Package ‘flextable’

October 23, 2023

**Type**  Package

**Title**  Functions for Tabular Reporting

**Version**  0.9.4

**Description**  Use a grammar for creating and customizing pretty tables. The following formats are supported: 'HTML', 'PDF', 'RTF', 'Microsoft Word', 'Microsoft PowerPoint' and R 'Grid Graphics'. '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

**Imports**  stats, utils, grDevices, graphics, grid, rmarkdown, knitr, htmltools, rlang, ragg, officer (>= 0.6.2), gdtools (>= 0.3.3), xml2, data.table (>= 1.13.0), uuid (>= 0.1-4)

**RoxygenNote**  7.2.3

**Suggests**  testthat (>= 2.1.0), magick, equatags, commonmark, ggplot2, scales, doconv (>= 0.3.0), xtable, tables (>= 0.9.17), broom, broom.mixed, mgev, cluster, lme4, nle, bookdown (>= 0.34), pdf2tools, officedown, pkgdown (>= 2.0.0), webshot2, svglite

**Encoding**  UTF-8

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

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

**VignetteBuilder**  knitr

**NeedsCompilation**  no
Author  David Gohel [aut, cre],
ArData [cph],
Clementine Jager [ctb],
Panagiotis Skintzos [aut],
Quentin Fazilleau [ctb],
Maxim Nazarov [ctb] (rmarkdown for docx output),
Titouan Robert [ctb],
Michael Barrowman [ctb] (inline footnotes),
Atsushi Yasumoto [ctb] (support for bookdown cross reference),
Paul Julian [ctb] (support for gam objects),
Sean Browning [ctb] (work on footnote positioning system),
Rémi Thériault [ctb] (<https://orcid.org/0000-0003-4315-6788>, theme_apa),
Samuel Jobert [ctb] (work on pagination)

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

Repository  CRAN

Date/Publication  2023-10-22 22:00:03 UTC

R topics documented:

- flextable-package ........................................ 5
- add_body .................................................. 6
- add_body_row ............................................. 7
- add_footer ................................................ 9
- add_footer_lines ....................................... 10
- add_footer_row .......................................... 11
- add_header ............................................... 12
- add_header_lines ...................................... 13
- add_header_row ........................................ 15
- align ..................................................... 16
- append_chunks .......................................... 17
- as_b ..................................................... 18
- as_bracket ............................................... 19
- as_chunk ................................................ 20
- as_equation ............................................. 21
- as_flextable ............................................ 22
- as_flextable.data.frame ................................ 22
- as_flextable.gam ....................................... 24
- as_flextable.glm ....................................... 25
- as_flextable.grouped_data ............................. 25
- as_flextable.htest ..................................... 26
- as_flextable.kmeans ................................... 27
- as_flextable.lm ........................................ 28
- as_flextable.merMod .................................... 29
- as_flextable.pam ........................................ 30
- as_flextable.summarizor ............................... 31
- as_flextable.table ..................................... 32
R topics documented:

- as_flextable.tabular .................................................. 32
- as_flextable.tabulator ................................................ 34
- as_flextable.xtable ................................................... 37
- as_grouped_data ....................................................... 39
- as_highlight ............................................................ 40
- as.i ................................................................. 40
- as_image ............................................................... 41
- as_paragraph ............................................................ 42
- as_sub ................................................................. 43
- as_sup ................................................................. 44
- as_word_field ......................................................... 45
- autofit ................................................................. 47
- before ................................................................. 48
- bg ................................................................. 49
- body_add_flextable .................................................... 50
- body_replace_flextable_at_bkm ..................................... 51
- bold ................................................................. 52
- border_inner ........................................................... 53
- border_inner_h ........................................................ 53
- border_inner_v ........................................................ 54
- border_outer .......................................................... 55
- border_remove ......................................................... 56
- colformat_char ........................................................ 56
- colformat_date ........................................................ 57
- colformat_datetime .................................................. 58
- colformat_double ..................................................... 59
- colformat_image ..................................................... 60
- colformat_int ........................................................ 61
- colformat_lgl ........................................................ 62
- colformat_num ........................................................ 63
- color ................................................................. 65
- colorize ............................................................... 66
- compose ............................................................... 67
- continuous_summary .................................................. 68
- delete_columns ....................................................... 69
- delete_part ........................................................... 70
- delete_rows ........................................................... 70
- df_printer ............................................................ 71
- dim.flextable ......................................................... 72
- dim.flextableGrob .................................................... 73
- dim_pretty ........................................................... 73
- empty_blanks .......................................................... 74
- fit_to_width .......................................................... 75
- fix_border_issues ..................................................... 76
- flextable ............................................................ 77
- flextable_dim ........................................................ 79
- flextable_to_rmd ..................................................... 79
- fmt_2stats ............................................................ 80
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmt_avg_dev</td>
<td>82</td>
</tr>
<tr>
<td>fmt_dbl</td>
<td>83</td>
</tr>
<tr>
<td>fmt_header_n</td>
<td>84</td>
</tr>
<tr>
<td>fmt_int</td>
<td>84</td>
</tr>
<tr>
<td>fmt_n_percent</td>
<td>85</td>
</tr>
<tr>
<td>fmt_pct</td>
<td>86</td>
</tr>
<tr>
<td>font</td>
<td>87</td>
</tr>
<tr>
<td>footnote</td>
<td>88</td>
</tr>
<tr>
<td>fp_border_default</td>
<td>91</td>
</tr>
<tr>
<td>fp_text_default</td>
<td>92</td>
</tr>
<tr>
<td>gen_grob</td>
<td>93</td>
</tr>
<tr>
<td>get_flextable_defaults</td>
<td>96</td>
</tr>
<tr>
<td>gg_chunk</td>
<td>97</td>
</tr>
<tr>
<td>grid_chunk</td>
<td>98</td>
</tr>
<tr>
<td>height</td>
<td>99</td>
</tr>
<tr>
<td>highlight</td>
<td>100</td>
</tr>
<tr>
<td>hline</td>
<td>101</td>
</tr>
<tr>
<td>hline_bottom</td>
<td>102</td>
</tr>
<tr>
<td>hline_top</td>
<td>103</td>
</tr>
<tr>
<td>hrule</td>
<td>104</td>
</tr>
<tr>
<td>htmltools_value</td>
<td>105</td>
</tr>
<tr>
<td>hyperlink_text</td>
<td>106</td>
</tr>
<tr>
<td>italic</td>
<td>107</td>
</tr>
<tr>
<td>keep_with_next</td>
<td>108</td>
</tr>
<tr>
<td>knit_print.flextable</td>
<td>109</td>
</tr>
<tr>
<td>labelizor</td>
<td>112</td>
</tr>
<tr>
<td>linerange</td>
<td>113</td>
</tr>
<tr>
<td>line_spacing</td>
<td>115</td>
</tr>
<tr>
<td>lollipop</td>
<td>115</td>
</tr>
<tr>
<td>merge_at</td>
<td>117</td>
</tr>
<tr>
<td>merge_h</td>
<td>118</td>
</tr>
<tr>
<td>merge_h_range</td>
<td>118</td>
</tr>
<tr>
<td>merge_none</td>
<td>119</td>
</tr>
<tr>
<td>merge_v</td>
<td>120</td>
</tr>
<tr>
<td>minibar</td>
<td>121</td>
</tr>
<tr>
<td>ncol_keys</td>
<td>122</td>
</tr>
<tr>
<td>nrow_part</td>
<td>123</td>
</tr>
<tr>
<td>padding</td>
<td>124</td>
</tr>
<tr>
<td>paginate</td>
<td>125</td>
</tr>
<tr>
<td>ph_with.flextable</td>
<td>127</td>
</tr>
<tr>
<td>plot.flextable</td>
<td>128</td>
</tr>
<tr>
<td>plot.flextableGrob</td>
<td>129</td>
</tr>
<tr>
<td>plot_chunk</td>
<td>129</td>
</tr>
<tr>
<td>prepend_chunks</td>
<td>131</td>
</tr>
<tr>
<td>print.flextable</td>
<td>131</td>
</tr>
<tr>
<td>proc_freq</td>
<td>132</td>
</tr>
<tr>
<td>rotate</td>
<td>133</td>
</tr>
</tbody>
</table>
rtf_add.flextable .................................................. 135
save_as_docx ....................................................... 136
save_as_html ....................................................... 137
save_as_image .................................................... 138
save_as_pptx ...................................................... 139
save_as_rtf ....................................................... 140
separate_header .................................................. 141
set_caption ....................................................... 142
set_flextable_defaults .......................................... 147
set_formatter ..................................................... 150
set_header_footer_df ............................................. 151
set_header_labels ............................................... 153
set_table_properties ............................................. 154
shift_table ....................................................... 156
style ............................................................. 159
summarizor ....................................................... 161
surround .......................................................... 162
tabulator ........................................................... 163
tabulator_colnames ............................................... 166
theme_alafoli .................................................... 168
theme_apa .......................................................... 169
theme_booktabs ................................................... 170
theme_box ........................................................ 171
theme_tron ....................................................... 172
theme_tron_legacy ............................................... 173
theme_vader ...................................................... 174
theme_vanilla ..................................................... 175
theme_zebra ...................................................... 176
to_html.flextable ................................................ 177
use_df_printer ..................................................... 178
use_model_printer ................................................ 179
valign ............................................................ 179
vline ............................................................... 180
vline_left ........................................................ 181
vline_right ....................................................... 182
void ............................................................... 182
width .............................................................. 183

Index 184

---

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")`. Function `mk_par()` lets customise text of cells.

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.

See Also


---

add_body Add column values as new lines in body

Description

The function adds a list of values to be inserted as new rows in the body. The values are inserted in existing columns of the input data of the flextable. Rows can be inserted at the top or the bottom of the body.

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
- `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).
add_body_row

See Also

flextable()

Other functions for row and column operations in a flextable: add_body_row(), add_footer_lines(),
add_footer_row(), add_footer(), add_header_row(), add_header(), delete_columns(), delete_part(),
delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

Examples

```r
def <- flextable(head(iris),
col_keys = c(
  "Sepal.Width", "Petal.Width"
)
)
def <- add_body(
x = def, Sepal.Length = 1:5,
  Sepal.Width = 1:5 * 2, Petal.Length = 1:5 * 3,
  Petal.Width = 1:5 + 10, Species = "Blah", top = FALSE
)
def <- theme_booktabs(def)
def
```

---

<table>
<thead>
<tr>
<th>add_body_row</th>
<th>Add body labels</th>
</tr>
</thead>
</table>

Description

Add a row of new columns labels in body part. Labels can be spanned along multiple columns, as
merged cells.

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

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

Arguments

- `x` a 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 for-
matting. 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 functions for row and column operations in a flextable: add_body(), add_footer_lines(),
add_footer_row(), add_footer(), add_header_row(), add_header(), delete_columns(), delete_part(),
delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

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
)
Description

The function adds a list of values to be inserted as new rows in the footer. The values are inserted in existing columns of the input data of the flextable. Rows can be inserted at the top or the bottom of the footer.

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

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 functions for row and column operations in a flextable: add_body_row(), add_body(), add_footer_lines(), add_footer_row(), add_header_row(), add_header(), delete_columns(), delete_part(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

Examples

ew_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 labels as new rows in the footer

**Description**

Add labels as new rows in the footer, where all columns are merged.

This is a sugar function to be used when you need to add labels in the footer, a footnote for example.

**Usage**

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

**Arguments**

- `x` a flextable object
- `values` a character vector or a call to `as_paragraph()` to get formatted 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 functions for row and column operations in a flextable: `add_body_row()`, `add_body()`, `add_footer_row()`, `add_footer()`, `add_header_row()`, `add_header()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `separate_header()`, `set_header_footers_df`, `set_header_labels()`

**Examples**

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

Add a row of new columns labels in footer part. Labels can be spanned along multiple columns, as merged cells.

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

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

### Arguments

- **x**: a `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 functions for row and column operations in a `flextable`: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer()`, `add_header_row()`, `add_header()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `separate_header()` `set_header_footer_df`, `set_header_labels()`

### Examples

```r
library(flextable)

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

pars <- as_paragraph(
```
add_header

Add column values as new lines in header

Description

The function adds a list of values to be inserted as new rows in the header. The values are inserted in existing columns of the input data of the flextable. Rows can be inserted at the top or the bottom of the header.

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

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 functions for row and column operations in a flextable: add_body_row(), add_body(),
add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), delete_columns(),
delete_part(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

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 labels as new rows in the header
add_header_lines

Description

Add labels as new rows in the header, where all columns are merged.

This is a sugar function to be used when you need to add labels in the header, most of the time it will be used to adding titles on the top rows of the flextable.

Usage

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

Arguments

x
a flextable object

values
a character vector or a call to as_paragraph() to get formatted 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.

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

Description

Add a row of new columns labels in header part. Labels can be spanned along multiple columns, as merged cells.

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

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 functions for row and column operations in a flextable: add_body_row(), add_body(), add_footer_lines(), add_footer_row(), add_footer(), add_header(), delete_columns(), delete_part(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

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)
)

```r
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 text alignment of selected rows and columns of a flextable.

**Usage**

```r
align(
  x,
  i = NULL,
  j = NULL,
  align = c("left", "center", "right", "justify"),
  part = "body"
)
```

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

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

**Arguments**

- `x` - a flextable object
- `i` - rows selection
- `j` - columns selection
- `align` - text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.
append_chunks

part partname of the table (one of 'all', 'body', 'header', 'footer')
header should the header be aligned with the body
footer should the footer be aligned with the body

See Also
Other sugar functions for table style: bg(), bold(), color(), empty_blanks(), fontsize(), font(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), valign()

Examples

ft <- flextable(head(mtcars)[, 3:6])
ft <- align(ft, align = "right", part = "all")
ft <- theme_tron_legacy(ft)
ft
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
... chunks to be appended, see as_chunk(), gg_chunk() and other chunk elements for paragraph.
i rows selection
j column selection
part partname of the table (one of 'body', 'header', 'footer')

See Also
as_chunk(), as_sup(), as_sub(), colorize()
Other functions for mixed content paragraphs: as_paragraph(), compose(), prepend_chunks()
Examples

```r
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
```

<table>
<thead>
<tr>
<th>as_b</th>
<th><strong>Bold chunk</strong></th>
</tr>
</thead>
</table>

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 `prependChunks()`.

Usage

```r
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_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

Examples

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

ft <- compose(ft,
  j = "dummy",
  value = as_paragraph(
```

```r
```
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

```r
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_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

Examples

```r
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

Chunk of text wrapper

Description

The function lets add formatted 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_bracket()`, `as_b()`, `as_equation()`, `as_highlight()`, `as_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

Examples

```r
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

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; also `equatags::mathjax_install()` must be executed only once to install necessary dependencies.

Usage

```r
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_bracket()`, `as_b()`, `as_chunk()`, `as_highlight()`, `as_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

Examples

```r
library(flextable)
if (require("equatags") && mathjax_available()) {
  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))
  )
}
```
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.

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 as_flextable methods: as_flextable.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.h test(), 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()
as_flextable.data.frame

Usage

```r
## 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.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()`, `as_flextable()`

Examples

```r
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.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(), as_flextable()

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.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.summarizer(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), as_flextable()

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

```r
## 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.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()`, `as_flextable()`

Examples

```r
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

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

Arguments

- `x`: htest object
- `...`: unused argument

See Also

Other as_flextable methods: `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.summarizer()`, `as_flextable.table()`, `as_flextable.tabular()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `as_flextable()`

Examples

```r
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**

```r
## 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
as_flextable.lm

Transform a 'lm' object into a flextable

Usage

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

Arguments

x  
lm model

...  
unused argument

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.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.summarizor(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable() as_flextable()

Examples

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

Examples

if (require("broom")) {
  lmod <- lm(rating ~ complaints + privileges +
              learning + raises + critical, data = attitude)
  ft <- as_flextable(lmod)
  ft
}
as_flextable.merMod  Transform a mixed model 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

```r
## 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.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.summarizer(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), as_flextable()
Examples

```r
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

```r
## 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.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()`, `as_flextable()`

Examples

```r
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.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.h.test(), as_flextable.kmeans(), as_flextable.lm(), as_flextable.merMod(), as_flextable.pam(), as_flextable.table(), as_flextable.tabular(), as_flextable.tabulator(), as_flextable.xtable(), as_flextable()

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.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 formatted 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:

Usage

```r
## 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.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.summarizer()`, `as_flextable.tabular()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `as_flextable()`

Examples

```r
tab <- with(warpbreaks, table(wool, tension))
tf <- as_flextable(tab)
tf
```
-- "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

```r
## S3 method for class 'tabular'
as_flextabular(
  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 `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.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.tabulator()`, `as_flextable.xtable()`, `as_flextable()`

Examples

```r
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)
}

Transform a 'tabulator' object into a flextable

Description

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

Usage

```r
## 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 separate the groups with an horizontal line.
- `big_border, small_border` big and small border properties defined by a call to `fp_border_default()` or `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.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(), as_flextable()

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"
  )
}
```r
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 = "rownames",
  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.data.frame(), as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.hst(), 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()

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|c|r|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","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

Add row separators to grouped data

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_bracket()`, `as_b()`, `as_chunk()`, `as_equation()`, `as_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

**Examples**

```r
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

- `color`  
  color to use as text highlighting color as character vector.

**See Also**

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

**Examples**

```r
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_image

Usage

as_i(x)

Arguments

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

See Also

Other chunk elements for paragraph: as_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), lollipop(), 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 src, width = NULL, height = NULL, unit = "in", guess_size = TRUE, ...)

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".

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, ...)

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 package 'magick' is installed, this option can be used (set it to TRUE and don’t provide values for parameters width and height). When the flextable will be printed, the images will be read and width and height will be guessed. This should be avoided if possible as it can be an extensive task when several images.

Note
This chunk option requires package officedown in a R Markdown context with Word output format.
PowerPoint cannot mix images and text in a paragraph, images are removed when outputting to PowerPoint format.

See Also
compose(), as_paragraph()

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

Examples

```r
img.file <- file.path(R.home("doc"),
  "html", "logo.jpg"
)
if (require("magick")) {
  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**  
Concatenate chunks in a flextable
as_sub

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

```r
as_chunk(...) as_sub
```

Arguments

- `...` chunk elements that are defining paragraph. 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 paragraph. If specified argument `...` is unused.

See Also

- `as_chunk()`, `minibar()`, `as_image()`, `hyperlink_text()`

Other functions for mixed content paragraphs: `append_chunks()`, `compose()`, `prepend_chunks()`

Examples

```r
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
```
Usage

    as_sub(x)

Arguments

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

See Also

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

Examples

```r
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_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), lollipop(), minibar(), plot_chunk()

Examples

```r
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' computed field

Description

This function is used to insert 'Word' computed field 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().

This has only effect on 'Word' output. If you want to condition its execution only for Word output, you can use it in the post processing step (see set_flextable_defaults(post_process_docx = ...))

Do not forget to update the computed field in Word. Fields are defined but are not computed, this computing is an operation that has to be made by 'Microsoft Word' (select all text and hit F9 when on mac os).

Usage

```r
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".
as_word_field

See Also

Other chunk elements for paragraph: as_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_sup(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), lollipop(), 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**

*Adjusts cell widths and heights*

**Description**

compute and apply optimized widths and heights (minimum estimated widths and heights for each table columns and rows in inches returned by function `dim_pretty()`).

This function is to be used when the table widths and heights should be adjusted to fit the size of the content.

The function does not let you adjust a content that is too wide in a paginated document. It simply calculates the width of the columns so that each content has the minimum width necessary to display the content on one line.

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**

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

**Arguments**

- **x**
  - *flextable object*

- **add_w**
  - extra width to add in inches

- **add_h**
  - extra height to add in inches

- **part**
  - partname of the table (one of 'all', 'body', 'header' or 'footer')

- **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 flextable dimensions: `dim.flextable()`, `dim_pretty()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

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

---

### before

**Is an element before a match with entries**

#### Description

return a logical vector of the same length as `x`, indicating if elements are located before a set of entries to match or not.

#### Usage

```r
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

```r
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
```
```r
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
```

---

**Description**

Change 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` has to be used to define what are the columns to be used for the color definition and the argument `j` has to be used to define where to apply the colors and only accept values from `colkeys`.

**Usage**

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

**Arguments**

- `x`: a flextable object
- `i`: rows selection
- `j`: columns selection
- `bg`: color to use as background color. If a function, function need to return a character vector of colors.
- `part`: partname of the table (one of `all`, `body`, `header`, `footer`)
- `source`: if `bg` is a function, source is specifying the dataset column to be used as argument to `bg`. This is only useful if `j` is colored with values contained in other columns.

**Note**

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

Add flextab into a Word document

add a flextab into a Word document.

Usage

body_add_flextab(
  x,
  value,
  align = NULL,
  pos = "after",
  split = NULL,
  topcaption = TRUE,
  keepnext = NULL
)
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'.

---

**bold**

Set bold font

---

Description

description change 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
- **i**: rows selection
- **j**: columns selection
- **bold**: boolean value
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

Other sugar functions for table style: align(), bg(), color(), empty_blanks(), fontsize(), font(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), valign()

Examples

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

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

**Usage**

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

**Arguments**

- `x`: a flextable object
- `border`: border properties defined by a call to `fp_border()`
- `part`: partname of the table (one of `"all"`, `"body"`, `"header"`, `"footer"`)

**See Also**

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

**Examples**

```r
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 borders

**Description**

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

**Usage**

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

x  a flextable object
border  border properties defined by a call to \texttt{fp\_border()}
part  partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

Other borders management: \texttt{border\_inner\_v()}, \texttt{border\_inner()}, \texttt{border\_outer()}, \texttt{border\_remove()}, \texttt{hline\_bottom()}, \texttt{hline\_top()}, \texttt{hline()}, \texttt{surround()}, \texttt{vline\_left()}, \texttt{vline\_right()}, \texttt{vline()}

Examples

\begin{verbatim}
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
\end{verbatim}
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

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
  border           border properties defined by a call to fp_border()
  part             partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

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

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

Description
The function is deleting all borders of the flextable object.

Usage
border_remove(x)

Arguments
x a flextable object

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

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

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
- `i`  rows selection
- `j`  columns selection.
- `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_datetime()`, `colformat_date()`, `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
- `i`: rows selection
- `j`: columns selection.
- `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_datetime()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

```r
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

```r
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 = "" 
)
```
Format numeric cells

Description

Format numeric cells in a flextable.

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
i rows selection
j columns selection.
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_datetime(), colformat_date(), 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 = ""
)
Arguments

- `x` a flextable object
- `i` rows selection
- `j` columns selection.
- `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

See Also

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

Examples

```r
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
```

Description

Format integer cells in a flextable.

Usage

```r
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,
```

```r```
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 = ""
)
colformat_num

Arguments

- `x`: a `flextable` object
- `i`: rows selection
- `j`: columns selection.
- `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_datetime()`, `colformat_date()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_num()`, `set_formatter()`

Examples

```r
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)
```

## Description

Format numeric cells in a `flextable`.

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

```r
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
i  rows selection
j  columns selection.
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_datetime(), colformat_date(), 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 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) as a common use case. The argument source has to be used to define what are the columns to be used for the color definition and the argument j has to be used to define where to apply the colors and only accept values from colkeys.

Usage

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

Arguments

x a flextable object
i rows selection
j columns selection
color color to use as font color. If a function, function need to return a character vector of colors.
part partname of the table (one of 'all', 'body', 'header', 'footer')
source if color is a function, source is specifying the dataset column to be used as argument to color. This is only useful if j is colored with values contained in other columns.

See Also

Other sugar functions for table style: align(), bg(), bold(), empty_blanks(), fontsize(), font(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), 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)

ft_2 <- flextable(x)
ft_2 <- color(ft_2, j = x$colname, color = scale)
ft_2 <- set_formatter_type(ft_2)
ft_2

---

**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_bracket()`, `as_b()`, `as_chunk()`, `as_equation()`, `as_highlight()`,
`as_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`,
`linerange()`, `lollipop()`, `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"))
)```
**Compose**

Define displayed values and mixed content

**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()`, formatted 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**

```r
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
- **i**: rows selection
- **j**: column selection
- **value**: a call to function `as_paragraph()`. 
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')
- **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 jth column.

**See Also**

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

Other functions for mixed content paragraphs: `append_chunks()`, `as_paragraph()`, `prepend_chunks()`

**Examples**

```r
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
  ), use_dot = TRUE
)
ft_2 <- theme_vader(ft_2)
ft_2 <- autofit(ft_2)
ft_2

---

`continuous_summary`  **Continuous columns summary**

**Description**

create a data.frame summary for continuous variables

**Usage**

```r
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
j
  columns selection

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 functions for row and column operations in a flextable: add_body_row(), add_body(), add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), add_header(), delete_part(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

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
part   partname of the table to delete (one of 'body', 'header' or 'footer').

See Also

Other functions for row and column operations in a flextable: add_body_row(), add_body(),
add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), add_header(),
delete_columns(), delete_rows(), separate_header(), set_header_footer_df, set_header_labels()

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
i     rows selection
part   partname of the table (one of 'all', 'body', 'header', 'footer')
**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 functions for row and column operations in a flextable: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_row()`, `add_header()`, `delete_columns()`, `delete_part()`, `separate_header()`, `set_header_footer_df`, `set_header_labels()`

**Examples**

```r
def_printer(head(iris))
def_printer(delete_rows(ft, i = 1:5, part = "body"))
```

---

`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:

```yaml
---
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:

```r
use_df_printer()
```

**Usage**

`df_printer(dat, ...)`

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>the data.frame</td>
</tr>
<tr>
<td>...</td>
<td>unused argument</td>
</tr>
</tbody>
</table>
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 flextable print function: `as_raster()`, `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()`.

Examples

df_printer(head(mtcars))

---

**dim.flexttable**

Get widths and heights of flextable

Description

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

Usage

```r
## S3 method for class 'flexttable'
dim(x)
```

Arguments

- **x**: flextable object

See Also

Other flextable dimensions: `autofit()`, `dim.pretty()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`
### 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

```r
## 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

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

### dim_pretty

**Calculate pretty dimensions**

#### Description

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

#### Usage

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

- **x**: a flextable object
- **part**: partname of the table (one of 'all', 'body', 'header' or 'footer')
- **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 flextable dimensions: `autofit()`, `dim.flextable()`, `fit_to_width()`, `flextab_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

Examples

```r
ftab <- flextab(head(mtcars))
dim_pretty(ftab)
```

empty_blanks

Make blank columns as transparent

Description

Blank columns are set as transparent. This is a shortcut function that will delete top and bottom borders, change background color to transparent, display empty content and set blank columns' width.

Usage

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

Arguments

- **x**: a flextable object
- **width**: width of blank columns (.1 inch by default).
- **unit**: unit for width, one of "in", "cm", "mm".
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

Other sugar functions for table style: `align()`, `bg()`, `bold()`, `color()`, `fontsize()`, `font()`, `highlight()`, `italic()`, `keep_with_next()`, `line_spacing()`, `padding()`, `rotate()`, `valign()`
Examples

typology <- data.frame(
  col_keys = c(
    "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",
  "break2", "Petal.Length", "Petal.Width"
))
tab <- set_header_df(ftab, mapping = typology, key = "col_keys")
tab <- merge_h(tab, part = "header")
tab <- theme_vanilla(tab)
tab <- empty_blanks(tab)
tab <- width(tab, j = c(2, 5), width = .1)
tab

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

Description

decrease font size for each cell incrementally until it fits a given max_width.

Usage

Arguments

x 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 flextable dimensions: autofit(), dim.flextab(), dim.pretty(), flextable_dim(), height(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()
Examples

```r
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
```

---

**fix_border_issues**  
*Fix border issues when cell are merged*

**Description**

When cells are merged, the rendered borders will be those of the first cell. If a column is made of three merged cells, the bottom border that will be seen will be the bottom border of the first cell in the column. From a user point of view, this is wrong, the bottom should be the one defined for cell 3. This function modify the border values to avoid that effect.

**Usage**

```r
fix_border_issues(x, part = "all")
```

**Arguments**

- `x`  
  flextable object

- `part`  
  partname of the table (one of ‘all’, ‘body’, ‘header’, ‘footer’)

**Examples**

```r
library(officer)
dat <- data.frame(a = 1:5, b = 6:10)
ft <- flextable(dat)
ft <- theme_box(ft)
ft <- merge_at(ft, i = 4:5, j = 1, part = "body")
ft <- hline(ft,
            i = 5, part = "body",
            border = fp_border(color = "red", width = 5))
print(ft)
ft <- fix_border_issues(ft)
print(ft)
```
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

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

```r
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.

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 re-set 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"

**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 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()
```

**Examples**

```r
ft <- flextable(head(mtcars))
ft
```
**flextable_dim**  
*Get width and height of a flextable object*

**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 width, height and aspect_ratio.

**Usage**

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

**Arguments**

- `x` a flextable object
- `unit` unit for returned values, one of "in", "cm", "mm".

**See Also**

Other flextable dimensions: `autofit()`, `dim.flextable()`, `dim.pretty()`, `fit_to_width()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

**Examples**

```r
ftab <- flextable(head(iris))
flextile_dim(ftab)
ftab <- autofit(ftab)
flextile_dim(ftab)
```

---

**flextable_to_rmd**  
*flextable raw code*

**Description**

Print openxml, latex or html code of a flextable. 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 `knitr_print.flextable` for more details.

**Usage**

```r
flextable_to_rmd(x, ...)
```
Arguments

x a flextable object

... unused argument

See Also

Other flextable print function: as_raster(), df_printer(), 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()

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 content for data generated with summarizor()

Description

This function was written to allow easy demonstrations of flextable’s ability to produce table sum-
maries (with summarizor()). It assumes that we have either a quantitative variable, in which case
we will display the mean and the standard deviation, or a qualitative variable, in which case we will
display the count and the percentage corresponding to each modality.

Usage

fmt_2stats(
  stat,
  num1,
  num2,
  cts,
fmt_2stats

pcts,
num1_mask = "%.01f",
num2_mask = "%(01f)",
cts_mask = "%.0f",
pcts_mask = "%(02f%%)"
)

fmt_summarizor(
  stat,
  num1,
  num2,
  cts,
  pcts,
  num1_mask = "%.01f",
  num2_mask = "%(01f)",
  cts_mask = "%.0f",
  pcts_mask = "%(02f%%)"
)

Arguments

stat  a character column containing the name of statistics
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
num1_mask  format associated with num1, a format string used by sprintf().
num2_mask  format associated with num2, a format string used by sprintf().
cts_mask  format associated with cts, a format string used by sprintf().
pcts_mask  format associated with pcts, a format string used by sprintf().

See Also

summarizor(), tabulator(), mk_par()

Other text formatter functions: fmt_avg_dev(), fmt_dbl(), fmt_header_n(), fmt_int(), fmt_n_percent(), fmt_pct()

Examples

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

  tab_1 <- tabulator(
    x = z,
    rows = c("variable", "stat"),
    columns = "Species",
    )
fmt_avg_dev

Format content for mean and sd

Description

The function formats 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 text formatter functions: fmt_2stats(), fmt_dbl(), fmt_header_n(), fmt_int(), fmt_n_percent(), fmt_pct()
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 numerical data as percentages

Description

The function formats numeric vectors as percentages.

Usage

fmt_dbl(x)

Arguments

x numeric values

See Also

tabulator(), mk_par()

Other text formatter functions: fmt_2stats(), fmt_avg_dev(), fmt_header_n(), fmt_int(), fmt_n_percent(), fmt_pct()

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 data for headers

Description

The function formats counts as \( N=XX \). This helper function is used to add counts in columns 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 text formatter functions: fmt_2stats(), fmt_avg_dev(), fmt_dbl(), fmt_int(), fmt_n_percent(), fmt_pct()

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 numerical data as integer

Description

The function formats numeric vectors as integer.

Usage

fmt_int(x)
**fmt_n_percent**

**Arguments**

- `x` numeric values

**See Also**

`tabulator()`, `mk_par()`

Other text formatter functions: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_n_percent()`, `fmt_pct()`

**Examples**

```r
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 content for count data*

**Description**

The function formats counts and percentages as n (xx.x%). If percentages are missing, they are not printed.

**Usage**

```r
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 text formatter functions: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_pct()`
Examples

```r
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 numerical data as percentages

Description

The function formats numeric vectors as percentages.

Usage

```r
fmt_pct(x)
```

Arguments

- `x` numeric values

See Also

`tabulator()`, `mk_par()`

Other text formatter functions: `fmt_2stats()`, `fmt_avg_dev()`, `fmt_dbl()`, `fmt_header_n()`, `fmt_int()`, `fmt_n_percent()`
Examples

```r
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
```

---

## font

### Description

Change 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 in flextable with function `gdtools::register_gfont()`. When output is HTML, the font will be automatically added in the HTML document.

### Usage

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

### Arguments

- **x** a flextable object
- **i** rows selection
- **j** columns selection
- **fontname** single character value, the font family name. With Word and PowerPoint output, the value specifies the font to be used to format characters in the Unicode range (U+0000-U+007F).
part

<table>
<thead>
<tr>
<th>part</th>
<th>partname of the table (one of 'all', 'body', 'header', 'footer')</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.family</td>
<td>Optional font to be used to format characters in a complex script Unicode range. For example, Arabic text might be displayed using the &quot;Arial Unicode MS&quot; font. Used only with Word and PowerPoint outputs. Its default value is the value of fontname.</td>
</tr>
<tr>
<td>hansi.family</td>
<td>optional. Specifies the font to be used to format characters in a Unicode range which does not fall into one of the other categories. Used only with Word and PowerPoint outputs. Its default value is the value of fontname.</td>
</tr>
<tr>
<td>eastasia.family</td>
<td>optional font to be used to format characters in an East Asian Unicode range. For example, Japanese text might be displayed using the &quot;MS Mincho&quot; font. Used only with Word and PowerPoint outputs. Its default value is the value of fontname.</td>
</tr>
</tbody>
</table>

See Also

Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), fontsize(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), 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

<table>
<thead>
<tr>
<th>Set font size</th>
</tr>
</thead>
</table>

Description

change font size of selected rows and columns of a flextable.

Usage

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

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>a flextable object</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>rows selection</td>
</tr>
<tr>
<td>j</td>
<td>columns selection</td>
</tr>
<tr>
<td>size</td>
<td>integer value (points)</td>
</tr>
<tr>
<td>part</td>
<td>partname of the table (one of 'all', 'body', 'header', 'footer')</td>
</tr>
</tbody>
</table>
footnote

See Also

Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), font(), highlight(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), 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 = "; "
)
```

Arguments

- **x**: a flextable object
- **i, j**: cellwise rows and columns selection
- **value**: a call to function as_paragraph().
- **ref_symbols**: character value, symbols to append that will be used as references to notes.
- **part**: partname of the table (one of 'body', 'header', 'footer')
- **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.
Examples

```r
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**

Border formatting properties

---

**Description**

Create a `fp_border()` object that uses default values defined in flextable defaults formatting properties, i.e. default border color (see `set_flextable_defaults()`).

**Usage**

```r
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: "none" or "solid" or "dotted" or "dashed"
- `width`: border width - an integer value: 0>= value

**See Also**

`hline()`, `vline()`

Other functions for defining formatting properties: `fp_text_default()`

**Examples**

```r
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 | Text formatting properties |

**Description**

Create a `fp_text()` object that uses default values defined in the flextable it applies to.

`fp_text_default()` is a handy function that will allow you to specify certain formatting values to be applied to a piece of text, the formatting values that are not specified will simply be the existing formatting values.

For example, if you set the text in the cell to red previously, using the code `fp_text_default(bold = TRUE)`, the formatting will be 'bold' but it will also be 'red'.

On the other hand, the `fp_text()` function forces you to specify all the parameters, so we strongly recommend working with `fp_text_default()` which was created to replace the use of the former.

See also `set_flextable_defaults()` to modify flextable defaults formatting properties.

**Usage**

```r
fp_text_default(
  color = flextable_global$defaults$font.color,
  font.size = flextable_global$defaults$font.size,
  bold = FALSE,
  italic = FALSE,
  underlined = 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
- **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 functions for defining formatting properties: `fp_border_default()`

Examples

```r
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()
```

**Convert a flextable to a grid grob object**
Description

It uses Grid Graphics (package grid) to Convert a flextable into a grob object with scaling and text wrapping capabilities.

This method can be used to insert a flextable inside a ggplot2 plot, it can also be used with package 'patchwork' or 'cowplot' to combine ggplots and flextables into the same graphic.

User can vary the size of the elements according to the size of the graphic window. The text behavior is controllable, user can decide to make the paragraphs (texts and images) distribute themselves correctly in the available space of the cell. It is possible to define resizing options, for example by using only the width, or by distributing the content so that it occupies the whole graphic space. It is also possible to freeze or not the size of the columns.

It is not recommended to use this function for large tables because the calculations can be long.

Limitations: equations (see as_equation()) and hyperlinks (see hyperlink_ftext()) will not be displayed.

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

... 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 \texttt{min}, except that the content sizes are scaled fully, they will increase or decrease, according to the size of the drawing surface.
- **fixed** or \texttt{FALSE}: The content sizes will not be scaled.

\textbf{wrapping} \hfill Determines the soft wrapping (line breaking) method for the table cell contents. One of \texttt{TRUE}, \texttt{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.

\textbf{autowidths} \hfill If \texttt{TRUE} (default) the column widths are adjusted in order to fit the contents of the cells (taking into account the \texttt{wrapping} setting).

\textbf{just} \hfill Justification of viewport layout, same as \texttt{just} argument in \texttt{grid::grid.layout()}. When set to \texttt{NULL} (default), it is determined according to the \texttt{fit} argument.

\textbf{Value}

- a grob (gTree) object made with package \texttt{grid}

\textbf{size}

The size of the flextable can be known by using the method \texttt{dim} on the grob.

\textbf{See Also}

Other \texttt{flextable} print function: \texttt{as_raster()}, \texttt{df_printer()}, \texttt{flextable_to_rmd()}, \texttt{htmltools_value()}, \texttt{knit_print.flextable()}, \texttt{plot.flextable()}, \texttt{print.flextable()}, \texttt{save_as_docx()}, \texttt{save_as_html()}, \texttt{save_as_image()}, \texttt{save_as_pptx()}, \texttt{save_as_rtf()}, \texttt{to_html.flextable()}

\textbf{Examples}

```r
ft <- flextable(head(mtcars))
ft <- autofit(ft)
gr <- gen_grob(ft)

used_family <- get_flextable_defaults()$font.family
if (gdtools::font_family_exists(used_family) && require("ragg")) {
  png_f <- tempfile(fileext = ".png")
  # get the size
  dms <- dim(gr)
} else {
  # use better default
  col_sizes <- ft$autowidth()
  gr <- gen_grob(ft, col_sizes = col_sizes)
}
```
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 functions related to themes: set_flextable_defaults(), theme_alafoli(), theme_apa(), theme_booktabs(), theme_box(), theme_tron_legacy(), theme_tron(), theme_vader(), theme_vanilla(), theme_zebra()

Examples

get_flextable_defaults()
**Description**

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

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

**Usage**

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

**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

**Note**

This chunk option requires package officedown in a R Markdown context with Word output format. PowerPoint cannot mix images and text in a paragraph, images are removed when outputting to PowerPoint format.

**See Also**

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

**Examples**

```r
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"
  }
```
grid_chunk

'Grid Graphics' chunk wrapper

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)

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
Note

This chunk option requires package officedown in a R Markdown context with Word output format. 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_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), hyperlink_text(), linerange(), lollipop(), minibar(), plot_chunk()

Examples

```r
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

```r
height(x, i = NULL, height, part = "body", unit = "in")
height_all(x, height, part = "all", unit = "in")
```
Arguments

- **x**: flextable object
- **i**: rows selection
- **height**: height in inches
- **part**: partname of the table
- **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 flextable dimensions: `autofit()`, `dim.flextable()`, `dim_pretty()`, `fit_to_width()`, `flextable_dim()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

**Examples**

```r
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

<table>
<thead>
<tr>
<th>highlight</th>
<th>Text highlight color</th>
</tr>
</thead>
</table>

**Description**

Change 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`) as a common use case. The argument `source` has to be used to define what are the columns to be used for the color definition and the argument `j` has to be used to define where to apply the colors and only accept values from `colkeys`. 
Usage

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

Arguments

x
  a flextable object

i
  rows selection

j
  columns selection

color
  color to use as text highlighting color. If a function, function need to return a character vector of colors.

part
  partname of the table (one of 'all', 'body', 'header', 'footer')

source
  if color is a function, source is specifying the dataset column to be used as argument to color. This is only useful if j is colored with values contained in other columns.

See Also

Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), fontsize(), font(), italic(), keep_with_next(), line_spacing(), padding(), rotate(), 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

Description

The function is applying an horizontal border to inner content of one or all parts of a flextable. The lines are the bottom borders of selected cells.

Usage

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

- **x**: a flextable object
- **i**: rows selection
- **j**: columns selection
- **border**: border properties defined by a call to `fp_border()`
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

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

Examples

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

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

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

---

**hline_bottom**

*Set bottom horizontal border*

Description

The function is applying an horizontal border to the bottom of one or all parts of a flextable. The line is the bottom border of selected parts.

Usage

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

Arguments

- **x**: a flextable object
- **j**: columns selection
- **border**: border properties defined by a call to `fp_border()`
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')
See Also

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

Examples

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

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

# add/replace horizontal border on bottom
ft <- hline_bottom(ft, part = "body", border = big_border)
ft
```

---

**hline_top**  
*Set top horizontal border*

### Description

The function is applying an horizontal border to the top of one or all parts of a flextable. The line is the top border of selected parts.

### Usage

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

### Arguments

- `x`: a flextable object
- `j`: columns selection
- `border`: border properties defined by a call to `fp_border()`
- `part`: partname of the table (one of 'all', 'body', 'header', 'footer')

### See Also

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

### Examples

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

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

*Set flextable rule for rows heights*

**Description**

control rules of each height for a part of the flextable, this is only for Word and PowerPoint outputs, it will not have any effect when output is HTML or PDF.

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

**Usage**

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

**Arguments**

- `x` flextable object
- `i` rows selection
- `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` partname of the table, one of "all", "header", "body", "footer"

**See Also**

Other flextable dimensions: `autofit()`, `dim.flextable()`, `dim.pretty()`, `fit_to_width()`, `flextable_dim()`, `height()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`, `width()`

**Examples**

```r
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

Description

get a 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
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 HTML ready to be used within a call to shiny::renderUI for example.

See Also

Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), gen_grob(), 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()

Examples

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

Chunk of text with hyperlink

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

```r
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.

See Also

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

Examples

```r
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",
```
italic

value = as_paragraph(
    "This is a link: ",
    hyperlink_text(x = col, url = href)
)
)
)
ftab

\begin{itemize}
\item \textbf{italic} \textit{Set italic font}
\end{itemize}

\section*{Description}

change font decoration of selected rows and columns of a flextable.

\section*{Usage}

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

\section*{Arguments}

\begin{itemize}
\item \textbf{x} \hspace{1cm} a flextable object
\item \textbf{i} \hspace{1cm} rows selection
\item \textbf{j} \hspace{1cm} columns selection
\item \textbf{italic} \hspace{1cm} boolean value
\item \textbf{part} \hspace{1cm} partname of the table (one of 'all', 'body', 'header', 'footer')
\end{itemize}

\section*{See Also}

Other sugar functions for table style: \texttt{align()}, \texttt{bg()}, \texttt{bold()}, \texttt{color()}, \texttt{empty_blanks()}, \texttt{fontsize()}, \texttt{font()}, \texttt{highlight()}, \texttt{keep_with_next()}, \texttt{line_spacing()}, \texttt{padding()}, \texttt{rotate()}, \texttt{valign()}

\section*{Examples}

\begin{verbatim}
ft <- flextable(head(mtcars))
ft <- italic(ft, italic = TRUE, part = "header")
\end{verbatim}
keep_with_next

Description

The 'Keep with next' functionality in 'Word', applied to the rows of a table, ensures that the rows with that attribute stays together and does not break across multiple pages.

This function allows much better control of breaks between pages than the global keep_with_next parameter.

Usage

```r
keep_with_next(x, i = NULL, value = TRUE, part = "body")
```

Arguments

- **x**: a flextable object
- **i**: rows selection
- **value**: TRUE or FALSE. When applied to a group, all rows except the last one should be flagged with attribute 'Keep with next'.
- **part**: partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

paginate()

Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), fontsize(), font(), highlight(), italic(), line_spacing(), padding(), rotate(), valign()

Examples

```r
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"))
```
Description

Function used to render flextable in knitr/rmarkdown documents.

You should not call this method directly. This function is used by the knitr package to automatically display a flextable in an "R Markdown" document from a chunk. However, it is recommended to read its documentation in order to get familiar with the different options available.

R Markdown outputs can be:

- HTML
- 'Microsoft Word'
- 'Microsoft PowerPoint'
- PDF

Table captioning is a flextable feature compatible with R Markdown documents. The feature is available for HTML, PDF and Word documents. Compatibility with the "bookdown" package is also ensured, including the ability to produce captions so that they can be used in cross-referencing. For Word, it's recommended to work with package 'officedown' that supports all features of flextable.

Usage

```r
## S3 method for class 'flextable'
knit_print(x, ...)
```

Arguments

- `x`: a flextable object
- `...`: unused.

Chunk options

Some features, often specific to an output format, are available to help you configure some global settings relative to the table output. knitr’s chunk options are to be used to change the default settings:

- HTML, PDF and Word:
  - `ft.align`: flextable alignment, supported values are 'left', 'center' and 'right'. Its default value is 'center'.
- HTML only:
  - `ft.htmlscroll`, can be TRUE or FALSE (default) to enable horizontal scrolling. Use `set_table_properties()` for more options about scrolling.
- Word only:
  - `ft.split` Word option 'Allow row to break across pages’ can be activated when TRUE (default value).
- `ft.keepnext` defunct in favor of `paginate()`

**PDF only:**
- `ft.tabcolsep` space between the text and the left/right border of its containing cell, the default value is 0 points.
- `ft.arraystretch` height of each row relative to its default height, the default value is 1.5.
- `ft.latex.float` type of floating placement in the 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

**PowerPoint only:**
- `ft.left`, `ft.top` Position should be defined with these options. These are the top left coordinates in inches of the placeholder that will contain the table. Their default values are 1 and 2 inches.

If some values are to be used all the time in the same document, it is recommended to set these values in a 'knitr r chunk' by using function `knitr::opts_chunk$set(ft.split=FALSE, ...)`. 

### Table caption

Captions can be defined in two ways.

The first is with the `set_caption()` function. If it is used, the other method will be ignored. The second method is by using knitr chunk option `tab.cap`.

```r
set_caption(x, caption = "my caption")
```

If `set_caption` function is not used, caption identifier will be read from knitr's chunk option `tab.id`. Note that in a bookdown and when not using `officedown::rdocx_document()`, the usual numbering feature of bookdown is used.

`tab.id='my_id'`.

Some options are available to customise captions for any output:

<table>
<thead>
<tr>
<th>label</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word stylename to use for table captions.</td>
<td><code>tab.cap.style</code></td>
<td>NULL</td>
</tr>
<tr>
<td>caption id/bookmark</td>
<td><code>tab.id</code></td>
<td>NULL</td>
</tr>
<tr>
<td>caption</td>
<td><code>tab.cap</code></td>
<td>NULL</td>
</tr>
<tr>
<td>display table caption on top of the table or not</td>
<td><code>tab.topcaption</code></td>
<td>TRUE</td>
</tr>
</tbody>
</table>
| caption table sequence identifier. | `tab.lp` | "tab:"

Word output when `officedown::rdocx_document()` is used is coming with more options such as ability to choose the prefix for numbering chunk for example. The table below expose these options:
HTML output

HTML output is using shadow dom to encapsule the table into an isolated part of the page so that no clash happens with styles.

PDF output

Some features are not implemented in PDF due to technical infeasibility. These are the padding, line_spacing and height properties. Note also justified text is not supported and is transformed to left.

It is recommended to set these values in a `knitr r chunk` so that they are permanent all along the document: `knitr::opts_chunk$set(ft.tabcolsep=0, ft.latex.float = "none")`.

See `add_latex_dep()` if caching flextable results in 'R Markdown' documents.

PowerPoint output

Auto-adjust Layout is not available for PowerPoint, PowerPoint only support fixed layout. It’s then often necessary to call function `autofit()` so that the columns’ widths are adjusted if user does not provide the withs.

Images cannot be integrated into tables with the PowerPoint format.

Note

Supported formats require some minimum pandoc versions:

<table>
<thead>
<tr>
<th>Output format</th>
<th>pandoc minimal version</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>&gt;= 1.12</td>
</tr>
<tr>
<td>Word (docx)</td>
<td>&gt;= 2.0</td>
</tr>
<tr>
<td>PowerPoint (pptx)</td>
<td>&gt;= 2.4</td>
</tr>
<tr>
<td>PDF</td>
<td>&gt;= 1.12</td>
</tr>
</tbody>
</table>

See Also

paginate()

Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), plot.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.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**  
Change displayed labels

**Description**

The function replace text values in a flextable with labels. The labels are defined with character named vector.

The function is not written to be fast but to be handy. It does not replace the values in the underlying dataset but replace the defined content in the flextable (as defined with `compose()`).

**Usage**

`labelizor(x, j = NULL, labels, part = "all")`

**Arguments**

- `x` a flextable object
- `j` columns selection
- `labels` a named vector whose names will be used to identify values to replace and values will be used as labels.
- `part` partname of the table (one of 'all', 'body', 'header', 'footer')

**See Also**

`mk_par()`, `append_chunks()`, `prepend_chunks()`
Examples

```r
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 wrapper*

**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"
)
```
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".

Note

This chunk option requires package officedown in a R Markdown context with Word output format. PowerPoint cannot mix images and text in a paragraph, images are removed when outputting to PowerPoint format.

See Also

compose(), as_paragraph()

Other chunk elements for paragraph: as_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), lollipop(), 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_scaping

Set text alignment

Description
change text alignment of selected rows and columns of a flextable.

Usage
line_scaping(x, i = NULL, j = NULL, space = 1, part = "body")

Arguments
x a flextable object
i rows selection
j columns selection
space space between lines of text, 1 is single line spacing, 2 is double line spacing.
part partname of the table (one of 'all', 'body', 'header', 'footer')

See Also
Other sugar functions for table style: align(), bg(), bold(), color().empty.blanks(), fontsize(), font(), highlight(), italic(), keep.with.next(), padding(), rotate(), valign()

Examples
ft <- flextable(head(mtcars)[, 3:6])
ft <- line_scaping(ft, space = 1.6, part = "all")
ft <- set_table_properties(ft, layout = "autofit")
ft

Lollipop

Mini lollipop chart chunk wrapper

Description
This function is used to insert lollipop charts into flextable with functions:

- compose() and as_paragraph(),
- appendchunks(),
- prependchunks().
Usage

```r
lollipop(
  value,
  min = NULL,
  max = NULL,
  rangecol = "#CCCCCC",
  bg = "transparent",
  width = 1,
  height = 0.2,
  unit = "in",
  raster_width = 30,
  positivecol = "#00CC00",
  negativecol = "#CC0000",
  neutralcol = "#CCCCCC",
  neutralrange = c(0, 0),
  rectanglesize = 2
)
```

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
- **bg**: background color
- **width, height**: size of the resulting png file in inches
- **unit**: unit for width and height, one of "in", "cm", "mm".
- **raster_width**: number of pixels used as width
- **positivecol**: box color of positive values
- **negativecol**: box color of negative values
- **neutralcol**: box color of neutral values
- **neutralrange**: minimal and maximal range of neutral values (default: 0)
- **rectanglesize**: size of the rectangle (default: 2, max: 5) when interpolating value.

Note

This chunk option requires package officedown in a R Markdown context with Word output format. PowerPoint cannot mix images and text in a paragraph, images are removed when outputting to PowerPoint format.

See Also

- `compose()`, `as_paragraph()`
- Other chunk elements for paragraph: `as_bracket()`, `as_b()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_image()`, `as_1()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `minibar()`, `plot_chunk()`
merge_at

Examples

```r
ft <- flextable(tail(iris, n = 10))

ft <- compose(ft,
  j = "Sepal.Ratio", value = as_paragraph(
    lollipop(value = Sepal.Ratio, min = -.25, max = .25)
  ),
  part = "body"
)

ft <- autofit(ft)
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

```r
merge_at(x, i = NULL, j = NULL, part = "body")
```

Arguments

- `x`: flextable object
- `i, j`: columns and rows to merge
- `part`: partname of the table where merge has to be done.

See Also

Other flextable merging function: `merge_h_range()`, `merge_h()`, `merge_none()`, `merge_v()`

Examples

```r
ft_merge <- flextable(head(mtcars), cwidth = .5)
ft_merge <- merge_at(ft_merge, i = 1:2, j = 1:2)
ft_merge
```
merge_h

Merger 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
flextable object

i
rows where cells have to be merged.

part
partname of the table where merge has to be done.

See Also

Other flextable merging function: merge_at(), merge_h_range(), merge_none(), merge_v()

Examples

dummy_df <- data.frame(
  col1 = letters,
  col2 = letters, stringsAsFactors = FALSE
)
dummy_df <- flextable(dummy_df)
dummy_df <- merge_h(x = dummy_df)
dummy_df

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")
merge_none

Arguments

x flextable object
i selected rows
j1, j2 selected columns that will define the range of columns to merge.
part partname of the table where merge has to be done.

See Also

Other flextable merging function: `merge_at()`, `merge_h()`, `merge_none()`, `merge_v()`

Examples

```r
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 informations

Description

Delete all merging informations from a flextable.

Usage

`merge_none(x, part = "all")`

Arguments

x flextable object
part partname of the table where merge has to be done.

See Also

Other flextable merging function: `merge_at()`, `merge_h_range()`, `merge_h()`, `merge_v()`

Examples

```r
typology <- data.frame(
  measure = c("Length", "Width", "Length", "Width", "Species"),
  stringsAsFactors = FALSE
)
```
merge_v <- flextable(head(iris))
merge_v <- set_header_df(merge_v, mapping = typology, key = "col_keys")
merge_v <- merge_v(merge_v, j = c("Species"))
merge_v <- theme_tron_legacy(merge_v)
merge_v

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 flextable object
j column to used to find consecutive values to be merged. Columns from original dataset can also be used.
target columns names where cells have to be merged.
part partname of the table where merge has to be done.
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 flextable merging function: merge_at(), merge_h_range(), merge_h(), 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", "another", "yet_another")))

minibar

```
"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"))
```

minibar

Mini barplots chunk wrapper

**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"
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>values containing the bar size</td>
</tr>
<tr>
<td>max</td>
<td>max bar size</td>
</tr>
<tr>
<td>barcol</td>
<td>bar color</td>
</tr>
<tr>
<td>bg</td>
<td>background color</td>
</tr>
<tr>
<td>width, height</td>
<td>size of the resulting png file in inches</td>
</tr>
<tr>
<td>unit</td>
<td>unit for width and height, one of &quot;in&quot;, &quot;cm&quot;, &quot;mm&quot;.</td>
</tr>
</tbody>
</table>

Note

This chunk option requires package officedown in a R Markdown context with Word output format.

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_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), grid_chunk(), hyperlink_text(), linerange(), lollipop(), plot_chunk()

Examples

```r
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
```

<table>
<thead>
<tr>
<th>ncol_keys</th>
<th>Number of columns</th>
</tr>
</thead>
</table>

Description

returns the number of columns displayed

Usage

col_keys(x)
Arguments

x flextable object

See Also

Other flextable dimensions: autofit(), dim.flextable(), dim_pretty(), fit_to_width(), flextable_dim(), height(), hrule(), nrow_part(), set_table_properties(), width()

Examples

library(flextable)
ft <- qflextable(head(cars))
ncol_keys(ft)

<table>
<thead>
<tr>
<th>nrow_part</th>
<th>Number of rows of a part</th>
</tr>
</thead>
</table>

Description

returns the number of lines in a part of flextable.

Usage

nrow_part(x, part = "body")

Arguments

x flextable object

part partname of the table (one of 'body', 'header', 'footer')

See Also

Other flextable dimensions: autofit(), dim.flextable(), dim_pretty(), 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 paddings 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
i rows selection
j columns selection
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 partname of the table (one of 'all', 'body', 'header', 'footer')

See Also
Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), fontsize(), font(), highlight(), italic(), keep_with_next(), line_spacing(), rotate(), 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)
paginate <- 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  Paginate tables

**Description**

Prevents breaks between tables rows you want to stay together. This feature only applies to Word and RTF output.

**Usage**

```r
paginate(
  x, 
  init = NULL, 
  hdr_ftr = TRUE, 
  group = character(), 
  group_def = c("rle", "nonempty")
)
```

**Arguments**

- **x**: 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
- **group_def**: algorithm to be used to identify groups that should not be split into two pages, one of 'rle', 'nonempty':
  - '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()`.

**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:

```r
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:

\[
\text{paginate}(x, \text{init} = \text{FALSE}, \text{hdr_ftr} = \text{TRUE})
\]

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.

\[
\text{paginate}(x, \text{group} = \text{“grp”}, \text{group_def} = \text{”rle”})
\]

**Value**

updated flextable object

**Examples**

```r
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) |> paginate(group = "cut", group_def = "rle")

save_as_docx(ft_1, path = tempfile(fileext = ".docx"))
save_as_rtf(ft_1, path = tempfile(fileext = ".rtf"))
```
Add a flextable into a PowerPoint slide

Description

Add a flextable in a PowerPoint document object produced by `officer::read_pptx()`.

Usage

```r
## 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.

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).

Examples

```r
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)
```
Description
plots a flextable as a grid grob object and display the result in a new graphics window. 'ragg' or 'svglite' or 'ggraph' graphical device drivers should be used to ensure a correct rendering.

Usage
## S3 method for class 'flextable'
plot(x, ...)

Arguments
x
  a flextable object

...  additional arguments passed to gen_grob().

See Also
Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable()

Examples
library(gdtools)
library(ragg)
register_liberationsans()
set_flextable_defaults(font.family = "Liberation Sans")
ftab <- as_flextable(cars)

tf <- tempfile(fileext = ".png")
agg_png(
  filename = tf, width = 1.7, height = 3.26, unit = "in",
  background = "transparent", res = 150
)
plot(ftab)
dev.off()
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 plots chunk wrapper

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",
  ...
)
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".
- **...**: arguments sent to plot functions (see `par()`)

Note

This chunk option requires package officedown in a R Markdown context with Word output format. 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_bracket()`, `as_b()`, `as_chunk()`, `as_equation()`, `as_highlight()`, `as_image()`, `as_i()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_chunk()`, `grid_chunk()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`

Examples

```r
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

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

Arguments

- `x`: a flextable object
- `...`: chunks to be prepended, see `as_chunk()`, `gg_chunk()` and other chunk elements for paragraph.
- `i`: rows selection
- `j`: column selection
- `part`: partname of the table (one of 'body', 'header', 'footer')

See Also

Other functions for mixed content paragraphs: `append_chunks()`, `as_paragraph()`, `compose()`

Examples

```r
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

flextable printing

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

```r
## S3 method for class 'flextable'
print(x, preview = "html", align = "center", ...)
```

Arguments

- `x`: 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 flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`

---

**proc_freq**

*Frequency table*

Description

This function compute a one or two way contingency table and create 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

```r
proc_freq(
  x,
  row = character(),
  col = character(),
  include.row_percent = TRUE,
  include.column_percent = TRUE,
)
rotate

include.table_percent = TRUE,
weight = character(),
...
)

Arguments

x  a data.frame object containing variable(s) to use for counts.
row character column names for row
col character 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
weight character column name for weight
... unused arguments

Examples

proc_freq(mtcars, "vs", "gear")
proc_freq(mtcars, "gear", "vs", weight = "wt")

rotate

Rotate cell text

Description

It can be useful to be able to change the direction, when the table headers are huge for example, header labels can be rendered as "tbrl" (top to bottom and right to left) corresponding to a 90 degrees rotation or "btlr" corresponding to a 270 degrees rotation. The function change cell text direction. By default, it is "ltrb" which mean from left to right and top to bottom.

'Word' and 'PowerPoint' don't handle auto height with rotated headers. So you need to set header heights (with function height()) and set rule "exact" for rows heights (with function hrule()) otherwise Word and PowerPoint outputs will have small height not corresponding to the necessary height to display the text.

flextable doesn’t do the rotation by any angle. It only rotates by a number of right angles. This choice is made to ensure the same rendering between 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

\texttt{x} \quad \text{a flextable object}
\texttt{i} \quad \text{rows selection}
\texttt{j} \quad \text{columns selection}
\texttt{rotation} \quad \text{one of "lrtb", "tbrl", "btlr".}
\texttt{align} \quad \text{vertical alignment of paragraph within cell, one of "center" or "top" or "bottom".}
\texttt{part} \quad \text{partname of the table (one of 'all', 'body', 'header', 'footer')} \\

Details

When function \texttt{autofit} is used, the rotation will be ignored. In that case, use \texttt{dim_pretty} and \texttt{width} instead of \texttt{autofit}.

See Also

Other sugar functions for table style: \texttt{align()}, \texttt{bg()}, \texttt{bold()}, \texttt{color()}, \texttt{empty_blanks()}, \texttt{fontsize()}, \texttt{font()}, \texttt{highlight()}, \texttt{italic()}, \texttt{keep_with_next()}, \texttt{line_spacing()}, \texttt{padding()}, \texttt{valign()}

Examples

library(flextable)

\texttt{ft_1 <- flextable(head(iris))}
\texttt{ft_1 <- rotate(ft_1, j = 1:4, align = "bottom", rotation = "tbrl", part = "header")}
\texttt{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
\texttt{ft_1 <- height(ft_1, height = 1.2, part = "header")}
\texttt{ft_1 <- hrule(ft_1, i = 1, rule = "exact", part = "header")}

\texttt{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")
)

\texttt{ft_2 <- flextable(dat)}
\texttt{ft_2 <- theme_box(ft_2)}
\texttt{ft_2 <- height_all(x = ft_2, height = 1.3, part = "body")}
\texttt{ft_2 <- hrule(ft_2, rule = "exact")}
\texttt{ft_2 <- rotate(ft_2, rotation = "tbrl")}
\texttt{ft_2 <- width(ft_2, width = 1.3)}

\texttt{ft_2 <- align(ft_2, j = 1, align = "left")}
\texttt{ft_2 <- align(ft_2, j = 2, align = "center")}
Add a 'flextable' into an RTF document

**Description**

*rtf_add()* method for adding flextable objects into 'RTF' documents.

**Usage**

```r
## S3 method for class 'flextable'
rtf_add(x, value, ...)
```

**Arguments**

- `x`  
  rtf object, created by *rtf_doc()*.
- `value`  
  a flextable object
- `...`  
  unused arguments

**Examples**

```r
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 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 flextable print function: as_raster(), df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), save_as_html(), save_as_image(), save_as_pptx(), save_as_rtf(), to_html.flextable()

Examples

```r

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",
...
```
**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 recommended to use R Markdown instead).

**Usage**

```r
save_as_html(..., values = NULL, path, lang = "en", title = "")
```

**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 flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `gen_grob()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_image()`, `save_as_pptx()`, `save_as_rtf()`, `to_html.flextable()`
Examples

```r
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.

Usage

```r
save_as_image(x, path, expand = 10, res = 200, ...)
```

Arguments

- **x**: a 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

See Also

Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), gen_grob(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_pptx(), save_as_rtf(), to_html_flextable()
Examples

library(gdtools)
register_liberationsans()
set_flextable_defaults(font.family = "Liberation Sans")

ft <- flextable(head(mtcars))
ft <- autofit(ft)
if <- 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 a 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 flextable print function: as_raster(), 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_rtf(), to_html.flextable()
Examples

```r
ft1 <- flextable(head(iris))
ft <- tempfile(fileext = ".pptx")
save_as_pptx(ft1, path = tf)

ft2 <- flextable(head(mtcars))
ft <- 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

```r
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 `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 flextable print function: `as_raster()`, `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()`, `to_html.flextable()`
Examples

```r
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")),
    qflexttable(data.frame(a = 1L))
  )
)

tf <- tempfile(fileext = "rtf")
save_as_rtf(
  'iris table' = ft1, 'mtcars table' = ft2,
  path = tf, pr_section = sect_properties
)
```

---

**separate_header**

*Separate collapsed colnames into multiple rows*

**Description**

If your variable names contain multiple delimited labels, they will be separated and placed in their own rows.

**Usage**

```r
separate_header(
  x,
  opts = c("span-top", "center-hspan", "bottom-vspan", "default-theme"),
  split = "\[\[_\]\]",
  fixed = FALSE
)
```

**Arguments**

- *x* a flextable object
set_caption

opts  optional treatments to apply to the resulting header part as a character vector with multiple supported values. The supported values are:
  • "span-top": span empty cells with the first non empty cell, this operation is made column by column.
  • "center-hspan": center the cells that are horizontally spanned.
  • "bottom-vspan": bottom align the cells treated when "span-top" is applied.
  • "default-theme": apply to the new header part the theme set in set_flextable_defaults(theme_fun = ...).

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 functions for row and column operations in a flextable: add_body_row(), add_body(), add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), add_header(), delete_columns(), delete_part(), delete_rows(), set_header_footer_df, set_header_labels()

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
**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**

```r
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` - 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.
set_caption

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 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 alignment 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:

<table>
<thead>
<tr>
<th>label</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word stylename to use for table captions.</td>
<td>tab.cap.style</td>
<td>NULL</td>
</tr>
<tr>
<td>caption id/bookmark</td>
<td>tab.id</td>
<td>NULL</td>
</tr>
<tr>
<td>caption</td>
<td>tab.cap</td>
<td>NULL</td>
</tr>
<tr>
<td>display table caption on top of the table or not</td>
<td>tab.topcaption</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
| 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.ind     | 0                        |
| separator to use between title number and table number. | tab.cap.ins     | "."                     |
| 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()`

Examples

```r
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

```r
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,
  split = NULL,
  keep_with_next = NULL,
  tabcolsep = NULL,
  arraystretch = 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).
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 strftime(). 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()`.

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.

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()`.

deprecated arguments...

Value

a list containing previous default values.

See Also

Other functions related to themes: `get_flextable_defaults()` , `theme_alafoli()` , `theme_apa()` , `theme_booktabs()` , `theme_box()` , `theme_tron_legacy()` , `theme_tron()` , `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")`

`set_formatter_type(
    x,
    fmt_double = "%.03f",
    fmt_integer = "%d",
    fmt_date = "%Y-%m-%d",
    fmt_datetime = "%Y-%m-%d %H:%M:%S",
    true = "true",
    false = "false",
    na_str = ""
)`

**Arguments**

- `x` a 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 of the table (one of 'body' or 'header' or 'footer') where to apply the formatter functions.
- `fmt_double, fmt_integer` arguments used by `sprintf` to format double and integer columns.
set_header_footer_df

fmt_date, fmt_datetime
  arguments used by format to format date and date time columns.
false, true  string to be used for logical columns
na_str  string for NA values

set_formatter_type

set_formatter_type is an helper function to quickly define formatter functions regarding to column types.
This function will be deprecated in favor of the colformat_* functions, for example colformat_double().

See Also

Other cells formatters: colformat_char(), colformat_datetime(), colformat_date(), 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  Set flextable’s header or footer rows

Description

Use a data.frame to specify flextable’s header or footer rows.
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
- `mapping` a data.frame specifying for each colname content of the column.
- `key` column to use as key when joining `data_mapping`.

See Also

Other functions for row and column operations in a flextable: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_row()`, `add_header()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `separate_header()`, `set_header_labels()`

Examples

typology <- data.frame(
  col_keys = c(
    "Petal.Width", "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(
    "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

Change headers labels

Description

This function set labels for specified columns in the bottom row header of a flextable.

Usage

```
set_header_labels(x, ..., values = NULL)
```

Arguments

- `x` a 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 unnamed character vector, in that case, it must have the same length than the number of columns of the flextable.

See Also

Other functions for row and column operations in a flextable: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_row()`, `add_header()`, `delete_columns()`, `delete_part()`, `delete_rows()`, `separate_header()`, `set_header_footer_df`

Examples

```r
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
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

```r
set_table_properties(
  x,
  layout = "fixed",
  width = 0,
  align = "center",
  opts_html = list(),
  opts_word = list(),
  opts_pdf = list(),
  word_title = NULL,
  word_description = NULL
)
```

Arguments

- `x` : flextable object
- `layout` : 'autofit' or 'fixed' algorithm. Default to 'autofit'.
- `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;`

**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.

**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.
• '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.
• '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**

Other flextable dimensions: `autofit()`, `dim.flextable()`, `dim.pretty()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `width()`
Examples

```r
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",
)`
shift_table


cn_grade = "LBNRIND",
cn_usubjid = "USUBJID",
cn_lab_cat = NA_character_,
cn_is_baseline = "LBBFL",
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 identifiers, 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 "LBBFL".

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. Additional 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'n' = as_paragraph(N),
  \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
Description

Modify flextable text, paragraphs and cells formatting properties. It allows to specify a set of formatting properties for a selection instead of using multiple functions (i.e. bold, italic, bg) that should all be applied to the same selection of rows and columns.

Usage

```
style(
  x,
  i = NULL,
  j = NULL,
  pr_t = NULL,
  pr_p = NULL,
  pr_c = NULL,
  part = "body"
)
```

Arguments

- **x**: a flextable object
- **i**: rows selection
- **j**: columns selection
- **pr_t**: object(s) of class `fp_text`
- **pr_p**: object(s) of class `fp_par`
- **pr_c**: object(s) of class `fp_cell`
- **part**: partname of the table (one of ‘all’, ‘body’, ‘header’ or ‘footer’)

Examples

```r
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  

Data summary preparation

Description

It performs a univariate statistical analysis of a dataset by group and formats the results so that they can be used with the \texttt{tabulator()} function or directly with \texttt{as_flextable}.

Usage

\begin{verbatim}
summarizor(
  x,
  by = character(),
  overall_label = NULL,
  num_stats = c("mean_sd", "median_iqr", "range"),
  hide_null_na = TRUE
)
\end{verbatim}

Arguments

\begin{itemize}
  \item \textit{x} \hspace{1cm} dataset
  \item \textit{by} \hspace{1cm} columns names to be used as grouping columns
  \item \textit{overall_label} \hspace{1cm} label to use as overall label
  \item \textit{num_stats} \hspace{1cm} available statistics for numerical columns to show, available options are "mean_sd", "median_iqr" and "range".
  \item \textit{hide_null_na} \hspace{1cm} if TRUE (default), NA counts will not be shown when 0.
\end{itemize}

Note

This is very first version of the function; be aware it can evolve or change.

See Also

\texttt{fmt_summarizor()}, \texttt{labelizor()}

Examples

\begin{verbatim}
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
\end{verbatim}
z <- summarizer(CO2[-c(1, 4)])
ft_3 <- as_flexttable(z, sep_w = 0, spread_first_col = TRUE)
ft_3

---

**surround**

Set borders for a selection of cells

**Description**

Highlight specific cells with borders.

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**

```r
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
- `i`  rows selection
- `j`  columns selection
- `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`  partname of the table (one of 'all', 'body', 'header', 'footer')
See Also

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

Examples

```r
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**

Tabulation of aggregations
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

```r
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)))
  )
### Example 3 ----

#### data.table version

```r
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

```r
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")
```

```r
tab_3 <- tabulator(
  x = dat, rows = c("Species"),
  columns = "name",
  \`mean (sd)` = as_paragraph(  
    as_chunk(avg),
    " (", as_chunk(sd), ")"
  )
)
```

```r
ft_3 <- as_flextable(tab_3)
ft_3
```

### tabulator_colnames

#### Description

The function provides a way to get column keys associated with the flextable corresponding to a `tabulator()` object. It helps in customizing or programming 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

```r
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, 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, 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
}
theme_apa

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 additional 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 functions related to themes: get_flextable_defaults(), set_flextable_defaults(), theme_apa(), theme_booktabs(), theme_box(), theme_tron_legacy(), theme_tron(), theme_vader(), theme_vanilla(), theme_zebra()

Examples

```r
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**

```r
theme_apa(x, ...)
```

**Arguments**

- `x` a flextable object
- `...` unused
behaviour

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 additional 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 functions related to themes: get_flextable_defaults(), set_flextable_defaults(), theme_alafoli(), theme_booktabs(), theme_box(), theme_tron_legacy(), theme_tron(), theme_vader(), theme_vanilla(), theme_zebra()

Examples

```r
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
- bold_header: header will be bold if TRUE.
- ...: unused
theme_box

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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```r
ft <- flextable(head(airquality))
ft <- theme_booktabs(ft)
ft
```

---

### theme_box

**Apply box theme**

**Description**

Apply theme box to a flextable

**Usage**

`theme_box(x)`

**Arguments**

- `x` a 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 additional 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 functions related to themes: get_flextable_defaults(), set_flextable_defaults(), theme_alafoli(), theme_apa(), theme_booktabs(), theme_tron_legacy(), theme_tron(), theme_vader(), theme_vanilla(), theme_zebra

Examples

```r
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
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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```r
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**

```r
theme_tron_legacy(x)
```

**Arguments**

- `x` a 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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`

Examples

```r
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**

```r
theme_vader(x, ...)
```

**Arguments**

- `x`: a `flextable` object
- `...`: unused
### theme_vanilla

**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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vanilla()`, `theme_zebra()`

### Examples

```r
ft <- flextable(head(airquality))
ft <- theme_vader(ft)
ft
```

<table>
<thead>
<tr>
<th>theme_vanilla</th>
<th>Apply vanilla theme</th>
</tr>
</thead>
</table>

### 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

```r
theme_vanilla(x)
```

#### Arguments

- `x` a 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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vader()`, `theme_zebra()`

Examples

```r
ft <- flextable(head(airquality))
ft <- theme_vanilla(ft)
ft
```

<table>
<thead>
<tr>
<th>theme_zebra</th>
<th>Apply zebra theme</th>
</tr>
</thead>
</table>

Description

Apply theme zebra to a flextable

Usage

```r
theme_zebra(
  x,
  odd_header = "#CFCFCF",
  odd_body = "#EFEFEF",
  even_header = "transparent",
  even_body = "transparent"
)
```
Arguments

- `x`: a 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 additional 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 functions related to themes: `get_flextable_defaults()`, `set_flextable_defaults()`, `theme_alafoli()`, `theme_apa()`, `theme_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`

Examples

```r
ft <- flextable(head(airquality))
ft <- theme_zebra(ft)
ft
```

Description

Generate HTML code of corresponding flextable as an HTML table or an HTML image.

Usage

```r
## S3 method for class 'flextable'
to_html(x, type = c("table", "img"), ...)
```
Arguments

\begin{itemize}
\item \textbf{x} \hspace{1cm} a flextable object
\item \textbf{type} \hspace{1cm} output type. one of "table" or "img".
\item \ldots \hspace{1cm} unused
\end{itemize}

Value

If \texttt{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 flextable print function: \texttt{as_raster()}, \texttt{df_printer()}, \texttt{flextable_to_rmd()}, \texttt{gen_grob()}, \texttt{htmltools_value()}, \texttt{knit_print.flextable()}, \texttt{plot.flextable()}, \texttt{print.flextable()}, \texttt{save_as_docx()}, \texttt{save_as_html()}, \texttt{save_as_image()}, \texttt{save_as_pptx()}, \texttt{save_as_rtf()}

Examples

\begin{verbatim}
library(officer)
library(flextable)
x <- to_html(as_flextab(cars))
\end{verbatim}

\begin{verbatim}
use_df_printer
\end{verbatim}

Set data.frame automatic printing as a flextable

Description

Define \texttt{df_printer()} as data.frame print method in an R Markdown document.

In a setup run chunk:

\begin{verbatim}
flextable::use_df_printer()
\end{verbatim}

Usage

\begin{verbatim}
use_df_printer()
\end{verbatim}

See Also

\texttt{df_printer()}, \texttt{flextable()}

\texttt{use_df_printer} Set data.frame automatic printing as a flextable
use_model_printer  set model automatic printing as a flextable

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()

valign  Set vertical alignment

Description

change 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
i  rows selection
j  columns selection
valign  vertical alignment of paragraph within cell, one of "center" or "top" or "bottom".
part  partname of the table (one of 'all', 'body', 'header', 'footer')
vline

Set vertical borders

Description

The function is applying vertical borders to inner content of one or all parts of a flextable. The lines are the right borders of selected cells.

Usage

vline(x, i = NULL, j = NULL, border = NULL, part = "all")

Arguments

- x: a flextable object
- i: rows selection
- j: columns selection
- border: border properties defined by a call to fp_border()
- part: partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

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

Examples

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

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

Set flextable left vertical borders

Description

The function is applying vertical borders to the left side of one or all parts of a flextable. The line is the left border of selected cells of the first column.

Usage

vline_left(x, i = NULL, border = NULL, part = "all")

Arguments

x a flextable object
i rows selection
border border properties defined by a call to fp_border()
part partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

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

Examples

library(officer)
std_border <- fp_border(color = "orange")

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

# add vertical border on the left side of the table
ft <- vline_left(ft, border = std_border)
ft
vline_right

Description

The function is applying vertical borders to the right side of one or all parts of a flextable. The line is the right border of selected cells of the last column.

Usage

vline_right(x, i = NULL, border = NULL, part = "all")

Arguments

x a flextable object
i rows selection
border border properties defined by a call to fp_border()
part partname of the table (one of 'all', 'body', 'header', 'footer')

See Also

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

Examples

library(officer)
std_border <- fp_border(color = "orange")

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

# add vertical border on the left side of the table
ft <- vline_right(ft, border = std_border)
ft

void

Description

Set content display as a blank " ".

Usage

void(x, j = NULL, part = "body")
width

Arguments

- x: a `flextable` object
- j: columns selection
- part: partname of the table

Examples

```r
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

```r
width(x, j = NULL, width, unit = "in")
```

Arguments

- x: a `flextable()` object
- j: columns selection
- 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 flextable dimensions: `autofit()`, `dim.flextable()`, `dim.pretty()`, `fit_to_width()`, `flextable_dim()`, `height()`, `hrule()`, `ncol_keys()`, `nrow_part()`, `set_table_properties()`

Examples

```r
ft <- flextable(head(iris))
ft <- width(ft, width = 1.5)
ft
```
Index

* as_flextable methods
  as_flextable, 22
  as_flextable.data.frame, 22
  as_flextable.gam, 24
  as_flextable.glm, 25
  as_flextable.grouped_data, 25
  as_flextable.htest, 26
  as_flextable.kmeans, 27
  as_flextable.lm, 28
  as_flextable.merMod, 29
  as_flextable.pam, 30
  as_flextable.summarizer, 31
  as_flextable.table, 32
  as_flextable.tabular, 32
  as_flextable.tabulator, 35
  as_flextable.xtable, 37

* borders management
  border_inner, 53
  border_inner_h, 53
  border_inner_v, 54
  borderOuter, 55
  border_remove, 56
  hline, 101
  hline_bottom, 102
  hline_top, 103
  surround, 162
  vline, 180
  vline_left, 181
  vline_right, 182

* cells formatters
  colformat_char, 56
  colformat_date, 57
  colformat_datetime, 58
  colformat_double, 59
  colformat_image, 60
  colformat_int, 61
  colformat_lgl, 62
  colformat_num, 63
  set_formatter, 150

* chunk elements for paragraph
  as_b, 18
  as_bracket, 19
  as_chunk, 20
  as_equation, 21
  as_highlight, 40
  as_i, 40
  as_image, 41
  as_sub, 43
  as_sup, 44
  as_word_field, 45
  colorize, 66
  gg_chunk, 97
  grid_chunk, 98
  hyperlink_text, 106
  linerange, 113
  lollipop, 115
  minibar, 121
  plot_chunk, 129

* flextable dimensions
  autofit, 47
  dim.flextable, 72
  dim_pretty, 73
  fit_to_width, 75
  flextable_dim, 79
  height, 99
  hrule, 104
  ncol_keys, 122
  nrow_part, 123
  set_table_properties, 154
  width, 183

* flextable merging function
  merge_at, 117
  merge_h, 118
  merge_h_range, 118
  merge_none, 119
  merge_v, 120

* flextable print function
  df_printer, 71
flextable_to_rmd, 79
gen_grob, 93
htmltools_value, 105
knit_print.flextable, 109
plot.flextable, 128
print.flextable, 131
save_as_docx, 136
save_as_html, 137
save_as_image, 138
save_as_pptx, 139
save_as_rtf, 140
to_html.flextable, 177

* functions for defining formatting properties
fp_border_default, 91
fp_text_default, 92

* functions for mixed content paragraphs
append_chunks, 17
as_paragraph, 42
compose, 67
prepend_chunks, 131

* functions for row and column operations in a flextable
add_body, 6
add_body_row, 7
add_footer, 9
add_footer_lines, 10
add_footer_row, 11
add_header, 12
add_header_row, 15
delete_columns, 69
delete_part, 70
delete_rows, 70
separate_header, 141
set_header_footer_df, 151
set_header_labels, 153

* functions related to themes
get_flextable_defaults, 96
set_flextable_defaults, 147
theme_alafoli, 168
theme_apa, 169
theme_booktabs, 170
tHEME_box, 171
theme_tron, 172
theme_tron_legacy, 173
theme_vader, 174
theme_vanilla, 175
theme_zebra, 176

* functions that add rows in the table
add_header_lines, 13

* sugar functions for table style
align, 16
bg, 49
bold, 52
color, 65
empty_blanks, 74
font, 87
fontsize, 88
highlight, 100
italic, 107
keep_with_next, 108
line_spacing, 115
padding, 124
rotate, 133
valign, 179

* text formatter functions
fmt_2stats, 80
fmt_avg_dev, 82
fmt_dbl, 83
fmt_header_n, 84
fmt_int, 84
fmt_n_percent, 85
fmt_pct, 86

* tools for clinical reporting
shift_table, 156

add_body, 6, 8–11, 13, 15, 69–71, 142, 152, 153
add_body_row, 7, 7, 9–11, 13, 15, 69–71, 142, 152, 153
add_footer, 7, 8, 9, 10, 11, 13, 15, 69–71, 142, 152, 153
add_footer_lines, 7–9, 10, 11, 13, 15, 69–71, 142, 152, 153
add_footer_lines(), 43
add_footer_row, 7–10, 11, 13, 15, 69–71, 142, 152, 153
add_header, 7–12, 15, 69–71, 142, 152, 153
add_header_lines, 13
add_header_lines(), 43
add_header_row, 7–11, 13, 15, 69–71, 142, 152, 153
add_latex_dep, 111
align_nottext_col (align), 16
align_text_col (align), 16
append_chunks, 17, 43, 67, 131
append_chunks(), 18–21, 40, 41, 43–45, 66, 67, 93, 97, 98, 106, 112, 113, 115, 121, 129
as_b, 18, 19–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_b(), 67
as_bracket, 18, 19, 20, 21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_chunk, 18, 19, 20, 21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_chunk(), 17, 43, 67, 93, 131
as_equation, 18–20, 21, 40–42, 44–46, 66, 97, 99, 114, 116, 122, 130
as_equation(), 94
as_flextable, 22, 23–33, 36, 38, 161, 164
as_flextable(), 6, 31, 35, 179
as_flextable.brmsfit
(as_flextable.merMod), 29
as_flextable.data.frame, 22, 22, 24–33, 36, 38
as_flextable.gam, 22, 23, 24, 25–33, 36, 38
as_flextable.glm, 22–24, 25, 26–33, 36, 38
as_flextable.glmmadmb
(as_flextable.merMod), 29
as_flextable.glmmTMB
(as_flextable.merMod), 29
as_flextable.gls (as_flextable.merMod), 29
as_flextable.grouped_data, 22–25, 27–33, 36, 38
as_flextable.grouped_data(), 39
as_flextable.htest, 22–26, 26, 28–33, 36, 38
as_flextable.kmeans, 22–27, 27, 28–33, 36, 38
as_flextable.lm, 22–28, 28, 29–33, 36, 38
as_flextable.lmer (as_flextable.merMod), 29
as_flextable.merMod, 22–28, 29, 30–33, 36, 38
as_flextable.nlme
(as_flextable.merMod), 29
as_flextable.pam, 22–29, 30, 31–33, 36, 38
as_flextable.summarizor, 22–30, 31, 32, 33, 36, 38
as_flextable.table, 22–31, 32, 33, 36, 38
as_flextable.tabular, 22–32, 32, 36, 38
as_flextable.tabular(), 125
as_flextable.tabulator, 22–33, 34, 38
as_flextable.tabulator(), 31, 165, 167
as_flextable.xtable, 22–33, 36, 37
as_grouped_data, 39
as_grouped_data(), 25, 26, 36, 165
as_highlight, 18–21, 40, 41, 42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_i, 18–21, 40, 40, 42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_image, 18–21, 40, 41, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
as_image(), 43, 67
as_paragraph, 17, 42, 67, 131
as_paragraph(), 7, 8, 10, 11, 14, 15, 20, 33, 41, 42, 44, 67, 89, 97, 98, 113–116, 121, 122, 129, 143, 164
as_raster, 72, 80, 95, 105, 111, 128, 132, 136–140, 178
as_sub, 18–21, 40–42, 43, 45, 46, 66, 97, 99, 106, 114, 116, 122, 130
as_sub(), 17
as_sup, 18–21, 40–42, 44, 44, 46, 66, 97, 99, 106, 114, 116, 122, 130
as_sup(), 17, 67
as_word_field, 18–21, 40–42, 44, 45, 45, 66, 97, 99, 106, 114, 116, 122, 130
as_word_field(), 67
autofit, 47, 72, 74, 75, 79, 100, 104, 123, 134, 155, 183
autofit(), 23, 78, 111, 127
before, 48
body_add_flextable, 50
body_add_flextable(). 144
body_replace_flextable_at_bkm, 51
border_inner, 53, 54–56, 102, 103, 163, 180–182
border_inner_h, 53, 53, 54–56, 102, 103, 163, 180–182
border_inner_h(), 162
border_inner_v, 53, 54, 54, 55, 56, 102, 103, 163, 180–182
border_inner_v(), 162
INDEX

border_outer, 53, 54, 55, 56, 102, 103, 163, 180–182
border_outer(), 162
border_remove, 53–55, 56, 102, 103, 163, 180–182

cluster::pam(), 30
colformat_char, 56, 58–64, 151
colformat_date, 57, 57, 59–64, 151
colformat_datetime, 57, 58, 58, 60–64, 151
colformat_double, 57–59, 59, 61–64, 151
colformat_double(), 63, 148, 151
colformat_image, 57–60, 60, 62–64, 151
colformat_int, 57–61, 61, 63, 64, 151
colformat_int(), 148
colformat_lgl, 57–62, 62, 64, 151
colformat_num, 57–63, 63, 151
colformat_num(), 6, 9, 13, 148
colorize, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
colorize(), 17
compose, 17, 43, 67, 131
compose(), 18–21, 40–45, 66, 78, 93, 97, 98, 106, 112–116, 121, 122, 129, 164
continuous_summary, 68
delete_columns, 7–11, 13, 15, 69, 70, 71, 142, 152, 153
delete_part, 7–11, 13, 15, 69, 70, 71, 142, 152, 153
delete_rows, 7–11, 13, 15, 69, 70, 71, 142, 152, 153
df_printer, 71, 80, 95, 105, 111, 128, 132, 136–140, 178
df_printer(), 178
dim, 95
dim.flextable, 48, 72, 74, 75, 79, 100, 104, 123, 155, 183
dim.flextableGrob, 73
dim_pretty, 48, 72, 73, 75, 79, 100, 104, 123, 134, 155, 183
dim_pretty(), 47, 127
div(), 105
fit_to_width, 48, 72, 74, 75, 79, 100, 104, 123, 155, 183
fix_border_issues, 76
flextable, 77
flextable(), 6–8, 11, 15, 146, 178, 179, 183
flextable-package, 5
flextable_dim, 48, 72, 74, 75, 79, 100, 104, 123, 155, 183
flextable_to_rmd, 72, 79, 95, 105, 111, 128, 132, 136–140, 178
fmt_2stats, 80, 82–86
fmt_avg_dev, 81, 82, 83–86
fmt_dbl, 81, 82, 83, 84–86
fmt_header_n, 81–83, 84, 85, 86
fmt_int, 81–84, 85, 86
fmt_n_percent, 81–85, 85, 86
fmt_pct, 81–85, 86
fmt_summarizor(fmt_2stats), 80
fmt_summarizor(), 161
font, 17, 50, 52, 65, 74, 87, 89, 101, 107, 108, 115, 124, 134, 180
footnote, 89
footnote(), 78
format, 62–64
formatC, 148
formatC(), 60, 64
fp_border, 35, 53–55, 91, 102, 103, 180–182
fp_border_default, 91, 93
fp_border_default(), 35
fp_par(), 33, 144
fp_text, 92
fp_text_default, 91, 92
fp_text_default(), 20, 21, 45, 67, 106
gdtools::register_gfont, 87
gen_grob, 72, 80, 93, 105, 111, 128, 132, 136–140, 178
gen_grob(), 128
get_flextable_defaults, 96, 149, 169–177
get_flextable_defaults(), 78, 147
gg_chunk, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
gg_chunk(), 17, 131
ggplot::ggplot(), 95
ggplot::ggplot(), 18–21, 40–42, 44–46, 66, 97, 98, 106, 114, 116, 122, 130
merge_h, 117, 118–120
merge_h(), 117, 118, 119, 120
merge_h, 117–119, 120
merge_h(), 13
minibar, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 121, 130
minibar(), 43
mk_par (compose), 67
mk_par(), 6, 81–86, 112, 164
ncol_keys, 48, 72, 74, 75, 79, 100, 104, 122, 123, 155, 183
ncol_keys(), 178
nrow_part, 48, 72, 74, 75, 79, 100, 104, 123, 155, 183
nrow_part(), 178
 officer::fp_text(), 20, 21, 45, 106
office::ph_location_type(), 127
office::read_pptx(), 127
office::run_autonum(), 143
office::styles_info(), 144
paginate, 125
paginate(), 33, 51, 108, 110, 111, 136, 140, 149
par(), 130
ph_with.flextable, 127
plot.flextable, 72, 80, 95, 105, 111, 128, 132, 136–140, 178
plot.flextableGrob, 129
plot_chunk, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
prepend_chunks, 17, 43, 67, 131
prepend_chunks(), 18–21, 40, 41, 43–45, 66, 67, 93, 97, 98, 106, 112, 113, 115, 121, 129
print.flextable, 72, 80, 95, 105, 111, 128, 131, 136–140, 178
proc_freq, 132
proc_freq(), 32
prop_section, 136, 140
qflextable (flextable), 77
rotate(), 135
rtf_add(), 135
rtf_add.flextable, 135

height, 48, 72, 74, 75, 79, 99, 104, 123, 155, 183
height(), 127, 133
highlight, 17, 50, 52, 65, 74, 88, 89, 100, 107, 108, 115, 124, 134, 180
hline, 53–56, 101, 103, 163, 180–182
hline(), 48, 91, 162
hline_bottom, 53–56, 102, 103, 163, 180–182
hline_top, 53–56, 102, 103, 163, 180–182
hline_top(), 162
hrule, 48, 72, 74, 75, 79, 100, 104, 123, 155, 183
hrule(), 99, 100, 133
HTML, 105
htmltools_value, 72, 80, 95, 105, 111, 128, 132, 136–140, 178
hyperlink_f_text(), 94
hyperlink_text, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 116, 122, 130
hyperlink_text(), 43
init_flextable_defaults
(set_flextable_defaults), 147
init_flextable_defaults(), 78
kmeans(), 27
knit_print.flextable, 72, 79, 80, 95, 105, 109, 128, 132, 136–140, 145, 178
knit_print.flextable(), 77, 78, 132
labelizor, 112
labelizor(), 67, 161
linerange, 18–21, 40–42, 44–46, 66, 97, 99, 106, 113, 116, 122, 130
lollipop, 18–21, 40–42, 44–46, 66, 97, 99, 106, 114, 115, 122, 130
merge_at, 117, 118–120
merge_h, 117, 118, 119, 120
merge_h(), 13
rtf_doc(). 135

save_as_docx, 72, 80, 95, 105, 111, 128, 132, 136, 137–140, 178
save_as_html, 72, 80, 95, 105, 111, 128, 132, 136, 137, 138–140, 178
save_as_image, 72, 80, 95, 105, 111, 128, 132, 136, 137, 138, 139, 140, 178
save_as_pptx, 72, 80, 95, 105, 111, 128, 132, 136–138, 139, 140, 178
save_as_rtf, 72, 80, 95, 105, 111, 128, 132, 136–139, 140, 178
separate_header, 7–11, 13, 15, 69–71, 141, 152, 153
set_caption, 142
set_caption(), 8, 11, 15, 78, 110, 139
set_flextable_defaults, 96, 147, 169–177
set_flextable_defaults(), 6, 77, 78, 91, 92, 169–177
set_footer_df(set_header_footer_df), 151
set_formatter, 57–64, 150
set_formatter(), 64
set_formatter_type(set_formatter), 150
set_header_df(set_header_footer_df), 151
set_header_footer_df, 7–11, 13, 15, 69–71, 142, 151, 153
set_header_labels, 7–11, 13, 15, 69–71, 142, 152, 153
set_table_properties, 48, 72, 74, 75, 79, 100, 104, 123, 154, 183
set_table_properties(), 47, 78, 109, 149, 183
shift_table, 156
sprintf(), 81
strptime(), 58, 59, 148
style, 159
style(), 78
summarizor, 161
summarizor(), 31, 36, 80, 81, 165
summary.tabulator(tabulator), 163
surround, 53–56, 102, 103, 162, 180–182
tables::tabular(), 32, 33
tabulator, 163
tabulator(), 35, 81–86, 125, 161, 166, 167
tabulator_colnames, 166
tabulator_colnames(), 165
theme_alafoli, 96, 149, 168, 170–177
theme_apa, 96, 149, 169, 171–177
theme_booktabs, 96, 149, 169, 170, 171–177
theme_booktabs(), 78
theme_box, 96, 149, 169–171, 171, 173–177
theme_tron, 96, 149, 169–172, 172, 174–177
theme_tron_legacy, 96, 149, 169–173, 173, 175–177
theme_vader, 96, 149, 169–174, 174, 176, 177
theme_vanilla, 96, 149, 169–175, 175, 177
theme_zebra, 96, 149, 169–176, 176
to_html.flextable, 72, 80, 95, 105, 111, 128, 132, 136–140, 177
use_df_printer, 178
use_df_printer(), 71, 179
use_model_printer, 179
vline, 53–56, 102, 103, 163, 180, 181, 182
vline(). 91, 162
vline_left, 53–56, 102, 103, 163, 180, 181, 182
vline_left(). 162
vline_right, 53–56, 102, 103, 163, 180, 181, 182
void, 182
width, 48, 72, 74, 75, 79, 100, 104, 123, 134, 155, 183
width(), 127