraystretch` = 1 is recommended, the paddings providing the vertical spacing.

`float` type of floating placement in the PDF document, one of:

- 'none' (the default value), table is placed after the preceding paragraph.
- 'float', table can float to a place in the text where it fits best
- 'wrap-r', wrap text around the table positioned to the right side of the text
- 'wrap-l', wrap text around the table positioned to the left side of the text
- 'wrap-i', wrap text around the table positioned inside edge-near the binding
- 'wrap-o', wrap text around the table positioned outside edge-far from the binding

`fonts_ignore` if TRUE, pdf-engine pdflatex can be used instead of xelatex or lualatex. If pdflatex is used, fonts will be ignored because they are not supported by pdflatex, whereas with the xelatex and lualatex engines they are.

`theme_fun` a single character value (the name of the theme function to be applied) or a theme function (input is a flextable, output is a flextable).

`post_process_all` Post-processing function that will allow you to customize the the table. It will be executed before call to post_process_pdf(), post_process_docx(), post_process_html(), post_process_pptx().

`post_process_pdf`, `post_process_docx`, `post_process_html`, `post_process_pptx` Post-processing functions that will allow you to customize the display by output type (pdf, html, docx, pptx). They are executed just before printing the table.

`...` unused or deprecated arguments

Value

a list containing previous default values.

See Also

Other themes and defaults: `get_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft_1 <- qflextable(head(airquality))
ft_1

old <- set_flextable_defaults(
  font.color = "#AA8855",
  border.color = "#8855AA"
)
ft_2 <- qflextable(head(airquality))
ft_2

do.call(set_flextable_defaults, old)
```

`set_formatter` *Set column formatter functions*

Description

Apply formatter functions to column keys.

Functions should have a single argument (the vector) and should return the formatted values as a character vector.

Usage

```
set_formatter(x, ..., values = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`...`	Name-value pairs of functions, names should be existing col_key values
`values`	format functions, If values is supplied argument `...` is ignored. • It can be a list of name-value pairs of functions, names should be existing col_key values. • If `values` is a single function, it will be applied to each column.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' is not allowed by the function.

See Also

Other cells formatters: `colformat_char()`, `colformat_date()`, `colformat_datetime()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`

Examples

```
ft <- flextable(head(iris))
ft <- set_formatter(
  x = ft,
  Sepal.Length = function(x) sprintf("%.02f", x),
  Sepal.Width = function(x) sprintf("%.04f", x)
)
ft <- theme_vanilla(ft)
ft
```

set_header_footer_df *Replace the entire header or footer from a data frame*

Description

Replace all header or footer rows using a mapping data frame. Unlike set_header_labels() which only renames the bottom header row, this function rebuilds the entire header (or footer) structure.

The data.frame must contain a column whose values match flextable col_keys argument, this column will be used as join key. The other columns will be displayed as header or footer rows. The leftmost column is used as the top header/footer row and the rightmost column is used as the bottom header/footer row.

Usage

```
set_header_df(x, mapping = NULL, key = "col_keys")

set_footer_df(x, mapping = NULL, key = "col_keys")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
mapping	a data.frame specyfing for each colname content of the column.
key	column to use as key when joigning data_mapping.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_labels(), split_columns(), split_rows(), split_to_pages()

Examples

```
typology <- data.frame(
  col_keys = c(
    "Sepal.Length", "Sepal.Width", "Petal.Length",
    "Petal.Width", "Species"
  ),
  what = c("Sepal", "Sepal", "Petal", "Petal", "Species"),
  measure = c("Length", "Width", "Length", "Width", "Species"),
  stringsAsFactors = FALSE
)

ft_1 <- flextable(head(iris))
ft_1 <- set_header_df(ft_1, mapping = typology, key = "col_keys")
```

```
ft_1 <- merge_h(ft_1, part = "header")
ft_1 <- merge_v(ft_1, j = "Species", part = "header")
ft_1 <- theme_vanilla(ft_1)
ft_1 <- fix_border_issues(ft_1)
ft_1


typology <- data.frame(
  col_keys = c(
    "Sepal.Length", "Sepal.Width", "Petal.Length",
    "Petal.Width", "Species"
  ),
  unit = c("(cm)", "(cm)", "(cm)", "(cm)", ""),
  stringsAsFactors = FALSE
)
ft_2 <- set_footer_df(ft_1, mapping = typology, key = "col_keys")
ft_2 <- italic(ft_2, italic = TRUE, part = "footer")
ft_2 <- theme_booktabs(ft_2)
ft_2 <- fix_border_issues(ft_2)
ft_2
```

`set_header_labels` *Rename column labels in the header*

Description

Change the display labels in the bottom row of the header. Unlike `set_header_df()` which replaces the entire header structure, this function only modifies column labels in the last header row.

Usage

```
set_header_labels(x, ..., values = NULL)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	named arguments (names are data colnames), each element is a single character value specifying label to use.
values	a named list (names are data colnames), each element is a single character value specifying label to use. If provided, argument ... will be ignored. It can also be a unamed character vector, in that case, it must have the same length than the number of columns of the flextable.

See Also

Other row and column operations: `add_body()`, `add_body_row()`, `add_footer()`, `add_footer_lines()`, `add_footer_row()`, `add_header()`, `add_header_lines()`, `add_header_row()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `paginate()`, `separate_header()`, `set_header_footer_df`, `split_columns()`, `split_rows()`, `split_to_pages()`

Examples

```
ft <- flextable(head(iris))
ft <- set_header_labels(ft,
  Sepal.Length = "Sepal length",
  Sepal.Width = "Sepal width", Petal.Length = "Petal length",
  Petal.Width = "Petal width"
)

ft <- flextable(head(iris))
ft <- set_header_labels(ft,
  values = list(
    Sepal.Length = "Sepal length",
    Sepal.Width = "Sepal width",
    Petal.Length = "Petal length",
    Petal.Width = "Petal width"
  )
)
ft

ft <- flextable(head(iris))
ft <- set_header_labels(
  x = ft,
  values = c(
    "Sepal length",
    "Sepal width", "Petal length",
    "Petal width", "Species")
)
ft
```

---

set_table_properties *Set table layout and width properties*

Description

Set table layout and table width. Default to fixed algorithm.

If layout is fixed, column widths will be used to display the table; `width` is ignored.

If layout is autofit, column widths will not be used; table width is used (as a percentage).

Usage

```
set_table_properties(
  x,
  layout = "fixed",
  width = 0,
  align = NULL,
  opts_html = list(),
  opts_word = list(),
  opts_pdf = list(),
```

```
  word_title = NULL,
  word_description = NULL
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
layout	'autofit' or 'fixed' algorithm. Default to 'fixed'.
width	The parameter has a different effect depending on the output format. Users should consider it as a minimum width. In HTML, it is the minimum width of the space that the table should occupy. In Word, it is a preferred size and Word may decide not to strictly stick to it. It has no effect on PowerPoint and PDF output. Its default value is 0, as an effect, it only use necessary width to display all content. It is not used by the PDF output.
align	alignment in document (only Word, HTML and PDF), supported values are 'left', 'center' and 'right'.
opts_html	html options as a list. Supported elements are: • 'extra_css': extra css instructions to be integrated with the HTML code of the table. • 'scroll': NULL or a list if you want to add a scroll-box. – Use an empty list to add an horizontal scroll. The with is fixed, corresponding to the container's width. – If the list has a value named `height` it will be used as height and the scroll will happen also vertically. The height will be in pixel if numeric, if a string it should be a valid css measure. – If the list has a value named `freeze_first_column` set to `TRUE`, the first column is set as a *sticky* column. – If the list has a value named `add_css` it will be used as extra css to add, .i.e: `border:1px solid red;`. • 'extra_class': extra classes to add in the table tag
opts_word	Word options as a list. Supported elements are: • 'split': Word option 'Allow row to break across pages' can be activated when TRUE. • 'keep_with_next': Word option 'keep rows together' is activated when TRUE. It avoids page break within tables. This is handy for small tables, i.e. less than a page height. • 'repeat_headers': Word option 'Repeat as header row' is activated and associated to header rows when TRUE.
opts_pdf	PDF options as a list. Supported elements are: • 'tabcolsep': space between the text and the left/right border of its containing cell. • 'arraystretch': height of each row relative to its default height, the default value is 1.5.

- 'padding_vertical': if TRUE, `padding.top` and `padding.bottom` are applied with the "fixed" layout. The default value is FALSE for backward compatibility (rows height only depends on 'arraystretch'). When activated, `arraystretch = 1` is recommended.
- 'float': type of floating placement in the PDF document, one of:
  - 'none' (the default value), table is placed after the preceding paragraph.
  - 'float', table can float to a place in the text where it fits best
  - 'wrap-r', wrap text around the table positioned to the right side of the text
  - 'wrap-l', wrap text around the table positioned to the left side of the text
  - 'wrap-i', wrap text around the table positioned inside edge-near the binding
  - 'wrap-o', wrap text around the table positioned outside edge-far from the binding
- 'fonts_ignore': if TRUE, pdf-engine 'pdflatex' can be used instead of 'xelatex' or 'lualatex.' If pdflatex is used, fonts will be ignored because they are not supported by pdflatex, whereas with the xelatex and lualatex engines they are.
- 'caption_repeat': a boolean that indicates if the caption should be repeated along pages. Its default value is `TRUE`.
- 'footer_repeat': a boolean that indicates if the footer should be repeated along pages. Its default value is `TRUE`.
- 'default_line_color': default line color, restored globally after the flextable is produced.

`word_title` alternative text for Word table (used as title of the table)

`word_description` alternative text for Word table (used as description of the table)

Note

PowerPoint output ignore 'autofit layout'.

See Also

`flextable()`, `as_flextable()`, `autofit()`, `knit_print.flextable()`

Other functions for flextable size management: `autofit()`, `dim.flextable()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `width()`

Examples

```
library(flextable)
ft_1 <- flextable(head(cars))
ft_1 <- autofit(ft_1)
ft_2 <- set_table_properties(ft_1, width = .5, layout = "autofit")
ft_2
```

```
ft_3 <- set_table_properties(ft_1, width = 1, layout = "autofit")

# add scroll for HTML ----
set.seed(2)
dat <- lapply(1:14, function(x) rnorm(n = 20))
dat <- setNames(dat, paste0("colname", 1:14))
dat <- as.data.frame(dat)

ft_4 <- flextable(dat)
ft_4 <- colformat_double(ft_4)
ft_4 <- bg(ft_4, j = 1, bg = "#DDDDDD", part = "all")
ft_4 <- bg(ft_4, i = 1, bg = "#DDDDDD", part = "header")
ft_4 <- autofit(ft_4)
ft_4 <- set_table_properties(
  x = ft_4,
  opts_html = list(
    scroll = list(
      height = "500px",
      freeze_first_column = TRUE
    )
  )
)
ft_4
```

`shift_table` *Create a shift table*

Description

Create a shift table ready to be used with `tabulator()`.

The function is transforming a dataset representing some 'Laboratory Tests Results' structured as *CDISC clinical trial data sets* format to a dataset representing the shift table.

Shift tables are tables used in clinical trial analysis. They show the progression of change from the baseline, with the progression often being along time; the number of subjects is displayed in different range (e.g. low, normal, or high) at baseline and at selected time points or intervals.

Usage

```
shift_table(
  x,
  cn_visit = "VISIT",
  cn_visit_num = "VISITNUM",
  cn_grade = "LBNRIND",
  cn_usubjid = "USUBJID",
  cn_lab_cat = NA_character_,
  cn_is_baseline = "LBBLFL",
  baseline_identifier = "Y",
  cn_treatment = NA_character_,
```

```
  grade_levels = c("LOW", "NORMAL", "HIGH"),
  grade_labels = c("Low", "Normal", "High")
)
```

Arguments

x	Laboratory Tests Results data frame.
cn_visit	column name containing visit names, default to "VISIT".
cn_visit_num	column name containing visit numbers, default to "VISITNUM".
cn_grade	column name containing reference range indicators, default to "LBNRIND".
cn_usubjid	column name containing unique subject inditifiers, default to "USUBJID".
cn_lab_cat	column name containing lab tests or examination names, default to "LBTEST".
cn_is_baseline	column name containing baseline flags, default to "LBBLFL".
baseline_identifier	baseline flag value to use for baseline identification. Its default is "Y".
cn_treatment	column name containing treatment names, default to NA.
grade_levels	levels to use for reference range indicators
grade_labels	labels to use for reference range indicators

Value

the shift table as a data.frame. Additionnal elements are provided in attributes:

- "VISIT_N": count of unique subject id per visits, labs and eventually treatments. This element is supposed to be used as value for argument `hidden_data` of function `tabulator()`.
- "FUN_VISIT": a utility function to easily turn *visit* column as a factor column. It should be applied after the shift table creation.
- "FUN_GRADE": a utility function to easily turn *grade* column as a factor column. It adds "MISSING/Missing" and "SUM/Sum" at the end of the set of values specified in arguments `grade_levels` and `grade_labels`. It should be applied after the shift table creation.

Examples

```
library(data.table)
library(flextable)

# data simulation ----
USUBJID <- sprintf("01-ABC-%04.0f", 1:200)
VISITS <- c("SCREENING 1", "WEEK 2", "MONTH 3")
LBTEST <- c("Albumin", "Sodium")

VISITNUM <- seq_along(VISITS)
LBBLFL <- rep(NA_character_, length(VISITNUM))
LBBLFL[1] <- "Y"

VISIT <- data.frame(
  VISIT = VISITS, VISITNUM = VISITNUM,
```

```
  LBBLFL = LBBLFL, stringsAsFactors = FALSE
)
labdata <- expand.grid(
  USUBJID = USUBJID, LBTEST = LBTEST,
  VISITNUM = VISITNUM,
  stringsAsFactors = FALSE
)
setDT(labdata)

labdata <- merge(labdata, VISIT, by = "VISITNUM")

subject_elts <- unique(labdata[, .SD, .SDcols = "USUBJID"])
subject_elts <- unique(subject_elts)
subject_elts[, c("TREAT") := list(
  sample(x = c("Treatment", "Placebo"), size = .N, replace = TRUE)
)]
subject_elts[, c("TREAT") := list(
  factor(.SD$TREAT, levels = c("Treatment", "Placebo"))
)]
setDF(subject_elts)
labdata <- merge(labdata, subject_elts,
  by = "USUBJID", all.x = TRUE, all.y = FALSE
)
labdata[, c("LBNRIND") := list(
  sample(
    x = c("LOW", "NORMAL", "HIGH"), size = .N,
    replace = TRUE, prob = c(.03, .9, .07)
  )
)]

setDF(labdata)




# shift table calculation ----

SHIFT_TABLE <- shift_table(
  x = labdata, cn_visit = "VISIT",
  cn_grade = "LBNRIND",
  cn_usubjid = "USUBJID",
  cn_lab_cat = "LBTEST",
  cn_treatment = "TREAT",
  cn_is_baseline = "LBBLFL",
  baseline_identifier = "Y",
  grade_levels = c("LOW", "NORMAL", "HIGH")
)

# get attrs for post treatment ----
SHIFT_TABLE_VISIT <- attr(SHIFT_TABLE, "VISIT_N")
visit_as_factor <- attr(SHIFT_TABLE, "FUN_VISIT")
range_as_factor <- attr(SHIFT_TABLE, "FUN_GRADE")
```

```
# post treatments ----
SHIFT_TABLE$VISIT <- visit_as_factor(SHIFT_TABLE$VISIT)
SHIFT_TABLE$BASELINE <- range_as_factor(SHIFT_TABLE$BASELINE)
SHIFT_TABLE$LBNRIND <- range_as_factor(SHIFT_TABLE$LBNRIND)

SHIFT_TABLE_VISIT$VISIT <- visit_as_factor(SHIFT_TABLE_VISIT$VISIT)

# tabulator ----

my_format <- function(z) {
  formatC(z * 100,
    digits = 1, format = "f",
    flag = "0", width = 4
  )
}

tab <- tabulator(
  x = SHIFT_TABLE,
  hidden_data = SHIFT_TABLE_VISIT,
  row_compose = list(
    VISIT = as_paragraph(VISIT, "\n(N=", N_VISIT, ")")
  ),
  rows = c("LBTEST", "VISIT", "BASELINE"),
  columns = c("TREAT", "LBNRIND"),
  `n` = as_paragraph(N),
  `%` = as_paragraph(as_chunk(PCT, formatter = my_format))
)

# as_flextable ----

ft_1 <- as_flextable(
  x = tab, separate_with = "VISIT",
  label_rows = c(
    LBTEST = "Lab Test", VISIT = "Visit",
    BASELINE = "Reference Range Indicator"
  )
)

ft_1
```

`split_columns` *Split a flextable into pages by columns*

Description

Split a flextable into a list of flextables whose columns fit within a given width (in inches). This is useful for paginating wide tables in Word or PowerPoint output.

Usage

```
split_columns(x, max_width, rep_cols = NULL, unit = "in")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
max_width	Maximum width for each page (in inches by default).
rep_cols	Columns to repeat on every page. Can be a character vector of column names or an integer vector of column positions. NULL (default) means no repetition. Repeated columns appear at the beginning of each page in the order specified.
unit	Unit for max_width, one of "in", "cm", "mm".

Value

A list of flextable objects. If no splitting is needed, a single-element list is returned.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_rows(), split_to_pages()

Examples

```
ft <- flextable(head(mtcars))
ft_pages <- split_columns(ft, max_width = 5)
length(ft_pages)
```

---

split_rows *Split a flextable into pages by rows*

Description

Split a flextable into a list of flextables whose body rows fit within a given height (in inches). Header and footer are repeated on every page. An optional group argument keeps row groups together (no page break inside a group).

Usage

```
split_rows(x, max_height, group = integer(0), unit = "in")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
max_height	Maximum height for each page, including header and footer (in inches by default).

group	Integer vector of body row indices that start a new group. Rows belonging to the same group are kept together on a single page. Default is integer(0) (no grouping, every row is independent).
unit	Unit for max_height, one of "in", "cm", "mm".

Value

A list of flextable objects. If no splitting is needed, a single-element list is returned.

Note

Footnotes are currently repeated on every page, even when they reference rows that only appear on a specific page. This limitation will be resolved in a future version when footnotes are restructured to track their association with body rows.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_to_pages()

Examples

```
ft <- flextable(iris)
ft_pages <- split_rows(ft, max_height = 3)
length(ft_pages)
```

---

split_to_pages *Split a flextable into pages by rows and columns*

Description

Split a flextable into a list of flextables that fit within the given height and width constraints. Rows are split first, then columns are split within each row page.

This is a convenience wrapper around split_rows() and split_columns().

Usage

```
split_to_pages(
  x,
  max_width = NULL,
  max_height = NULL,
  rep_cols = NULL,
  group = integer(0),
  unit = "in"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
max_width	Maximum width per page (in inches by default). NULL means no column splitting.
max_height	Maximum height per page, including header and footer (in inches by default). NULL means no row splitting.
rep_cols	Columns to repeat on every horizontal page. Can be a character vector of column names, an integer vector of column positions, or NULL (default, no repetition).
group	Integer vector of body row indices that start a new group. Default is integer(0).
unit	Unit for max_width and max_height, one of "in", "cm", "mm".

Value

A list of flextable objects.

See Also

Other row and column operations: add_body(), add_body_row(), add_footer(), add_footer_lines(), add_footer_row(), add_header(), add_header_lines(), add_header_row(), delete_columns(), delete_part(), delete_rows(), paginate(), separate_header(), set_header_footer_df, set_header_labels(), split_columns(), split_rows()

Examples

```
ft <- flextable(iris)
ft_pages <- split_to_pages(ft, max_width = 4, max_height = 5)
length(ft_pages)
```

---

style *Set formatting properties on a flextable selection*

Description

style() applies text, paragraph and cell formatting properties to a selection of rows and columns in one call. It is a lower-level function that bundles what the convenience functions do individually:

- text (pr_t): font family, size, color, bold, italic, etc., same properties as bold(), italic(), color(), fontsize(), font(), highlight().
- paragraph (pr_p): alignment, padding, line spacing, same properties as align(), padding(), line_spacing().
- cell (pr_c): background colour, borders, vertical alignment, same properties as bg(), border(), valign().

Use style() when you need to set several property types at once on the same selection; use the convenience functions when you only need to change one aspect.

Note that style() modifies an existing flextable. To change the initial formatting applied to every new flextable, use set_flextable_defaults() instead.

Usage

```
style(
  x,
  i = NULL,
  j = NULL,
  pr_t = NULL,
  pr_p = NULL,
  pr_c = NULL,
  part = "body"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
j	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
pr_t	an `officer::fp_text()` or `officer::fp_text_lite()` object defining text formatting (font, size, color, bold, ...).
pr_p	an `officer::fp_par()` or `officer::fp_par_lite()` object defining paragraph formatting (alignment, padding, line spacing, ...).
pr_c	an `officer::fp_cell()` object defining cell formatting (background, borders, vertical alignment, ...).
part	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `tab_settings()`, `valign()`

Examples

```
library(officer)
def_cell <- fp_cell(border = fp_border(color = "wheat"))

def_par <- fp_par(text.align = "center")

ft <- flextable(head(mtcars))

ft <- style(ft, pr_c = def_cell, pr_p = def_par, part = "all")
ft <- style(ft, ~ drat > 3.5, ~ vs + am + gear + carb,
  pr_t = fp_text(color = "red", italic = TRUE)
)

ft
```

summarizor *Prepare descriptive statistics for flextable*

Description

It performs a univariate statistical analysis of a dataset by group and formats the results so that they can be used with the `tabulator()` function or directly with as_flextable.

Usage

```
summarizor(
  x,
  by = character(),
  overall_label = NULL,
  num_stats = c("mean_sd", "median_iqr", "range"),
  hide_null_na = TRUE,
  use_labels = TRUE
)
```

Arguments

x	dataset
by	columns names to be used as grouping columns
overall_label	label to use as overall label
num_stats	available statistics for numerical columns to show, available options are "mean_sd", "median_iqr" and "range".
hide_null_na	if TRUE (default), NA counts will not be shown when 0.
use_labels	Logical; if TRUE, any column labels or value labels present in the dataset will be used for display purposes. Defaults to TRUE.

Note

This is very first version of the function; be aware it can evolve or change.

See Also

`fmt_summarizor()`, `labelizor()`

Examples

```
z <- summarizor(CO2[-c(1, 4)],
  by = "Treatment",
  overall_label = "Overall"
)
ft_1 <- as_flextable(z)
ft_1
```

```
ft_2 <- as_flextable(z, sep_w = 0, spread_first_col = TRUE)
ft_2

z <- summarizor(CO2[-c(1, 4)])
ft_3 <- as_flextable(z, sep_w = 0, spread_first_col = TRUE)
ft_3
```

---

surround *Surround cells with borders*

Description

surround() draws borders around specific cells, highlighting them individually.

To set borders for the whole table, use border_outer(), border_inner_h() and border_inner_v().

All the following functions also support the row and column selector i and j:

- hline(): set bottom borders (inner horizontal)
- vline(): set right borders (inner vertical)
- hline_top(): set the top border (outer horizontal)
- vline_left(): set the left border (outer vertical)

Usage

```
surround(
  x,
  i = NULL,
  j = NULL,
  border = NULL,
  border.top = NULL,
  border.bottom = NULL,
  border.left = NULL,
  border.right = NULL,
  part = "body"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
border	border (shortcut for top, bottom, left and right)
border.top	border top

border.bottom	border bottom
border.left	border left
border.right	border right
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other borders management: border_inner(), border_inner_h(), border_inner_v(), border_outer(), border_remove(), hline(), hline_bottom(), hline_top(), vline(), vline_left(), vline_right()

Examples

```
library(officer)
library(flextable)

# cell to highlight
vary_i <- 1:3
vary_j <- 1:3

std_border <- fp_border(color = "orange")

ft <- flextable(head(iris))
ft <- border_remove(x = ft)
ft <- border_outer(x = ft, border = std_border)

for (id in seq_along(vary_i)) {
  ft <- bg(
    x = ft,
    i = vary_i[id],
    j = vary_j[id], bg = "yellow"
  )
  ft <- surround(
    x = ft,
    i = vary_i[id],
    j = vary_j[id],
    border.left = std_border,
    border.right = std_border,
    part = "body"
  )
}

ft <- autofit(ft)
ft
# # render
# print(ft, preview = "pptx")
# print(ft, preview = "docx")
# print(ft, preview = "pdf")
# print(ft, preview = "html")
```

tabulator *Create pivot-style summary tables*

Description

It tabulates a data.frame representing an aggregation which is then transformed as a flextable with as_flextable. The function allows to define any display with the syntax of flextable in a table whose layout is showing dimensions of the aggregation across rows and columns.

Usage

```
tabulator(
  x,
  rows,
  columns,
  datasup_first = NULL,
  datasup_last = NULL,
  hidden_data = NULL,
  row_compose = list(),
  ...
)

## S3 method for class 'tabulator'
summary(object, ...)
```

Arguments

x	an aggregated data.frame
rows	column names to use in rows dimensions
columns	column names to use in columns dimensions
datasup_first	additional data that will be merged with table and placed after the columns presenting the row dimensions.
datasup_last	additional data that will be merged with table and placed at the end of the table.
hidden_data	additional data that will be merged with table, the columns are not presented but can be used with compose() or mk_par() function.
row_compose	a list of call to as_paragraph() - these calls will be applied to the row dimensions (the name is used to target the displayed column).
...	named arguments calling function as_paragraph(). The names are used as labels and the values are evaluated when the flextable is created.
object	an object returned by function tabulator().

Value

an object of class tabulator.

Methods (by generic)

- `summary(tabulator)`: call `summary()` to get a data.frame describing mappings between variables and their names in the flextable. This data.frame contains a column named `col_keys` where are stored the names that can be used for further selections.

Note

This is very first version of the function; be aware it can evolve or change.

See Also

`as_flextable.tabulator()`, `summarizor()`, `as_grouped_data()`, `tabulator_colnames()`

Examples

```
## Not run:
set_flextable_defaults(digits = 2, border.color = "gray")

library(data.table)
# example 1 ----
if (require("stats")) {
  dat <- aggregate(breaks ~ wool + tension,
    data = warpbreaks, mean
  )

  cft_1 <- tabulator(
    x = dat, rows = "wool",
    columns = "tension",
    `mean` = as_paragraph(as_chunk(breaks)),
    `(N)` = as_paragraph(as_chunk(length(breaks), formatter = fmt_int))
  )

  ft_1 <- as_flextable(cft_1)
  ft_1
}

# example 2 ----
if (require("ggplot2")) {
  multi_fun <- function(x) {
    list(mean = mean(x), sd = sd(x))
  }

  dat <- as.data.table(ggplot2::diamonds)
  dat <- dat[cut %in% c("Fair", "Good", "Very Good")]

  dat <- dat[, unlist(lapply(.SD, multi_fun),
    recursive = FALSE
  ),
  .SDcols = c("z", "y"),
  by = c("cut", "color")
  ]
```

```
  tab_2 <- tabulator(
    x = dat, rows = "color",
    columns = "cut",
    `z stats` = as_paragraph(as_chunk(fmt_avg_dev(z.mean, z.sd, digit2 = 2))),
    `y stats` = as_paragraph(as_chunk(fmt_avg_dev(y.mean, y.sd, digit2 = 2)))
  )
  ft_2 <- as_flextable(tab_2)
  ft_2 <- autofit(x = ft_2, add_w = .05)
  ft_2
}

# example 3 ----
# data.table version
dat <- melt(as.data.table(iris),
  id.vars = "Species",
  variable.name = "name", value.name = "value"
)
dat <- dat[,
  list(
    avg = mean(value, na.rm = TRUE),
    sd = sd(value, na.rm = TRUE)
  ),
  by = c("Species", "name")
]
# dplyr version
# library(dplyr)
# dat <- iris |>
#   pivot_longer(cols = -c(Species)) |>
#   group_by(Species, name) |>
#   summarise(avg = mean(value, na.rm = TRUE),
#   sd = sd(value, na.rm = TRUE),
#   .groups = "drop")

tab_3 <- tabulator(
  x = dat, rows = c("Species"),
  columns = "name",
  `mean (sd)` = as_paragraph(
    as_chunk(avg),
    " (", as_chunk(sd), ")"
  )
)
ft_3 <- as_flextable(tab_3)
ft_3

init_flextable_defaults()

## End(Not run)
```

---

tabulator_colnames *Column keys of tabulator objects*

Description

The function provides a way to get column keys associated with the flextable corresponding to a `tabulator()` object. It helps in customizing or programing with `tabulator`.

The function is using column names from the original dataset, eventually filters and returns the names corresponding to the selection.

Usage

```
tabulator_colnames(x, columns, ..., type = NULL)
```

Arguments

`x`	a `tabulator()` object
`columns`	column names to look for
`...`	any filter conditions that use variables names, the same than the argument `columns` of function `tabulator()` (`tabulator(columns = c("col1", "col2"))`).
`type`	the type of column to look for, it can be: • 'columns': visible columns, corresponding to names provided in the '...' arguments of your call to 'tabulator()'. • 'hidden': unvisible columns, corresponding to names of the original dataset columns. • 'rows': visible columns used as 'row' content • 'rows_supp': visible columns used as 'rows_supp' content • NULL: any type of column

See Also

`tabulator()`, `as_flextable.tabulator()`

Examples

```
library(flextable)

cancer_dat <- data.frame(
  count = c(
    9L, 5L, 1L, 2L, 2L, 1L, 9L, 3L, 1L, 10L, 2L, 1L, 1L, 2L, 0L, 3L,
    2L, 1L, 1L, 2L, 0L, 12L, 4L, 1L, 7L, 3L, 1L, 5L, 5L, 3L, 10L,
    4L, 1L, 4L, 2L, 0L, 3L, 1L, 0L, 4L, 4L, 2L, 42L, 28L, 19L, 26L,
    19L, 11L, 12L, 10L, 7L, 10L, 5L, 6L, 5L, 0L, 3L, 4L, 3L, 3L,
    1L, 2L, 3L
  ),
  risktime = c(
    157L, 77L, 21L, 139L, 68L, 17L, 126L, 63L, 14L, 102L, 55L,
    12L, 88L, 50L, 10L, 82L, 45L, 8L, 76L, 42L, 6L, 134L, 71L,
    22L, 110L, 63L, 18L, 96L, 58L, 14L, 86L, 42L, 10L, 66L,
    35L, 8L, 59L, 32L, 8L, 51L, 28L, 6L, 212L, 130L, 101L,
    136L, 72L, 63L, 90L, 42L, 43L, 64L, 21L, 32L, 47L, 14L,
    21L, 39L, 13L, 14L, 29L, 7L, 10L
```

```
  ),
  time = rep(as.character(1:7), 3),
  histology = rep(as.character(1:3), 21),
  stage = rep(as.character(1:3), each = 21)
)

datasup_first <- data.frame(
  time = factor(1:7, levels = 1:7),
  zzz = runif(7)
)

z <- tabulator(cancer_dat,
  rows = "time",
  columns = c("histology", "stage"),
  datasup_first = datasup_first,
  n = as_paragraph(as_chunk(count))
)

j <- tabulator_colnames(
  x = z, type = "columns",
  columns = c("n"),
  stage %in% 1
)

src <- tabulator_colnames(
  x = z, type = "hidden",
  columns = c("count"),
  stage %in% 1
)

if (require("scales")) {
  colourer <- col_numeric(
    palette = c("wheat", "red"),
    domain = c(0, 45)
  )
  ft_1 <- as_flextable(z)
  ft_1 <- bg(
    ft_1,
    bg = colourer, part = "body",
    j = j, source = src
  )
  ft_1
}
```

`tab_settings` *Set tabulation marks configuration*

Description

Define tabulation marks configuration. Specifying the positions and types of tabulation marks in table paragraphs helps organize content, especially in clinical tables, by aligning numbers properly.

Usage

```
tab_settings(x, i = NULL, j = NULL, value = TRUE, part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* `i` *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* `j` *parameter* in <Selectors in flextable>.
value	an object generated by `officer::fp_tabs()`.
part	part selector, see section *Part selection with the* `part` *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: `align()`, `bg()`, `bold()`, `color()`, `empty_blanks()`, `font()`, `fontsize()`, `highlight()`, `indentation()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `style()`, `valign()`

Examples

```
library(officer)
library(flextable)

z <- data.frame(
  Statistic = c("Median (Q1 ; Q3)", "Min ; Max"),
  Value = c(
    "\t999.99\t(99.9 ; 99.9)",
    "\t9.99\t(9999.9 ; 99.9)"
  )
)

ts <- fp_tabs(
  fp_tab(pos = 0.4, style = "decimal"),
  fp_tab(pos = 1.4, style = "decimal")
)

zz <- flextable(z)
zz <- tab_settings(zz, j = 2, value = ts)
zz <- width(zz, width = c(1.5, 2))


save_as_docx(zz, path = tempfile(fileext = ".docx"))
```

`theme_alafoli` *Apply alafoli theme*

Description

Apply alafoli theme

Usage

```
theme_alafoli(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_alafoli(ft)
ft
```

theme_apa *Apply APA theme*

Description

Apply theme APA (the stylistic style of the American Psychological Association) to a flextable

Usage

```
theme_apa(x, ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	unused

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(mtcars * 22.22))
ft <- theme_apa(ft)
ft
```

`theme_booktabs` *Apply booktabs theme*

Description

Apply theme booktabs to a flextable

Usage

```
theme_booktabs(x, bold_header = FALSE, ...)
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`bold_header`	header will be bold if TRUE.
`...`	unused

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_booktabs(ft)
ft
```

`theme_borderless` *Apply borderless theme*

Description

Apply theme borderless to a flextable. All borders are removed. Header text is bold, text columns are left aligned, other columns are right aligned.

Usage

```
theme_borderless(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_borderless(ft)
ft
```

theme_box *Apply box theme*

Description

Apply theme box to a flextable

Usage

```
theme_box(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_box(ft)
ft
```

theme_tron *Apply tron theme*

Description

Apply theme tron to a flextable

Usage

```
theme_tron(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_tron(ft)
ft
```

`theme_tron_legacy` *Apply tron legacy theme*

Description

Apply theme tron legacy to a flextable

Usage

```
theme_tron_legacy(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_tron_legacy(ft)
ft
```

theme_vader *Apply Sith Lord Darth Vader theme*

Description

Apply Sith Lord Darth Vader theme to a flextable

Usage

```
theme_vader(x, ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
...	unused

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vanilla()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_vader(ft)
ft
```

`theme_vanilla` *Apply vanilla theme*

Description

Apply theme vanilla to a flextable: The external horizontal lines of the different parts of the table (body, header, footer) are black 2 points thick, the external horizontal lines of the different parts are black 0.5 point thick. Header text is bold, text columns are left aligned, other columns are right aligned.

Usage

```
theme_vanilla(x)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_zebra()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_vanilla(ft)
ft
```

theme_zebra *Apply zebra theme*

Description

Apply theme zebra to a flextable

Usage

```
theme_zebra(
  x,
  odd_header = "#CFCFCF",
  odd_body = "#EFEFEF",
  even_header = "transparent",
  even_body = "transparent"
)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
odd_header, odd_body, even_header, even_body	odd/even colors for table header and body

behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table **immediately**. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the `theme_fun` argument of `set_flextable_defaults()`; be aware that this theme function is applied as the last instruction when calling `flextable()` - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the `post_process_html` argument of `set_flextable_defaults()` (or `post_process_pdf`, `post_process_docx`, `post_process_pptx`) to specify a theme to be applied systematically before the `flextable()` is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

See Also

Other themes and defaults: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_borderless()`, `theme_box()`, `theme_tron()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`

Examples

```
ft <- flextable(head(airquality))
ft <- theme_zebra(ft)
ft
```

`to_html.flextable` *Get HTML code as a string*

Description

Generate HTML code of corresponding flextable as an HTML table or an HTML image.

Usage

```
## S3 method for class 'flextable'
to_html(x, type = c("table", "img"), ...)
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
type	output type. one of "table" or "img".
...	unused

Value

If `type='img'`, the result will be a string containing HTML code of an image tag, otherwise, the result will be a string containing HTML code of a table tag.

See Also

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `wrap_flextable()`

Examples

```
library(officer)
library(flextable)
x <- to_html(as_flextable(cars))
```

use_df_printer *Set data.frame automatic printing as a flextable*

Description

Define `df_printer()` as data.frame print method in an R Markdown document.

In a setup run chunk:

```
flextable::use_df_printer()
```

Usage

```
use_df_printer()
```

See Also

`df_printer()`, `flextable()`

Other flextable configuration: `add_latex_dep()`, `use_flextable_qmd()`, `use_model_printer()`

use_flextable_qmd *Install the flextable-qmd Quarto extension*

Description

Copies the `flextable-qmd` Quarto extension (bundled with flextable) into the `_extensions/` directory of a Quarto project. The extension provides Lua filters that resolve Quarto markdown content produced by `as_qmd()` inside flextable cells for HTML, PDF, Typst and Word (docx) output formats.

After installation, add the filter to your document or project YAML:

```
filters:
  - flextable-qmd
```

This single filter is all that is required for HTML, PDF and Typst (`format: typst`) output.

For Word (docx) output with labelled flextable chunks (e.g. `#| label: tbl-xxx`), add the post-render filter to remove the wrapper table Quarto creates around the flextable (this extra filter is not needed for Typst, where the table is embedded directly in the figure):

```
filters:
  - flextable-qmd
  - at: post-render
    path: _extensions/flextable-qmd/unwrap-float.lua
```

Usage

```
use_flextable_qmd(path = ".", quiet = FALSE)
```

Arguments

path	Path to the Quarto project root. Defaults to the current working directory.
quiet	If TRUE, suppress informational messages.

Value

The path to the installed extension (invisibly).

See Also

as_qmd() for creating Quarto markdown chunks, knit_print.flextable() for rendering options in knitr documents.

Other flextable configuration: add_latex_dep(), use_df_printer(), use_model_printer()

Examples

```
## Not run:
use_flextable_qmd()

## End(Not run)
```

---

use_model_printer *Set automatic flextable printing for models*

Description

Define as_flextable() as print method in an R Markdown document for models of class:

- lm
- glm
- models from package 'lme' and 'lme4'
- htest (t.test, chisq.test, ...)
- gam
- kmeans and pam

In a setup run chunk:

```
flextable::use_model_printer()
```

Usage

```
use_model_printer()
```

See Also

use_df_printer(), flextable()

Other flextable configuration: add_latex_dep(), use_df_printer(), use_flextable_qmd()

---

valign *Set vertical alignment*

Description

Change the vertical alignment of selected rows and columns of a flextable.

Usage

```
valign(x, i = NULL, j = NULL, valign = "center", part = "body")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
i	row selector, see section *Row selection with the* i *parameter* in <Selectors in flextable>.
j	column selector, see section *Column selection with the* j *parameter* in <Selectors in flextable>.
valign	vertical alignment of paragraph within cell, one of "center" or "top" or "bottom".
part	part selector, see section *Part selection with the* part *parameter* in <Selectors in flextable>. Value 'all' can be used.

See Also

Other formatting shortcuts: align(), bg(), bold(), color(), empty_blanks(), font(), fontsize(), highlight(), indentation(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), style(), tab_settings()

Examples

```
ft_1 <- flextable(iris[c(1:3, 51:53, 101:103), ])
ft_1 <- theme_box(ft_1)
ft_1 <- merge_v(ft_1, j = 5)
ft_1

ft_2 <- valign(ft_1, j = 5, valign = "top", part = "all")
ft_2
```

vline *Set vertical borders to the right of selected columns*

Description

`vline()` draws a vertical line to the **right** of each selected column by setting the right border of cells at column `j` (and the left border of cells at column `j + 1` so that the line renders consistently across output formats).

Use the `j` selector to target specific columns. When `j` is `NULL` (the default) the border is added to the right of every column, producing a full grid of inner vertical lines.

For the **outer** edges of the table, use `vline_left()` and `vline_right()` instead; those always target the very first or very last column.

Usage

```
vline(x, i = NULL, j = NULL, border = NULL, part = "all")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline_left()`, `vline_right()`

Examples

```
library(officer)
std_border <- fp_border(color = "orange")

ft <- flextable(head(iris))
ft <- border_remove(x = ft)

# add vertical borders to the right of every column
ft <- vline(ft, border = std_border)
ft
```

vline_left *Set the left border of the table*

Description

`vline_left()` draws a vertical line along the **left edge** of the table by setting the left border of the first column. It does not accept a column selector `j` because it always targets column 1.

An optional row selector `i` lets you restrict the line to specific rows (e.g. only the body, or only certain rows).

Unlike `vline()`, which adds inner lines to the right of arbitrary columns, `vline_left()` is meant for the outer left edge of the table.

Usage

```
vline_left(x, i = NULL, border = NULL, part = "all")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline()`, `vline_right()`

Examples

```
library(officer)
std_border <- fp_border(color = "orange")

ft <- flextable(head(iris))
ft <- border_remove(x = ft)

# add a border on the left edge of the table
ft <- vline_left(ft, border = std_border)
ft
```

vline_right *Set the right border of the table*

Description

`vline_right()` draws a vertical line along the **right edge** of the table by setting the right border of the last column. It does not accept a column selector `j` because it always targets the last column.

An optional row selector `i` lets you restrict the line to specific rows.

Unlike `vline()`, which adds inner lines to the right of arbitrary columns, `vline_right()` is meant for the outer right edge of the table.

Usage

```
vline_right(x, i = NULL, border = NULL, part = "all")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`i`	row selector, see section *Row selection with the* `i` *parameter* in <`Selectors in flextable`>.
`border`	border properties defined by a call to `officer::fp_border()`
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other borders management: `border_inner()`, `border_inner_h()`, `border_inner_v()`, `border_outer()`, `border_remove()`, `hline()`, `hline_bottom()`, `hline_top()`, `surround()`, `vline()`, `vline_left()`

Examples

```
library(officer)
std_border <- fp_border(color = "orange")

ft <- flextable(head(iris))
ft <- border_remove(x = ft)

# add a border on the right edge of the table
ft <- vline_right(ft, border = std_border)
ft
```

`void` *Clear the displayed content of selected columns*

Description

`void()` replaces the visible text of the selected columns with an empty string. The columns themselves (and their headers) remain in the table, but the cell values are no longer displayed.

This is useful when a column should stay in the layout (e.g. to preserve its width or to keep its header label) but its body values should be hidden, for instance after using `compose()` to build a richer display in a neighbouring column that already incorporates those values.

The underlying dataset is not modified; only the displayed content is affected. To remove a column entirely, use the `col_keys` argument of `flextable()` instead.

Usage

```
void(x, j = NULL, part = "body")
```

Arguments

`x`	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
`j`	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
`part`	part selector, see section *Part selection with the* `part` *parameter* in <`Selectors in flextable`>. Value 'all' can be used.

See Also

Other functions to compose cell content: `append_chunks()`, `as_paragraph()`, `compose()`, `footnote()`, `labelizor()`, `prepend_chunks()`

Examples

```
ftab <- flextable(head(mtcars))
ftab <- void(ftab, ~ vs + am + gear + carb)
ftab
```

width *Set columns width*

Description

Defines the widths of one or more columns in the table. This function will have no effect if you have used `set_table_properties(layout = "autofit")`.

`set_table_properties()` can provide an alternative to fixed-width layouts that is supported with HTML and Word output that can be set with `set_table_properties(layout = "autofit")`.

Usage

```
width(x, j = NULL, width, unit = "in")
```

Arguments

x	a 'flextable' object, see flextable-package to learn how to create 'flextable' object.
j	column selector, see section *Column selection with the* `j` *parameter* in <`Selectors in flextable`>.
width	width in inches
unit	unit for width, one of "in", "cm", "mm".

Details

Heights are not used when flextable is been rendered into HTML.

See Also

Other functions for flextable size management: `autofit()`, `dim.flextable()`, `dim_pretty()`, `fit_columns()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`

Examples

```
ft <- flextable(head(iris))
ft <- width(ft, width = 1.5)
ft
```

wrap_flextable *Wrap a flextable for use with patchwork*

Description

This function wraps a flextable as a patchwork-compliant patch, similar to what `patchwork::wrap_table()` does for gt tables. It allows flextable objects to be combined with ggplot2 plots in a patchwork layout, with optional alignment of table headers and body with plot panel areas.

Note this is experimental and may change in the future.

Usage

```
wrap_flextable(
  x,
  panel = c("body", "full", "rows", "cols"),
  space = c("free", "free_x", "free_y", "fixed"),
  n_row_headers = 0L,
  flex_body = FALSE,
  flex_cols = FALSE,
  expand = 0.6,
  just = c("left", "right", "center")
)
```

Arguments

x	A flextable object.
panel	What portion of the table should be aligned with the panel region? `"body"` means that header and footer will be placed outside the panel region. `"full"` means that the whole table will be placed inside the panel region. `"rows"` keeps all rows inside the panel but is otherwise equivalent to `"body"`. `"cols"` places all columns within the panel region but keeps column headers on top.
space	How should the dimension of the table influence the final composition? `"fixed"` means that the table width and height will set the dimensions of the area it occupies. `"free"` means the table dimensions will not influence the sizing. `"free_x"` and `"free_y"` allow freeing either direction.
n_row_headers	Number of leading columns to treat as row headers. These columns will be placed outside the panel region and will not participate in alignment with the plot axes.
flex_body	If `TRUE`, the table body row heights become flexible: the adjacent ggplot determines the panel height and the body rows stretch equally. With `panel = "body"` or `"cols"`, each row is centered on its category of a discrete y axis (see `expand`), header and footer keep their fixed size above and below the panel. With `panel = "full"` or `"rows"`, the whole table fills the panel: body rows take the height left by header and footer and are not aligned with the categories. This is useful to align table rows with discrete bars or categories in a neighbouring plot. Implies `free_y` for `space`.

`flex_cols` If `TRUE`, the data column widths (all columns after `n_row_headers`) become flexible: the adjacent ggplot determines the panel width and the data columns stretch to fill it. Row header columns keep their fixed width. This is useful to align table columns with discrete x-axis categories in a neighbouring plot. Implies `free_x` for `space`.

`expand` Expansion value matching the ggplot discrete axis expansion (`ggplot2::expansion(add = expand)`). Default is `0.6`, which is the ggplot2 default for discrete axes. Used when `flex_body = TRUE` (with `panel = "body"` or `"cols"`) or `flex_cols = TRUE`. With `0.5`, rows (or columns) fill the whole panel.

`just` Horizontal alignment of the table within its patchwork panel. One of `"left"` (default), `"right"`, or `"center"`. Useful when the table is narrower than the available panel width. Ignored when `flex_cols = TRUE` (columns fill the panel).

Value

A patchwork-compliant object that can be combined with ggplot2 plots using `+`, `|`, or `/` operators.

Placement and stretching

Two arguments set how the table relates to the panel of the plot (its plotting area, without axes and legends):

- `panel` sets **where** the table is placed: which part of the table is aligned with the panel (`"body"`: the body, header above and footer below; `"full"`: the whole table).
- `flex_body` sets **whether body rows stretch** to fill the panel. With `FALSE`, rows keep their natural height and the body may be shorter or taller than the plot. With `TRUE`, the plot sets the height and body rows share it.

To align each row with a category of a discrete y axis, use both `panel = "body"` and `flex_body = TRUE`:

	`flex_body = FALSE`	`flex_body = TRUE`
`panel = "body"`	body at the top of the panel, natural height, not aligned	rows aligned with the categories
`panel = "full"`	table at the top of the panel, natural height	table fills the height of the panel, not aligned

The same applies to columns: `flex_cols = TRUE` with `panel = "body"` or `"cols"` aligns data columns with the categories of a discrete x axis.

See Also

Other functions for flextable output and export: `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `repair_docx()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`

Examples

```
library(gdtools)
font_set_liberation()
library(ggplot2)
library(patchwork)

set_flextable_defaults(
  font.family = "Liberation Sans",
  font.size = 10,
  big.mark = "",
  border.color = "grey60"
)

# Adapted from <https://r-graph-gallery.com/web-dumbell-chart.html>

dataset <- data.frame(
  team = c(
    "FC Bayern Munchen", "SV Werder Bremen", "Borussia Dortmund",
    "VfB Stuttgart", "Borussia M'gladbach", "Hamburger SV",
    "Eintracht Frankfurt", "FC Schalke 04", "1. FC Koln",
    "Bayer 04 Leverkusen"
  ),
  matches = c(2000, 1992, 1924, 1924, 1898, 1866, 1856, 1832, 1754, 1524),
  won     = c(1206,  818,  881,  782,  763,  746,  683,  700,  674,  669),
  lost    = c( 363,  676,  563,  673,  636,  625,  693,  669,  628,  447)
)
dataset$win_pct  <- dataset$won  / dataset$matches * 100
dataset$loss_pct <- dataset$lost / dataset$matches * 100
dataset$team <- factor(dataset$team, levels = rev(dataset$team))

# -- dumbbell chart --
pal <- c(lost = "#EFAC00", won = "#28A87D")
df_long <- reshape(dataset, direction = "long",
  varying = list(c("loss_pct", "win_pct")),
  v.names = "pct", timevar = "type",
  times = c("lost", "won"), idvar = "team"
)

p <- ggplot(df_long, aes(x = pct / 100, y = team)) +
  stat_summary(
    geom = "linerange", fun.min = "min", fun.max = "max",
    linewidth = .7, color = "grey60"
  ) +
  geom_point(aes(fill = type), size = 4, shape = 21,
    stroke = .8, color = "white"
  ) +
  scale_x_continuous(
    labels = scales::percent,
    expand = expansion(add = c(.02, .02))
  ) +
  scale_y_discrete(name = NULL, guide = "none") +
  scale_fill_manual(
```

```
    values = pal,
    labels = c(lost = "Lost", won = "Won")
  ) +
  labs(x = NULL, fill = NULL) +
  theme_minimal(base_family = "Liberation Sans", base_size = 10) +
  theme(
    legend.position = "top",
    legend.justification = "left",
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_blank()
  )

# -- flextable --
ft_dat <- dataset[, c("matches", "win_pct", "loss_pct", "team")]
ft_dat$team <- as.character(ft_dat$team)

ft <- flextable(ft_dat)
ft <- border_remove(ft)
ft <- bold(ft, part = "header")
ft <- colformat_double(ft, j = c("win_pct", "loss_pct"),
  digits = 1, suffix = "%"
)
ft <- set_header_labels(ft,
  team = "Team", matches = "GP",
  win_pct = "", loss_pct = ""
)
ft <- color(ft, color = "#28A87D", j = 2)
ft <- color(ft, color = "#EFAC00", j = 3)
ft <- bold(ft, bold = TRUE, j = 2:3)
ft <- italic(ft, italic = TRUE, j = 4)
ft <- align(ft, align = "right", part = "all")
ft <- autofit(ft)


print(
  wrap_flextable(ft, flex_body = TRUE, just = "right") +
    p + plot_layout(widths = c(1.1, 2))
)
```

Index

∗ **as_flextable methods**
as_flextable.compact_summary, 24
as_flextable.data.frame, 25
as_flextable.gam, 26
as_flextable.glm, 27
as_flextable.grouped_data, 28
as_flextable.htest, 29
as_flextable.kmeans, 30
as_flextable.lm, 31
as_flextable.merMod, 31
as_flextable.pam, 33
as_flextable.summarizor, 33
as_flextable.table, 34
as_flextable.tabular, 36
as_flextable.tabulator, 38
as_flextable.xtable, 41
compact_summary, 76
∗ **borders management**
border_inner, 61
border_inner_h, 61
border_inner_v, 62
border_outer, 63
border_remove, 64
hline, 114
hline_bottom, 115
hline_top, 116
surround, 182
vline, 204
vline_left, 205
vline_right, 206
∗ **cell merging functions**
merge_at, 129
merge_h, 130
merge_h_range, 131
merge_none, 132
merge_v, 132
∗ **cell_content_composition**
append_chunks, 19
as_paragraph, 46
compose, 77
footnote, 101
labelizor, 126
prepend_chunks, 144
void, 207
∗ **cells formatters**
colformat_char, 65
colformat_date, 66
colformat_datetime, 67
colformat_double, 68
colformat_image, 69
colformat_int, 70
colformat_lgl, 71
colformat_num, 72
set_formatter, 167
∗ **chunk elements for paragraph**
as_b, 20
as_bracket, 21
as_chunk, 22
as_equation, 23
as_highlight, 43
as_i, 44
as_image, 45
as_qmd, 47
as_strike, 49
as_sub, 50
as_sup, 51
as_word_field, 52
colorize, 75
gg_chunk, 109
grid_chunk, 110
hyperlink_text, 119
linerange, 127
minibar, 134
plot_chunk, 142
∗ **default_formatting_properties**
fp_border_default, 103
fp_text_default, 104
∗ **flextable_configuration**

use_df_printer, 201
use_flextable_qmd, 201
use_model_printer, 202
∗ **flextable_constructors**
as_flextable, 24
flextable, 88
∗ **flextable_output_export**
df_printer, 82
flextable_to_rmd, 90
gen_grob, 106
htmltools_value, 118
knit_print.flextable, 123
plot.flextable, 141
print.flextable, 145
repair_docx, 147
save_as_docx, 151
save_as_html, 152
save_as_image, 153
save_as_pptx, 154
save_as_rtf, 155
to_html.flextable, 200
wrap_flextable, 209
∗ **formatting_shortcuts**
align, 18
bg, 55
bold, 60
color, 74
empty_blanks, 85
font, 98
fontsize, 100
highlight, 113
indentation, 120
italic, 121
keep_with_next, 122
line_spacing, 129
padding, 136
rotate, 148
style, 179
tab_settings, 188
valign, 203
∗ **functions for flextable size management**
autofit, 53
dim.flextable, 83
dim_pretty, 84
fit_columns, 86
fit_to_width, 87
flextable_dim, 90
height, 111
hrule, 117
ncol_keys, 135
nrow_part, 136
set_table_properties, 170
width, 208
∗ **officer_integration**
body_add_flextable, 57
body_replace_flextable_at_bkm, 59
ph_with.flextable, 140
rtf_add.flextable, 150
∗ **table_structure**
add_body, 8
add_body_row, 9
add_footer, 10
add_footer_lines, 12
add_footer_row, 12
add_header, 14
add_header_lines, 15
add_header_row, 16
delete_columns, 79
delete_part, 80
delete_rows, 81
paginate, 138
separate_header, 158
set_header_footer_df, 168
set_header_labels, 169
split_columns, 176
split_rows, 177
split_to_pages, 178
∗ **themes_and_defaults**
get_flextable_defaults, 108
set_flextable_defaults, 163
theme_alafoli, 190
theme_apa, 191
theme_booktabs, 192
theme_borderless, 193
theme_box, 194
theme_tron, 195
theme_tron_legacy, 196
theme_vader, 197
theme_vanilla, 198
theme_zebra, 199
∗ **tools for clinical reporting**
shift_table, 173
∗ **value formatters**
fmt_2stats, 91
fmt_avg_dev, 93
fmt_dbl, 93

fmt_header_n, 94
fmt_int, 95
fmt_n_percent, 96
fmt_pct, 97
fmt_signif_after_zeros, 98

add_body, 8
add_body(), *7*, *10–13*, *15–17*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_body_row, 9
add_body_row(), *7, 8*, *11–13*, *15–17*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_footer, 10
add_footer(), *7, 8*, *10*, *12, 13*, *15–17*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_footer_lines, 12
add_footer_lines(), *7, 8*, *10, 11*, *13*, *15–17*, *47*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_footer_row, 12
add_footer_row(), *7, 8*, *10–12*, *15–17*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_header, 14
add_header(), *7, 8*, *10–13*, *16, 17*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_header_lines, 15
add_header_lines(), *7, 8*, *10–13*, *15*, *17*, *47*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_header_row, 16
add_header_row(), *7, 8*, *10–16*, *79–81*, *139*, *158*, *168, 169*, *177–179*
add_latex_dep(), *125*, *201–203*
align, 18
align(), *56*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
align_nottext_col *(*align*)*, 18
align_text_col *(*align*)*, 18
append_chunks, 19
append_chunks(), *20–23*, *43–45*, *47*, *49–52*, *75*, *77*, *102*, *105*, *109, 110*, *119*, *126, 127*, *134*, *142*, *144*, *207*
as_b, 20
as_b(), *21–23*, *44*, *46*, *49–52*, *75*, *77*, *109*, *111*, *119*, *128*, *135*, *143*
as_bracket, 21
as_bracket(), *20*, *22, 23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *128*, *135*, *143*
as_chunk, 22
as_chunk(), *19–21*, *23*, *44*, *46, 47*, *49–52*, *75*, *77*, *104, 105*, *109*, *111*, *119*, *128*, *135*, *143, 144*
as_equation, 23
as_equation(), *20–22*, *44*, *46*, *49–52*, *75*, *106*, *109*, *111*, *119*, *128*, *135*, *143*
as_flextable, 24, *181*, *184*
as_flextable(), *6*, *24*, *33*, *38*, *76*, *89*, *172*, *202*
as_flextable.brmsfit *(*as_flextable.merMod*)*, 31
as_flextable.compact_summary, 24
as_flextable.compact_summary(), *26–35*, *37*, *39*, *41*, *76*
as_flextable.data.frame, 25
as_flextable.data.frame(), *25*, *27–35*, *37*, *39*, *41*, *76*
as_flextable.ElementaryTable *(*as_flextable.TableTree*)*, 35
as_flextable.gam, 26
as_flextable.gam(), *25–35*, *37*, *39*, *41*, *76*
as_flextable.glm, 27
as_flextable.glm(), *25–35*, *37*, *39*, *41*, *76*
as_flextable.glmmadmb *(*as_flextable.merMod*)*, 31
as_flextable.glmmTMB *(*as_flextable.merMod*)*, 31
as_flextable.gls *(*as_flextable.merMod*)*, 31
as_flextable.grouped_data, 28
as_flextable.grouped_data(), *25–27*, *29–35*, *37*, *39*, *41*, *43*, *76*
as_flextable.htest, 29
as_flextable.htest(), *25–28*, *30–35*, *37*, *39*, *41*, *76*
as_flextable.kmeans, 30
as_flextable.kmeans(), *25–29*, *31–35*, *37*, *39*, *41*, *76*
as_flextable.lm, 31
as_flextable.lm(), *25–30*, *32–35*, *37*, *39*, *41*, *76*
as_flextable.lme *(*as_flextable.merMod*)*, 31
as_flextable.merMod, 31
as_flextable.merMod(), *25–31*, *33–35*, *37*, *39*, *41*, *76*
as_flextable.nlme *(*as_flextable.merMod*)*, 31

as_flextable.pam, 33
as_flextable.pam(), *25–32*, *34, 35*, *37*, *39*, *41*, *76*
as_flextable.summarizor, 33
as_flextable.summarizor(), *25–33*, *35*, *37*, *39*, *41*, *76*
as_flextable.table, 34
as_flextable.table(), *25–34*, *37*, *39*, *41*, *76*
as_flextable.TableTree, 35
as_flextable.tabular, 36
as_flextable.tabular(), *25–35*, *39*, *41*, *76*, *138*
as_flextable.tabulator, 38
as_flextable.tabulator(), *25–35*, *37*, *41*, *76*, *185*, *187*
as_flextable.xtable, 41
as_flextable.xtable(), *25–35*, *37*, *39*, *76*
as_grouped_data, 42
as_grouped_data(), *28*, *39*, *185*
as_highlight, 43
as_highlight(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *128*, *135*, *143*
as_i, 44
as_i(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *128*, *135*, *143*
as_image, 45
as_image(), *20–23*, *44*, *46, 47*, *49–52*, *75*, *77*, *109*, *111*, *119*, *128*, *135*, *143*, *148*
as_paragraph, 46
as_paragraph(), *6*, *9*, *12, 13*, *15–17*, *20*, *22*, *37*, *45, 46*, *48*, *51*, *58*, *77*, *101, 102*, *109, 110*, *126–128*, *134, 135*, *142*, *144*, *159, 160*, *184*, *207*
as_qmd, 47
as_qmd(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *124, 125*, *128*, *135*, *143*, *201, 202*
as_strike, 49
as_strike(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *128*, *135*, *143*
as_sub, 50
as_sub(), *20–23*, *44*, *46*, *49*, *51, 52*, *75*, *109*, *111*, *119*, *128*, *135*, *143*
as_sup, 51
as_sup(), *20–23*, *44*, *46*, *49, 50*, *52*, *75*, *77*, *109*, *111*, *119*, *128*, *135*, *143*
as_word_field, 52
as_word_field(), *20–23*, *44*, *46*, *49–51*, *75*, *77*, *109*, *111*, *119*, *128*, *135*, *143*
autofit, 53
autofit(), *26*, *83*, *85–87*, *89, 90*, *112*, *117*, *125*, *135, 136*, *140*, *149*, *172*, *208*

before, 54
bg, 55
bg(), *18*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
body_add_flextable, 57
body_add_flextable(), *60*, *141*, *151*, *161*
body_replace_flextable_at_bkm, 59
body_replace_flextable_at_bkm(), *58*, *141*, *151*
bold, 60
bold(), *18*, *56*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
border(), *179*
border_inner, 61
border_inner(), *62–64*, *114–116*, *183*, *204–206*
border_inner_h, 61
border_inner_h(), *61*, *63, 64*, *114–116*, *182, 183*, *204–206*
border_inner_v, 62
border_inner_v(), *61–64*, *114–116*, *182, 183*, *204–206*
border_outer, 63
border_outer(), *61–64*, *114–116*, *182, 183*, *204–206*
border_remove, 64
border_remove(), *61–63*, *114–116*, *183*, *204–206*

cluster::pam(), *33*
colformat_char, 65
colformat_char(), *66–69*, *71–73*, *167*
colformat_date, 66
colformat_date(), *65*, *67–69*, *71–73*, *167*
colformat_datetime, 67
colformat_datetime(), *65, 66*, *68, 69*, *71–73*, *167*
colformat_double, 68
colformat_double(), *65–67*, *69*, *71–73*, *165*, *167*
colformat_image, 69
colformat_image(), *65–68*, *71–73*, *148*, *167*
colformat_int, 70

colformat_int(), *65–69*, *72, 73*, *165*, *167*
colformat_lgl, 71
colformat_lgl(), *65–69*, *71*, *73*, *167*
colformat_num, 72
colformat_num(), *8*, *11*, *14*, *65–69*, *71, 72*, *165*, *167*
color, 74
color(), *18*, *56*, *60*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
colorize, 75
colorize(), *20–23*, *44*, *46*, *49–52*, *109*, *111*, *119*, *128*, *135*, *143*
compact_summary, 76
compact_summary(), *25–35*, *37*, *39*, *41*
compose, 77
compose(), *20–23*, *43–47*, *49–52*, *75*, *89*, *102*, *105*, *109, 110*, *119*, *126–128*, *134, 135*, *142*, *144*, *184*, *207*
continuous_summary, 78

delete_columns, 79
delete_columns(), *7, 8*, *10–13*, *15–17*, *80, 81*, *139*, *158*, *168, 169*, *177–179*
delete_part, 80
delete_part(), *7, 8*, *10–13*, *15–17*, *79*, *81*, *139*, *158*, *168, 169*, *177–179*
delete_rows, 81
delete_rows(), *8*, *10–13*, *15–17*, *79, 80*, *139*, *158*, *168, 169*, *177–179*
df_printer, 82
df_printer(), *91*, *108*, *118*, *125*, *141*, *145*, *148*, *152–156*, *200, 201*, *210*
dim, *107*
dim.flextable, 83
dim.flextable(), *54*, *85*, *87*, *90*, *112*, *117*, *135, 136*, *172*, *208*
dim.flextableGrob, 84
dim_pretty, 84, *149*
dim_pretty(), *54*, *83*, *86, 87*, *90*, *112*, *117*, *135, 136*, *140*, *172*, *208*

empty_blanks, 85
empty_blanks(), *18*, *56*, *60*, *74*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *180*, *189*, *203*

fit_columns, 86
fit_columns(), *53, 54*, *83*, *85*, *87*, *90*, *112*, *117*, *135, 136*, *172*, *208*
fit_to_width, 87
fit_to_width(), *53, 54*, *83*, *85–87*, *90*, *112*, *117*, *135, 136*, *172*, *208*
flextable, 88
flextable(), *8*, *10*, *13*, *17*, *24, 25*, *163*, *172*, *201*, *203*, *207*
flextable-package, 6, *8, 9*, *11–15*, *17–19*, *54*, *56*, *60–72*, *74*, *77*, *79–81*, *83–87*, *90, 91*, *99–101*, *106*, *112–118*, *120–123*, *126*, *129–133*, *135–138*, *141*, *144, 145*, *149*, *154*, *158*, *160*, *167–169*, *171*, *177*, *179, 180*, *182*, *189–200*, *203–208*
flextable_dim, 90
flextable_dim(), *54*, *83*, *85*, *87*, *112*, *117*, *135, 136*, *172*, *208*
flextable_to_rmd, 90
flextable_to_rmd(), *83*, *108*, *118*, *125*, *141*, *145*, *148*, *152–156*, *200*, *210*
fmt_2stats, 91
fmt_2stats(), *93–98*
fmt_avg_dev, 93
fmt_avg_dev(), *92*, *94–98*
fmt_dbl, 93
fmt_dbl(), *92–98*
fmt_header_n, 94
fmt_header_n(), *92–98*
fmt_int, 95, *146*
fmt_int(), *92–94*, *96–98*
fmt_n_percent, 96
fmt_n_percent(), *92–95*, *97, 98*
fmt_pct, 97
fmt_pct(), *92–96*, *98*
fmt_signif_after_zeros, 98
fmt_signif_after_zeros(), *92–97*
fmt_summarizor *(*fmt_2stats*)*, 91
fmt_summarizor(), *181*
font, 98
font(), *18*, *56*, *60*, *74*, *85*, *100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
fontsize, 100
fontsize(), *18*, *56*, *60*, *74*, *85*, *99*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
footnote, 101
footnote(), *20*, *47*, *77*, *89*, *126*, *144*, *207*

format(), *70*, *72, 73*
formatC, *165*
formatC(), *24*, *68*, *73*
formatters::matrix_form(), *35*
fp_border_default, 103
fp_border_default(), *39*, *105*
fp_text_default, 104
fp_text_default(), *22, 23*, *52*, *77*, *103*, *119*

gdtools::register_gfont(), *98*
gen_grob, 106
gen_grob(), *83*, *91*, *118*, *125*, *141*, *145*, *148*, *152–156*, *200*, *210*
get_flextable_defaults, 108
get_flextable_defaults(), *89*, *163*, *166*, *190–199*
gg_chunk, 109
gg_chunk(), *19–23*, *44*, *46*, *49–52*, *75*, *111*, *119*, *128*, *135*, *143, 144*, *148*
grid::grid.layout(), *107*
grid_chunk, 110
grid_chunk(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *119*, *128*, *135*, *143*, *148*

height, 111
height(), *54*, *83*, *85*, *87*, *90*, *117*, *135, 136*, *140*, *149*, *172*, *208*
height_all *(*height*)*, 111
highlight, 113
highlight(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*
hline, 114
hline(), *54, 55*, *61–64*, *103*, *115, 116*, *182, 183*, *204–206*
hline_bottom, 115
hline_bottom(), *61–64*, *114*, *116*, *183*, *204–206*
hline_top, 116
hline_top(), *61–64*, *114, 115*, *182, 183*, *204–206*
hrule, 117
hrule(), *54*, *83*, *85*, *87*, *90*, *111, 112*, *135, 136*, *149*, *172*, *208*
htmltools::div(), *118*
htmltools::HTML, *118*
htmltools_value, 118
htmltools_value(), *83*, *91*, *108*, *125*, *141*, *145*, *148*, *152–156*, *200*, *210*
hyperlink_text, 119
hyperlink_text(), *20–23*, *44*, *46, 47*, *49–52*, *75*, *109*, *111*, *128*, *135*, *143*

indentation, 120
indentation(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *180*, *189*, *203*
init_flextable_defaults *(*set_flextable_defaults*)*, 163
init_flextable_defaults(), *89*
italic, 121
italic(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *149*, *179, 180*, *189*, *203*

keep_with_next, 122
keep_with_next(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *121*, *129*, *137*, *149*, *180*, *189*, *203*
kmeans(), *30*
knit_print.flextable, *90*, *118*, 123, *162*
knit_print.flextable(), *49*, *58*, *83*, *88, 89*, *91*, *108*, *118*, *139*, *141*, *145*, *148*, *152–156*, *163*, *172*, *200*, *202*, *210*

labelizor, 126
labelizor(), *20*, *47*, *77*, *102*, *144*, *181*, *207*
line_spacing, 129
line_spacing(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *137*, *149*, *179, 180*, *189*, *203*
linerange, 127
linerange(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *135*, *143*, *148*

merge_at, 129
merge_at(), *130–133*
merge_h, 130
merge_h(), *14*, *130–133*
merge_h_range, 131
merge_h_range(), *130*, *132, 133*
merge_none, 132
merge_none(), *130, 131*, *133*
merge_v, 132
merge_v(), *14*, *130–132*
minibar, 134
minibar(), *20–23*, *44*, *46, 47*, *49–52*, *75*, *109*, *111*, *119*, *128*, *143*, *148*

mk_par *(compose)*, 77
mk_par(), *6*, *47*, *92–98*, *126*, *184*

ncol_keys, 135
ncol_keys(), *54*, *83*, *85*, *87*, *90*, *112*, *117*, *136*, *172*, *208*
nrow_part, 136
nrow_part(), *54*, *83*, *85*, *87*, *90*, *112*, *117*, *135*, *172*, *208*

officer::fp_border(), *39*, *61–63*, *103*, *114–116*, *204–206*
officer::fp_cell(), *180*
officer::fp_par(), *37*, *160*, *180*
officer::fp_par_lite(), *180*
officer::fp_tabs(), *189*
officer::fp_text(), *22, 23*, *52*, *104*, *119*, *180*
officer::fp_text_lite(), *180*
officer::hyperlink_ftext(), *106*
officer::ph_location_type(), *140*
officer::prop_section, *151*, *156*
officer::read_pptx(), *140*
officer::rtf_add(), *150*
officer::rtf_doc(), *150*
officer::run_autonum(), *160*
officer::styles_info(), *160*

padding, 136
padding(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *120–122*, *129*, *149*, *165*, *179, 180*, *189*, *203*
paginate, 138
paginate(), *8*, *10–13*, *15–17*, *35, 36*, *58*, *79–81*, *122*, *125*, *152*, *156*, *158*, *165*, *168, 169*, *177–179*
par(), *143*
patchwork::wrap_table(), *209*
ph_with.flextable, 140
ph_with.flextable(), *58*, *60*, *151*
plot.flextable, 141
plot.flextable(), *83*, *91*, *108*, *118*, *125*, *145*, *148*, *152–156*, *200*, *210*
plot.flextableGrob, 142
plot_chunk, 142
plot_chunk(), *20–23*, *44*, *46*, *49–52*, *75*, *109*, *111*, *119*, *128*, *135*, *148*
prepend_chunks, 144
prepend_chunks(), *20–23*, *43–45*, *47*, *49–52*, *75*, *77*, *102*, *105*, *109, 110*, *119*, *126, 127*, *134*, *142*, *207*
print.flextable, 145
print.flextable(), *83*, *91*, *108*, *118*, *125*, *141*, *148*, *152–156*, *200*, *210*
proc_freq, 146
proc_freq(), *34*

qflextable *(flextable)*, 88

repair_docx, 147
repair_docx(), *46*, *83*, *91*, *108, 109*, *111*, *118, 119*, *125*, *128*, *134*, *141*, *143*, *145*, *152–156*, *200*, *210*
rotate, 148
rotate(), *18*, *56*, *60*, *74*, *85*, *99, 100*, *113*, *121, 122*, *129*, *137*, *180*, *189*, *203*
rtf_add.flextable, 150
rtf_add.flextable(), *58*, *60*, *141*

save_as_docx, 151
save_as_docx(), *58*, *83*, *91*, *108*, *118*, *125*, *141*, *145*, *147, 148*, *153–156*, *200*, *210*
save_as_html, 152
save_as_html(), *83*, *91*, *108*, *118*, *125*, *141*, *145*, *148*, *152*, *154–156*, *200*, *210*
save_as_image, 153
save_as_image(), *83*, *91*, *106*, *108*, *118*, *125*, *141*, *145*, *148*, *152, 153*, *155, 156*, *200*, *210*
save_as_pptx, 154
save_as_pptx(), *83*, *91*, *108*, *118*, *125*, *141*, *145*, *148*, *152–154*, *156*, *200*, *210*
save_as_rtf, 155
save_as_rtf(), *83*, *91*, *108*, *118*, *125*, *141*, *145*, *148*, *152–155*, *200*, *210*
save_as_typst, 157
separate_header, 158
separate_header(), *7, 8*, *10–13*, *15–17*, *79–81*, *139*, *168, 169*, *177–179*
set_caption, 159
set_caption(), *10*, *13*, *17*, *58*, *89*, *123*, *125*, *155*
set_flextable_defaults, 163
set_flextable_defaults(), *6*, *88, 89*, *103, 104*, *108*, *137*, *179*, *190–199*

set_footer_df *(set_header_footer_df)*, 168
set_formatter, 167
set_formatter(), *65–69*, *71–73*
set_header_df *(set_header_footer_df)*, 168
set_header_df(), *169*
set_header_footer_df, *8*, *10–13*, *15–17*, *79–81*, *139*, *158*, 168, *169*, *177–179*
set_header_footer_df(), *7*
set_header_labels, 169
set_header_labels(), *7*, *8*, *10–13*, *15–17*, *79–81*, *139*, *158*, *168*, *177–179*
set_table_properties, 170
set_table_properties(), *53*, *54*, *58*, *83*, *85*, *87*, *89*, *90*, *112*, *117*, *124*, *135–137*, *165*, *208*
shift_table, 173
split_columns, 176
split_columns(), *8*, *10–13*, *15–17*, *35*, *36*, *79–81*, *139*, *158*, *168*, *169*, *178*, *179*
split_rows, 177
split_rows(), *8*, *10–13*, *15–17*, *35*, *36*, *79–81*, *139*, *158*, *168*, *169*, *177–179*
split_to_pages, 178
split_to_pages(), *8*, *10–13*, *15–17*, *35*, *36*, *79–81*, *139*, *158*, *168*, *169*, *177*, *178*
strptime(), *66*, *67*, *165*
style, 179
style(), *18*, *56*, *60*, *74*, *85*, *89*, *99*, *100*, *113*, *121*, *122*, *129*, *137*, *149*, *189*, *203*
summarizor, 181
summarizor(), *34*, *39*, *91*, *92*, *185*
summary.tabulator *(tabulator)*, 184
surround, 182
surround(), *61–64*, *114–116*, *204–206*

tab_settings, 188
tab_settings(), *18*, *56*, *60*, *74*, *85*, *99*, *100*, *113*, *121*, *122*, *129*, *137*, *149*, *180*, *203*
tables::tabular(), *36*, *37*
tabulator, 184
tabulator(), *38*, *39*, *92–98*, *138*, *181*, *187*
tabulator_colnames, 186
tabulator_colnames(), *185*
theme_alafoli, 190
theme_alafoli(), *108*, *166*, *191–199*
theme_apa, 191
theme_apa(), *108*, *166*, *190*, *192–199*
theme_booktabs, 192
theme_booktabs(), *89*, *108*, *166*, *190*, *191*, *193–199*
theme_borderless, 193
theme_borderless(), *108*, *166*, *190–192*, *194–199*
theme_box, 194
theme_box(), *108*, *166*, *190–193*, *195–199*
theme_tron, 195
theme_tron(), *108*, *166*, *190–194*, *196–199*
theme_tron_legacy, 196
theme_tron_legacy(), *108*, *166*, *190–195*, *197–199*
theme_vader, 197
theme_vader(), *108*, *166*, *190–196*, *198*, *199*
theme_vanilla, 198
theme_vanilla(), *108*, *166*, *190–197*, *199*
theme_zebra, 199
theme_zebra(), *108*, *166*, *190–198*
to_html.flextable, 200
to_html.flextable(), *83*, *91*, *108*, *118*, *125*, *141*, *145*, *148*, *152–156*, *210*
toupper(), *126*

use_df_printer, 201
use_df_printer(), *82*, *202*, *203*
use_flextable_qmd, 201
use_flextable_qmd(), *47*, *49*, *124*, *125*, *201*, *203*
use_model_printer, 202
use_model_printer(), *201*, *202*

valign, 203
valign(), *18*, *56*, *60*, *74*, *85*, *99*, *100*, *113*, *121*, *122*, *129*, *137*, *149*, *179*, *180*, *189*
vline, 204
vline(), *61–64*, *103*, *114–116*, *182*, *183*, *205*, *206*
vline_left, 205
vline_left(), *61–64*, *114–116*, *182*, *183*, *204*, *206*
vline_right, 206
vline_right(), *61–64*, *114–116*, *183*, *204*, *205*
void, 207
void(), *20*, *47*, *77*, *102*, *126*, *144*

width, *149*, 208
width(), *54*, *83*, *85*, *87*, *90*, *112*, *117*, *135*, *136*, *140*, *172*
wrap_flextable, 209
wrap_flextable(), *83*, *91*, *106*, *108*, *118*, *125*, *141*, *145*, *148*, *152–156*, *200*