Package ‘flextable’

June 12, 2022

Type Package

Title Functions for Tabular Reporting

Version 0.7.2

Description Create pretty tables for 'HTML', 'PDF', 'Microsoft Word' and 'Microsoft PowerPoint' documents from 'R Markdown'. Functions are provided to let users create tables, modify and format their content. It also extends package 'officer' that does not contain any feature for customized tabular reporting.

License GPL-3

Imports stats, utils, grDevices, graphics, officer (>= 0.4.1), rmarkdown, knitr, htmltools, xml2, data.table (>= 1.13.0), uuid (>= 0.1-4), gdtools (>= 0.1.6), rlang, base64enc

RoxygenNote 7.2.0

Suggests testthat (>= 2.1.0), xtable, webshot, magick, ggplot2, scales, broom, broom.mixed, mgcv, cluster, lme4, nlme, bookdown, equatags, commonmark, pdf tools

Encoding UTF-8


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

VignetteBuilder knitr

NeedsCompilation no

Author David Gohel [aut, cre], Clementine Jager [ctb], 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)
Maintainer  David Gohel <david.gohel@ardata.fr>
Repository  CRAN
Date/Publication  2022-06-12 15:40:02 UTC

R topics documented:

- flextable-package ........................................... 5
- add_body ....................................................... 5
- add_body_row ............................................... 6
- add_footer .................................................... 7
- add_footer_lines ........................................... 8
- add_footer_row ............................................. 9
- add_header ................................................... 10
- add_header_lines .......................................... 12
- add_header_row ............................................ 13
- add_latex_dep .............................................. 14
- align .......................................................... 14
- append_chunks .............................................. 15
- as_b ............................................................ 16
- as_bracket .................................................... 17
- as_chunk .......................................................... 18
- as_equation .................................................. 19
- as_flextable ................................................ 20
- as_flextable.gam .......................................... 21
- as_flextable.glm .......................................... 22
- as_flextable.grouped_data ................................ 22
- as_flextable.htest .......................................... 24
- as_flextable.kmeans ....................................... 24
- as_flextable.lm ............................................. 25
- as_flextable.merMod ....................................... 26
- as_flextable.pam .......................................... 27
- as_flextable.tabulator .................................... 27
- as_flextable.xtable ....................................... 29
- as_grouped_data ........................................... 31
- as_highlight ............................................... 32
- as_i ............................................................. 33
- as_image ........................................................ 33
- as_paragraph ............................................... 35
- as_raster ..................................................... 36
- as_sub .......................................................... 37
- as_sup .......................................................... 38
- as_word_field .............................................. 39
- autofit ........................................................ 40
- before .......................................................... 42
- bg .............................................................. 43
- body_add_flextable ......................................... 44
- bold ............................................................ 46
<table>
<thead>
<tr>
<th>R topic documented:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_inner</td>
<td>46</td>
</tr>
<tr>
<td>border_inner_h</td>
<td>47</td>
</tr>
<tr>
<td>border_inner_v</td>
<td>48</td>
</tr>
<tr>
<td>border_outer</td>
<td>49</td>
</tr>
<tr>
<td>border_remove</td>
<td>50</td>
</tr>
<tr>
<td>colformat_char</td>
<td>50</td>
</tr>
<tr>
<td>colformat_date</td>
<td>51</td>
</tr>
<tr>
<td>colformat_datetime</td>
<td>52</td>
</tr>
<tr>
<td>colformat_double</td>
<td>53</td>
</tr>
<tr>
<td>colformat_image</td>
<td>55</td>
</tr>
<tr>
<td>colformat_int</td>
<td>56</td>
</tr>
<tr>
<td>colformat_lgl</td>
<td>57</td>
</tr>
<tr>
<td>colformat_num</td>
<td>58</td>
</tr>
<tr>
<td>color</td>
<td>59</td>
</tr>
<tr>
<td>colorize</td>
<td>61</td>
</tr>
<tr>
<td>compose</td>
<td>61</td>
</tr>
<tr>
<td>continuous_summary</td>
<td>63</td>
</tr>
<tr>
<td>delete_part</td>
<td>64</td>
</tr>
<tr>
<td>df_printer</td>
<td>64</td>
</tr>
<tr>
<td>dim.flextable</td>
<td>65</td>
</tr>
<tr>
<td>dim_pretty</td>
<td>66</td>
</tr>
<tr>
<td>empty_blanks</td>
<td>67</td>
</tr>
<tr>
<td>fit_to_width</td>
<td>68</td>
</tr>
<tr>
<td>fix_border_issues</td>
<td>69</td>
</tr>
<tr>
<td>flextable</td>
<td>69</td>
</tr>
<tr>
<td>flextable_dim</td>
<td>71</td>
</tr>
<tr>
<td>flextable_html_dependency</td>
<td>72</td>
</tr>
<tr>
<td>flextable_to_rmd</td>
<td>72</td>
</tr>
<tr>
<td>fmt_2stats</td>
<td>74</td>
</tr>
<tr>
<td>font</td>
<td>75</td>
</tr>
<tr>
<td>fontsize</td>
<td>76</td>
</tr>
<tr>
<td>footers_flextable_at_bkm</td>
<td>77</td>
</tr>
<tr>
<td>footnote</td>
<td>78</td>
</tr>
<tr>
<td>fp_border_default</td>
<td>79</td>
</tr>
<tr>
<td>fp_text_default</td>
<td>80</td>
</tr>
<tr>
<td>get_flextable_defaults</td>
<td>82</td>
</tr>
<tr>
<td>gg_chunk</td>
<td>83</td>
</tr>
<tr>
<td>headers_flextable_at_bkm</td>
<td>84</td>
</tr>
<tr>
<td>height</td>
<td>84</td>
</tr>
<tr>
<td>highlight</td>
<td>86</td>
</tr>
<tr>
<td>hline</td>
<td>87</td>
</tr>
<tr>
<td>hline_bottom</td>
<td>88</td>
</tr>
<tr>
<td>hline_top</td>
<td>89</td>
</tr>
<tr>
<td>hrule</td>
<td>90</td>
</tr>
<tr>
<td>htmltools_value</td>
<td>91</td>
</tr>
<tr>
<td>hyperlink_text</td>
<td>91</td>
</tr>
<tr>
<td>italic</td>
<td>92</td>
</tr>
<tr>
<td>knit_print.flextable</td>
<td>93</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------</td>
</tr>
<tr>
<td>linerange</td>
<td>97</td>
</tr>
<tr>
<td>line_spacing</td>
<td>98</td>
</tr>
<tr>
<td>lollipop</td>
<td>99</td>
</tr>
<tr>
<td>merge_at</td>
<td>101</td>
</tr>
<tr>
<td>merge_h</td>
<td>102</td>
</tr>
<tr>
<td>merge_h_range</td>
<td>102</td>
</tr>
<tr>
<td>merge_none</td>
<td>103</td>
</tr>
<tr>
<td>merge_v</td>
<td>104</td>
</tr>
<tr>
<td>minibar</td>
<td>105</td>
</tr>
<tr>
<td>ncol_keys</td>
<td>106</td>
</tr>
<tr>
<td>nrow_part</td>
<td>107</td>
</tr>
<tr>
<td>padding</td>
<td>108</td>
</tr>
<tr>
<td>ph_with.flextable</td>
<td>109</td>
</tr>
<tr>
<td>plot.flextable</td>
<td>110</td>
</tr>
<tr>
<td>plot_chunk</td>
<td>111</td>
</tr>
<tr>
<td>prepend_chunks</td>
<td>112</td>
</tr>
<tr>
<td>print.flextable</td>
<td>113</td>
</tr>
<tr>
<td>proc_freq</td>
<td>114</td>
</tr>
<tr>
<td>rotate</td>
<td>115</td>
</tr>
<tr>
<td>save_as_docx</td>
<td>116</td>
</tr>
<tr>
<td>save_as_html</td>
<td>117</td>
</tr>
<tr>
<td>save_as_image</td>
<td>118</td>
</tr>
<tr>
<td>save_as_pptx</td>
<td>120</td>
</tr>
<tr>
<td>separate_header</td>
<td>120</td>
</tr>
<tr>
<td>set_caption</td>
<td>122</td>
</tr>
<tr>
<td>set_flextable_defaults</td>
<td>123</td>
</tr>
<tr>
<td>set_formatter</td>
<td>126</td>
</tr>
<tr>
<td>set_header_footer_df</td>
<td>127</td>
</tr>
<tr>
<td>set_header_labels</td>
<td>128</td>
</tr>
<tr>
<td>set_table_properties</td>
<td>129</td>
</tr>
<tr>
<td>style</td>
<td>130</td>
</tr>
<tr>
<td>summarizor</td>
<td>132</td>
</tr>
<tr>
<td>surround</td>
<td>133</td>
</tr>
<tr>
<td>tabulator</td>
<td>135</td>
</tr>
<tr>
<td>tabulator_colnames</td>
<td>138</td>
</tr>
<tr>
<td>theme_alafoli</td>
<td>140</td>
</tr>
<tr>
<td>theme_booktabs</td>
<td>141</td>
</tr>
<tr>
<td>theme_box</td>
<td>142</td>
</tr>
<tr>
<td>theme_tron</td>
<td>143</td>
</tr>
<tr>
<td>theme_tron_legacy</td>
<td>144</td>
</tr>
<tr>
<td>theme_vader</td>
<td>145</td>
</tr>
<tr>
<td>theme_vanilla</td>
<td>146</td>
</tr>
<tr>
<td>theme_zebra</td>
<td>147</td>
</tr>
<tr>
<td>use_df_printer</td>
<td>148</td>
</tr>
<tr>
<td>use_model_printer</td>
<td>149</td>
</tr>
<tr>
<td>valign</td>
<td>150</td>
</tr>
<tr>
<td>vline</td>
<td>151</td>
</tr>
<tr>
<td>vline_left</td>
<td>152</td>
</tr>
</tbody>
</table>
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")`. To learn more about *flextable*, start with the vignettes: `browseVignettes(package = "flextable")`. The `flextable()` function is producing flexible tables where each cell can contain several chunks of text with their own set of formatting properties (bold, font color, etc.). Function `compose()` lets customise text of cells.

See Also

https://davidgohel.github.io/flextable/, `flextable()`

---

### 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).

See Also

flextable()

Other functions that add lines in the table: add_body_row(), add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), add_header()

Examples

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

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

ft <- theme_booktabs(ft)
ft

---

add_body_row Add body labels

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.

Usage

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 or a character() vector. If it is a list, it must be a named list using 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(), add_header_row()

Other functions that add lines in the table: add_body(), add_footer_lines(), add_footer_row(), add_footer(), add_header_row(), add_header()

Examples

ft <- flextable(head(iris))
ft <- add_body_row(ft, values = list(1000), colwidths = 5)
ft

---

add_footer

Add column values as new lines in footer

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.
Add labels as new rows in the footer

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.
add_footer_row

Usage

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

Arguments

    x        a flextable object
    values   a character vector, each element will be added as a new row.
    top      should the row be inserted at the top or the bottom. Default to TRUE.

Illustrations

See Also

Other functions that add lines in the table: add_body_row(), add_body(), add_footer_row(),
add_footer(), add_header_row(), add_header()

Other functions to add rows in header or footer: add_footer_row(), add_footer(), add_header_lines(),
add_header_row(), add_header(), separate_header(), set_header_footer_df, set_header_labels()

Examples

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

---

add_footer_row  Add footer labels

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.

Usage

    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 or a character() vector. If it is a list, it must be a named list using 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

Illustrations

See Also

Other functions that add lines in the table: add_body_row(), add_body(), add_footer_lines(), add_footer(), add_header_row(), add_header()

Other functions to add rows in header or footer: add_footer_lines(), add_footer(), add_header_lines(), add_header_row(), add_header(), separate_header(), set_header_footer_df, set_header_labels()

Examples

```r
ft_1 <- flextable(head(iris))
ft_1 <- add_footer_row(ft_1,
  values = "blah blah", colwidths = 5)
ft_1 <- add_footer_row(ft_1,
  values = c("blah", "blah"),
  colwidths = c(3, 2))
ft_1
```

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.
add_header

Usage

```r
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).

Illustrations

Note

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

See Also

Other functions that add lines in the table: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_row()`

Other functions to add rows in header or footer: `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_lines()`, `add_header_row()`, `separate_header()`, `set_header_footer_df`, `set_header_labels()`

Examples

```r
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))
```
add_header_lines

Add labels as new rows in the header

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, each element will be added as a new row.
top should the row be inserted at the top or the bottom. Default to TRUE.

Illustrations

See Also

Other functions to add rows in header or footer: add_footer_lines(), add_footer_row(), add_footer(),
add_header_row(), add_header(), separate_header(), set_header_footer_df, set_header_labels()

Examples

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
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.

**Usage**

```r
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) or a list.
- `colwidths`: the number of columns used for each label

**Illustrations**

**See Also**

Other functions that add lines in the table: `add_body_row()`, `add_body()`, `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header()`

Other functions to add rows in header or footer: `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_lines()`, `add_header()`, `separate_header()`, `set_header_footer_df`, `set_header_labels()`

**Examples**

```r
ft_1 <- flextable(head(iris))
ft_1 <- add_header_row(ft_1,
  values = "blah blah", colwidths = 5)
ft_1 <- add_header_row(ft_1,
  values = c("blah", "blah"),
  colwidths = c(3, 2))
ft_1
```
add_latex_dep

add latex dependencies

Description

Manually add flextable latex dependencies to the knitr session via `knit_meta_add()`.

When enabling caching in 'R Markdown' documents for PDF output, the flextable cached result is used directly. Call `add_latex_dep()` in a non cached chunk so that flextable latex dependencies are added to knitr metadata.

Usage

```r
add_latex_dep(float = FALSE, wrapfig = FALSE)
```

Arguments

- `float`: load package 'float'
- `wrapfig`: load package 'wrapfig'

Examples

```r
add_latex_dep()
```

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 = "left", part = "body")
align_text_col(x, align = "left", header = TRUE, footer = TRUE)
align_nottext_col(x, align = "right", header = TRUE, footer = TRUE)
```
append_chunks

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'.
- **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

Illustrations

See Also

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

Examples

```r
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
```

Description

append chunks to flextable content

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

Usage

```r
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')

Illustrations

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
```

---

**as_b**

*b*old chunk

---

Description

The function is producing a chunk with bold font.

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

Usage

```r
as_b(x)
```
Arguments

- `value`, if a chunk, the chunk will be updated

Illustrations

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()`, `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_b(Sepal.Length)
              ) )

ft
```

---

**as_bracket**  
*chunk with values in brackets*

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 `)`

Illustrations
See Also

Other chunk elements for paragraph: \texttt{as_b()}, \texttt{as_chunk()}, \texttt{as_equation()}, \texttt{as_highlight()}, \texttt{as_image()}, \texttt{as_i()}, \texttt{as_sub()}, \texttt{as_sup()}, \texttt{as_word_field()}, \texttt{colorize()}, \texttt{gg_chunk()}, \texttt{hyperlink_text()}, \texttt{linerange()}, \texttt{lollipop()}, \texttt{minibar()}, \texttt{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
```

Illustrations

See Also

Other chunk elements for paragraph: \texttt{as_bracket()}, \texttt{as_b()}, \texttt{as_equation()}, \texttt{as_highlight()}, \texttt{as_image()}, \texttt{as_i()}, \texttt{as_sub()}, \texttt{as_sup()}, \texttt{as_word_field()}, \texttt{colorize()}, \texttt{gg_chunk()}, \texttt{hyperlink_text()}, \texttt{linerange()}, \texttt{lollipop()}, \texttt{minibar()}, \texttt{plot_chunk()}

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 \texttt{compose()}, \texttt{append_chunks()} or \texttt{prepend_chunks()}. It should be used inside a call to \texttt{as_paragraph()}. 

Usage

\texttt{as_chunk(x, props = NULL, formatter = format_fun, ...)}

Arguments

\texttt{x} \hspace{1cm} text or any element that can be formatted as text with function provided in argument \texttt{formatter}.

\texttt{props} \hspace{1cm} an \texttt{officer::fp_text()} object to be used to format the text. If not specified, it will be the default value corresponding to the cell.

\texttt{formatter} \hspace{1cm} a function that will format \texttt{x} as a character vector.

\texttt{...} \hspace{1cm} additional arguments for \texttt{formatter} function.
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
```

<table>
<thead>
<tr>
<th>as_equation</th>
<th>equation chunk</th>
</tr>
</thead>
</table>

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")
```

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".

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()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`
Examples

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

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

---

as_flextable  
**method to convert object to flextable**

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

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

Arguments

- `x`  
  object to be transformed as flextable

- `...`  
  arguments for custom methods

See Also

Other as_flextable methods: `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.lm()`, `as_flextable.tabulator()`, `as_flextable.xtable()`
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

Illustrations

See Also

Other as_flextable methods: as_flextable.glm(), as_flextable.grouped_data(), as_flextable.htest(), as_flextable.lm(), 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

**Description**

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

**Usage**

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

**Arguments**

- `x`  
glm model
- `...`  
unused argument

**Illustrations**

**See Also**

Other as_flextable methods: `as_flextable.gam()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.lm()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `as_flextable()`

**Examples**

```r
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

**Description**

produce a flextable from a table produced by function `as_grouped_data()`.

**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`: object to be transformed as 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

Illustrations

See Also

- `as_grouped_data()`

Other as_flextable methods: `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.htest()`, `as_flextable.lm()`, `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  tabular summary for htest object

Description

produce a flextable describing an object of class htest.

Usage

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

Arguments

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

Illustrations

See Also

Other as_flextable methods: as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(), as_flextable.lm(), as_flextable.tabulator(), as_flextable.xtable(), as_flextable()

Examples

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

as_flextable.kmeans  tabular summary for kmeans

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_flextab(x, digits = 4, ...)
```

Arguments

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

Examples

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

---

**as_flextab.lm**  
*tabular summary for lm object*

Description

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

Usage

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

Arguments

- `x`: `lm` model
- `...`: unused argument

Illustrations

See Also

Other as_flextab methods: `as_flextable.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.tabulator()`, `as_flextable.xtable()`, `as_flextable()`
Examples

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

---

**as_flextable.merMod**  
*tabular summary for mixed model*

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.

Usage

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

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

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

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

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

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

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

Arguments

- `x`  
a mixed model
- `...`  
unused argument
Examples

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

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 `pam()` object
- `digits`: number of digits for the numeric columns
- `...`: unused argument

Examples

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

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",
  sep_w = 0.05,
  unit = "in",
  ...
)
```

Arguments

- `x` result from `tabulator()`
- `separate_with` columns used to separate the groups with a 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()`.
- `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.gam()`, `as_flextable.glm()`, `as_flextable.grouped_data()`, `as_flextable.htest()`, `as_flextable.lm()`, `as_flextable.xtable()`, `as_flextable()`

Examples

```r
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

set_flextable_defaults(padding = 1, font.size = 9, border.color = "orange")
ft_2 <- as_flextable(cft_1, sep_w = 0)
ft_2

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

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

init_flextable_defaults()

---

**as_flextable.xtable** get a flextable from a xtable object

### Description
Get a flextable object from a xtable object.

*xtable_to_flextable* will be deprecated in favor of *as_flextable.xtable*.

#### Usage

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

taxtable_to_flextable(
```
as_flextable.xtable

```
x,
  text.properties = fp_text_default(),
  format.args = gotOption("xtable.format.args", NULL),
  rowname_col = "rowname",
  hline.after = gotOption("xtable.hline.after", c(-1, 0, nrow(x))),
  NA.string = gotOption("xtable.NA.string", ""),
  include.rownames = TRUE,
  rotate.colnames = gotOption("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

Illustrations

See Also

Other as_flextable methods: as_flextable.gam(), as_flextable.glm(), as_flextable.grouped_data(),
  as_flextable.htest(), as_flextable.lm(), 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|l|r|
  ft_1 <- as_flextable(
    tli.table,
    rotate.colnames = TRUE,
```
as_grouped_data

grouped data transformation

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)

Arguments

x dataset

groups columns names to be used as row separators.

columns columns names to keep

See Also

as_flextable.grouped_data()
Examples

```r
# 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
```

---

### 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

```r
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()`, `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

### Illustrations

### 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()`, `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_i(Sepal.Length))
)

ft
```

---

**as_image**

---

**image chunk wrapper**

---

### Description

The function lets add images within flextable objects with function `compose()`. It should be used inside a call to `as_paragraph()`.
Usage

as_image(src, width = 0.5, height = 0.2, unit = "in", ...)

Arguments

src          image filename
width, height size of the png file in inches
unit         unit for width and height, one of "in", "cm", "mm".
...           unused argument

Illustrations

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_i(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), hyperlink_text(), linerange(), lollipop(), minibar(), plot_chunk()

Examples

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

myft <- flextable( head(iris))

myft <- compose( myft, i = 1:3, j = 1,
value = as_paragraph(
  as_image(src = img.file, width = .20, height = .15),
  " blah blah ",
  as_chunk(Sepal.Length, props = fp_text(color = "red"))
),
part = "body")

ft <- autofit(myft)
ft
as_paragraph

concatenate chunks in a flextable

Description

The function is concatenating text and images within paragraphs of a flextable object, this function is to be used with function `compose()`.

Usage

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

Arguments

- `...`: chunk elements that are defining paragraph
- `list_values`: a list of chunk elements that are defining paragraph. If specified argument `...` is unused.

Illustrations

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
```
**as_raster**

*get a flextable as a raster*

**Description**

save a flextable as an image and return the corresponding raster. This function has been implemented to let flextable be printed on a *ggplot* object.

**Usage**

```r
as_raster(x, zoom = 2, expand = 2, webshot = "webshot")
```

**Arguments**

- `x`: a flextable object
- `zoom`, `expand`: parameters used by `webshot` function.
- `webshot`: `webshot` package as a scalar character, one of "webshot" or "webshot2".

**Note**

This function requires packages: `webshot` and `magick`.

**See Also**

Other flextable print function: `df_printer()`, `flextable_to_rmd()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`

**Examples**

```r
ft <- qflextab( head( mtcars ) )
## Not run:
if( require("ggplot2") && require("webshot") ){  
  print(qplot(speed, dist, data = cars, geom = "point"))  
  grid::grid.raster(as_raster(ft))
}
## End(Not run)
```
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_sub(x)
```

Arguments

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

Illustrations

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()`, `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

Illustrations

Note

This is a sugar function that ease the composition of complex labels made of different formatings. 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()`, `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

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

Arguments

x
  computed field strings

props
  text properties (see fp_text_default() or officer::fp_text()) object to be used to format the text. If not specified, it will use the default text properties of the cell(s).

width, height
  size computed field

unit
  unit for width and height, one of "in", "cm", "mm".

See Also

Other chunk elements for paragraph: as_bracket(), as_b(), as_chunk(), as_equation(), as_highlight(), as_image(), as_i(), as_sub(), as_sup(), colorize(), gg_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.

**Illustrations**

**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
before(x, entries)

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

See Also
hline()

Examples
library(flextable)
library(officer)

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

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

*Set background color*

---

**Description**

Change 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`) 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**

```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.

**Illustrations**

**Note**

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

**See Also**

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

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

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

---

body_add_flextable  

**add flextable into a Word document**

Description

add a flextable into a Word document.

Usage

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

```r
body_replace_flextable_at_bkm(
x, bookmark, value, align = "center",
)```
Arguments

\begin{itemize}
  \item \textbf{x}\hspace{1cm}an \texttt{rdocx} object
  \item \textbf{value}\hspace{1cm}flextable object
  \item \textbf{align}\hspace{1cm}left, center (default) or right.
  \item \textbf{pos}\hspace{1cm}where to add the flextable relative to the cursor, one of "after", "before", "on" (end of line).
  \item \textbf{split}\hspace{1cm}set to \texttt{TRUE} if you want to activate Word option 'Allow row to break across pages'.
  \item \textbf{topcaption}\hspace{1cm}if \texttt{TRUE} caption is added before the table, if \texttt{FALSE}, caption is added after the table.
  \item \textbf{keepnext}\hspace{1cm}default \texttt{TRUE}. Word option 'keep rows together' is activated when \texttt{TRUE}. It avoids page break within tables. This is handy for small tables, i.e. less than a page height. Be careful, if you print long tables, you should rather set its value to \texttt{FALSE} to avoid that the tables also generate a page break before being placed in the Word document. Since Word will try to keep it with the next paragraphs that follow the tables.
  \item \textbf{bookmark}\hspace{1cm}bookmark id
\end{itemize}

\textbf{body_replace_flextable_at_bkm}

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.

Examples

\begin{verbatim}
library(officer)

# autonum for caption
autonum <- run_autonum(seq_id = "tab", bkm = "mtcars")

ftab <- flextable( head( mtcars ) )
ftab <- set_caption(ftab, caption = "mtcars data", autonum = autonum)
ftab <- autofit(ftab)
doc <- read_docx()
doc <- body_add_flextable(doc, value = ftab)
fileout <- tempfile(fileext = ".docx")
# fileout <- "test.docx" # uncomment to write in your working directory
print(doc, target = fileout)
\end{verbatim}
**bold**      *Set bold font*

---

**Description**

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

**Usage**

```r
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')

**Illustrations**

**See Also**

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

**Examples**

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

---

**border_inner**      *set vertical & horizontal inner borders*

---

**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")
```
**Description**

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

**Usage**

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

**Arguments**

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

**Illustrations**
See Also

Other borders management: `border_inner_v()`, `border_inner()`, `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 horizontal borders
ft <- border_inner_h(ft, border = std_border)
ft
```

Description

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

Usage

```r
border_inner_v(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')

Illustrations

See Also

Other borders management: `border_inner_h()`, `border_inner()`, `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_v(ft, border = std_border )
ft
```

---

**border_outer**

set outer borders

---

**Description**

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

**Usage**

```r
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')

**Illustrations**

**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**

```r
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

Illustrations

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

Description

Format date cells in a flextable.
Usage

```r
colformat_date(
  x,  # a flextable object
  i = NULL,  # rows selection
  j = NULL,  # columns selection
  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_date`: see `strptime()`
- `na_str`, `nan_str`: string to be used for NA and NaN values
- `prefix`, `suffix`: string to be used as prefix or suffix

See Also

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

Examples

```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

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

Arguments

x       a flextable object
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_date()`, `colformat_double()`, `colformat_image()`, `colformat_int()`, `colformat_lgl()`, `colformat_num()`, `set_formatter()`

Examples

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

---

**colformat_double**: format 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

Illustrations

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

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

Illustrations

See Also

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

Examples

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

dat <- head(iris)
dat$Species <- as.character(dat$Species)
dat[c(1, 3, 5), "Species"] <- img.file
myft <- flextable(dat)
myft <- colformat_image(
    myft, i = c(1, 3, 5),
    j = "Species", width = .20, height = .15)
ft <- autofit(myft)
ft

---

colformat_int  

format integer cells

Description
Format integer cells in a flextable.

Usage

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

Arguments

- **x**: a flextable object
- **i**: rows selection
- **j**: columns selection.
- **big.mark**: see `format()`
- **na_str, nan_str**: string to be used for NA and NaN values
- **prefix, suffix**: string to be used as prefix or suffix

See Also

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

Examples

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

Format logical cells in a flextable.

Usage

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

Arguments

- **x**: a flextable object
- **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(x) and not formatC(x). 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.

Illustrations

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.

Illustrations

See Also

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

Examples

```r
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
}
```
**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**

```r
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()`, `hyperlink_text()`, `linerange()`, `lollipop()`, `minibar()`, `plot_chunk()`

**Examples**

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

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

f
```

**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 preferred.
Usage

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

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

Arguments

x  a flextable object
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.

Illustrations

See Also

fp_text_default(), as_chunk(), as_b(), as_word_field()

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

Examples

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

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

ft_2 <- flextable(dat)
ft_2 <- mk_par(ft_2, j = ~ .-Species,
  value = as_paragraph(
    minibar(. , barcol = "white",
    height = .1)
### 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

#### Illustrations

#### Examples

```r
ft_1 <- continuous_summary(iris, names(iris)[1:4], by = "Species",
  hide_grouplabel = FALSE)
ft_1
```
**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`: part name of the table to delete (one of 'body', 'header' or 'footer').

**Illustrations**

**Examples**

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

---

**df_printer**

*data.frame automatic printing as a flextable*

**Description**

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

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

```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()
```
dim.flextable

Usage

df_printer(dat, ...)

Arguments

dat the data.frame

... unused argument

Details

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

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

See Also

Other flextable print function: as_raster(), flextable_to_rmd(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx()

Examples

df_printer(head(mtcars))

dim.flextable Get widths and heights of flextable

description

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

Usage

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

x 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()

Examples

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

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

Arguments

x 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(), flextable_dim(), height(), hrule(), ncol_keys(), nrow_part(), set_table_properties(), width()

Examples

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

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(), line_spacing(), padding(), rotate(), valign()

Examples

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

tab <- flextable(head(iris), col_keys = c(  
  "Species",  
  "break2", "Petal.Length", "Petal.Width"
))
tab <- set_header_df(tab, 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

fit a flextable to a maximum width

Description

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

Usage

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

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".

Illustrations

See Also

Other flextable dimensions: `autofit()`, `dim.flextable()`, `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

tf_2 <- fit_to_width(ft_1, max_width = 4)
ft_2
```
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
fix_border_issues(x, part = "all")

Arguments
x flextable object
part partname of the table (one of ‘all’, ‘body’, ‘header’, ‘footer’)

Examples
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
)
```

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

new lines and tabulations

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

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

Tabs are different depending on the output format:

- HTML is using entity `\em space`
- Word - a Word 'tab' element
- PowerPoint - a PowerPoint 'tab' element
- latex - tag "\quad"
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
ft <- flextable(head(mtcars))
ft

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

**htmlDependency for flextable objects**

**Description**

When using loops in an R Markdown for HTML document, the htmlDependency object for flextable must also be added at least once.

**Usage**

```r
flextable_html_dependency(htmlscroll = TRUE)
```

**Arguments**

- `htmlscroll` add a scroll if table is too big to fit into its HTML container, default to TRUE.

**Examples**

```r
if(require("htmltools"))
  div(flextable_html_dependency())
```

---

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'.

All arguments whose name starts with `ft.` can be set in the chunk options.

See `knit_print.flextable` for more details.

**Usage**

```r
flextable_to_rmd(
  x,
  ft.align = opts_current$get("ft.align"),
  ft.split = opts_current$get("ft.split"),
  ft.keepnext = opts_current$get("ft.keepnext"),
  ft.tabcolsep = opts_current$get("ft.tabcolsep"),
  ft.arraystretch = opts_current$get("ft.arraystretch"),
  ft.latex.float = opts_current$get("ft.latex.float"),
  ft.left = opts_current$get("ft.left"),
  ft.top = opts_current$get("ft.top"),
```
text_after = "",
webshot = opts_current$get("webshot"),
bookdown = FALSE,
pandoc2 = TRUE,
print = TRUE,
...
)

Arguments

x

a flextable object

ft.align flextable alignment, supported values are 'left', 'center' and 'right'.

ft.split Word option 'Allow row to break across pages' can be activated when TRUE.

ft.keepnext default TRUE. 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.

Be careful, if you print long tables, you should rather set its value to FALSE to avoid that the tables also generate a page break before being placed in the Word document. Since Word will try to keep it with the next paragraphs that follow the tables.

ft.tabcolsep space between the text and the left/right border of its containing cell, the default value is 8 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

ft.left, ft.top Position should be defined with options ft.left and ft.top. Theses are the top left coordinates in inches of the placeholder that will contain the table. Their default values are 1 and 2 inches.

text_after The string you put here will be added after printing the content of the flextable. For example, you can put "$pagebreak" here to have tables produced with page breaks.

webshot webshot package as a scalar character, one of "webshot" or "webshot2".

bookdown TRUE or FALSE (default) to support cross referencing with bookdown.

pandoc2 TRUE (default) or FALSE to get the string in a pandoc raw HTML attribute (only valid when pandoc version is >= 2.

print print output if TRUE

... unused arguments
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 summaries (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(
  num1,
  num2,
  cts,
  pcts,
  num1_mask = "%.01f",
  num2_mask = "(% .01f)",
  cts_mask = "%.0f",
  pcts_mask = "(% .02f %)"
)
font

Arguments

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.
cnts  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().
cnts_mask  format associated with cnts, a format string used by sprintf().
pcts_mask  format associated with pcts, a format string used by sprintf().

See Also

summarizor(), tabulator(), mk_par()

Description

change font of selected rows and columns of a flextable.

Usage

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

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

Usage

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

x a flextable object
i rows selection
j columns selection
size integer value (points)
part partname of the table (one of 'all', 'body', 'header', 'footer')

Illustrations

See Also

Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), font(), highlight(), italic(), 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

Description

replace in the footer of a document a paragraph containing a bookmark by a flextable. A bookmark will be considered as valid if enclosing words within a paragraph; i.e., a bookmark along two or more paragraphs is invalid, a bookmark set on a whole paragraph is also invalid, but bookmarking few words inside a paragraph is valid.

Usage

footers_flextable_at_bkm(x, bookmark, value)

Arguments

x an rdocx object
bookmark bookmark id
value a flextable object
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

```r
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.

Illustrations

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")
)
```
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 = 1
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>border color - single character value (e.g. &quot;#000000&quot; or &quot;black&quot;)</td>
</tr>
<tr>
<td>style</td>
<td>border style - single character value: &quot;none&quot; or &quot;solid&quot; or &quot;dotted&quot; or &quot;dashed&quot;</td>
</tr>
<tr>
<td>width</td>
<td>border width - an integer value: ( 0 \leq \text{value} )</td>
</tr>
</tbody>
</table>

See Also

- hline()
- vline()

Other functions for defining formatting properties: fp_text_default()

Examples

```r
library(flextable)

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()
```

Description

Create a \texttt{fp\_text()} object that uses default values defined in the flextable it applies to.

\texttt{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 \texttt{fp\_text\_default(\texttt{bold = TRUE})}, the formatting will be 'bold' but it will also be 'red'.

On the other hand, the \texttt{fp\_text()} function forces you to specify all the parameters, so we strongly recommend working with \texttt{fp\_text\_default()} which was created to replace the use of the former.

See also \texttt{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()
```

---

**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**

```r
get_flextable_defaults()
```

**Value**

A list containing default values.

**See Also**

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

**Examples**

```r
get_flextable_defaults()
```
This function is used to insert mini gg plots into flextable with function `compose()`. It should be used inside a call to `as_paragraph()`.

**Usage**

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

**Arguments**

- `value`: gg objects, stored in a list column.
- `width`, `height`: size of the resulting png file in inches
- `unit`: unit for width and height, one of "in", "cm", "mm".

**Illustrations**

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()`, `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")
    ggplot(data = cormat, aes(x=var1, y=var2, fill=correlation)) +
    geom_tile() + coord_equal() +
  }
  ```
scale_fill_gradient2(low = "blue",
                        mid = "white", high = "red", limits = c(-1, 1),
                        guide = FALSE) + theme_void()
}

z <- as.data.table(iris)
z <- z[, list(gg = list(my_cor_plot(.SD))), by = "Species"]
ft <- flextable(z)
ft <- mk_par(ft, j = "gg",
             value = as_paragraph(
                gg_chunk(value = gg, width = 1, height = 1)
             ))
ft

headers_flextable_at_bkm

add flextable at a bookmark location in document’s header

Description

replace in the header of a document a paragraph containing a bookmark by a flextable. A bookmark will be considered as valid if enclosing words within a paragraph; i.e., a bookmark along two or more paragraphs is invalid, a bookmark set on a whole paragraph is also invalid, but bookmarking few words inside a paragraph is valid.

Usage

headers_flextable_at_bkm(x, bookmark, value)

Arguments

- **x**: an rdocx object
- **bookmark**: bookmark id
- **value**: a flextable object

height

Set flextable rows height

Description

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

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

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

Arguments

x: flextable object
i: rows selection
height: height in inches
part: partname of the table
unit: unit for height, one of "in", "cm", "mm".

Illustrations

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

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

Text highlight color

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) 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
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.

Illustrations

See Also
Other sugar functions for table style: align(), bg(), bold(), color(), empty_blanks(), fontsize(), font(), italic(), 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
}

**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**

```r
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')

**Illustrations**

**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')

**Illustrations**

**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)
bigr = 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 = bigr )
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')

**Illustrations**

**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)

# add horizontal border on top
ft <- hline_top(ft, part="all", border = big_border )
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.

Usage

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"

Illustrations

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

flextable as an HTML object

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 = "center", ft.shadow = TRUE, ft.htmlscroll = TRUE)

Arguments

x
a flextable object

ft.align
flextable alignment, supported values are 'left', 'center' and 'right'.

ft.shadow
use shadow dom, this option is existing to disable shadow dom (set to FALSE) for pagedown that can not support it for now.

ft.htmlscroll
add a scroll if table is too big to fit into its HTML container, default to TRUE.

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(), knit_print.flextable(), plot.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx()

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

---

**italic**

*Set italic font*

**Description**

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

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

Arguments

x
  a flextable object
i
  rows selection
j
  columns selection
italic
  boolean value
part
  partname of the table (one of 'all', 'body', 'header', 'footer')

Illustrations

See Also

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

Examples

ft <- flextable(head(mtcars))
ft <- italic(ft, italic = TRUE, part = "header")

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
- `...`: arguments passed to `flextable_to_rmd()`.

Chunk options

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

<table>
<thead>
<tr>
<th>chunk option</th>
<th>property</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ft.align</td>
<td>flextable alignment, supported values are 'left', 'center' and 'right'</td>
<td>'center'</td>
</tr>
<tr>
<td>ft.shadow</td>
<td>HTML option, disable shadow dom (set to FALSE) for pagedown.</td>
<td>TRUE</td>
</tr>
<tr>
<td>ft.htmlscroll</td>
<td>HTML option, add a scroll if table is too big to fit into its HTML container.</td>
<td>TRUE</td>
</tr>
<tr>
<td>ft.split</td>
<td>Word option 'Allow row to break across pages' can be activated when TRUE.</td>
<td>FALSE</td>
</tr>
<tr>
<td>ft.keepnext</td>
<td>Word option 'keep rows together' can be deactivated when FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>ft.tabcolsep</td>
<td>space between the text and the left/right border of its containing cell</td>
<td>8.0</td>
</tr>
<tr>
<td>ft.arraystretch</td>
<td>height of each row relative to its default height</td>
<td>1.5</td>
</tr>
<tr>
<td>ft.latex.float</td>
<td>type of floating placement in the document, one of 'none', 'float', 'wrap-r', 'wrap-l', 'wrap-i', 'wrap-o'</td>
<td>none</td>
</tr>
<tr>
<td>ft.left</td>
<td>left coordinates in inches</td>
<td>1.0</td>
</tr>
<tr>
<td>ft.top</td>
<td>top coordinates in inches</td>
<td>2.0</td>
</tr>
</tbody>
</table>

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, ft.keepnext = FALSE, ...)`. See `flextable_to_rmd()` for more details about these options.

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>tab.cap.style</td>
<td>NULL</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>label</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix for numbering chunk (default to &quot;Table &quot;).</td>
<td>tab.cap.pre</td>
<td>Table</td>
</tr>
<tr>
<td>suffix for numbering chunk (default to &quot;: &quot;).</td>
<td>tab.cap.sep</td>
<td>&quot;:&quot;</td>
</tr>
<tr>
<td>title number depth</td>
<td>tab.cap.tnd</td>
<td>0</td>
</tr>
<tr>
<td>caption prefix formatting properties</td>
<td>tab.cap.fp_text</td>
<td>fp_text_lite(bold = TRUE)</td>
</tr>
<tr>
<td>separator to use between title number and table number.</td>
<td>tab.cap.tns</td>
<td>&quot;.-&quot;</td>
</tr>
</tbody>
</table>

**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. Some output may not support this feature. To our knowledge, only the pagedown output is concerned. Use knitr chunk option `ft.shadow = FALSE` to disable shadow dom.

If `ft.shadow = TRUE` some global CSS rules may change the desired output of flextables.

**PDF output**

Some features are not implemented in PDF due to technical infeasibility. These are the padding, line_spacing and height properties.

It is recommended to set theses 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")
```

Background color and merged cells does not work well together with PDF format. Authors are hoping to fix this issue in the future.

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>
</table>
See Also

Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), htmltools_value(), plot.flextable(), print.flextable(), save_as_docx(), save_as_html(), save_as_image(), save_as_pptx()

Examples

# simple examples -----

demo_docx <- system.file(package = "flextable", "examples/rmd", "demo.Rmd")
rmd_file <- tempfile(fileext = ".Rmd")
file.copy(demo_docx, to = rmd_file, overwrite = TRUE)
rmd_file # R Markdown document used for demo

if(require("rmarkdown", quietly = TRUE)){
  # knitr::opts_chunk$set(webshot = "webshot2")
  # render(input = rmd_file, output_format = "word_document", output_file = "doc.docx")
  # render(input = rmd_file, output_format = "pdf_document", output_file = "doc.pdf")
  # render(input = rmd_file, output_format = "html_document", output_file = "doc.html")
  # render(input = rmd_file, output_format = "powerpoint_presentation", output_file = "pres.pptx")
  # render(input = rmd_file, output_format = "slidy_presentation", output_file = "slidy.html")
  # render(input = rmd_file, output_format = "beamer_presentation", output_file = "beamer.pdf")
  # render(input = rmd_file, output_format = "pagedown::html_paged", output_file = "paged.html")
}

## bookdown examples with captions and cross ref -----

captions_example <- system.file(
  package = "flextable",
  "examples/rmd", "captions_example.Rmd")

dir_tmp <- tempfile(pattern = "dir")

dir.create(dir_tmp, showWarnings = FALSE, recursive = TRUE)

file.copy(captions_example, dir_tmp)

rmd_file <- file.path(dir_tmp, basename(captions_example))

if(require("rmarkdown", quietly = TRUE)){
  # render(input = rmd_file, output_format = word_document(), output_file = "doc.docx")
  # render(input = rmd_file, output_format = pdf_document(latex_engine = "xelatex"), output_file = "doc.pdf")
  # render(input = rmd_file, output_format = html_document(),)

linerange

mini linerange chunk wrapper

Description

This function is used to insert lineranges into flextable with function `compose()`. It should be used inside a call to `as_paragraph()`

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 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(), 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_spacing**

*Set text alignment*

Description

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

line_spacing(x, i = NULL, j = NULL, space = 1, part = "body", unit = "in")

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')
unit  
unit for space, one of "in", "cm", "mm".

Illustrations

See Also

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

Examples

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

lollipop


mini lollipop chart chunk wrapper

Description

This function is used to insert lollipop charts into flextable with function compose(). It should be used inside a call to as_paragraph().

Usage

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.

Illustrations

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_1(), as_sub(), as_sup(), as_word_field(), colorize(), gg_chunk(), hyperlink_text(), linerange(), minibar(), plot_chunk()
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

Merge flextable cells horizontally

Description

Merge flextable cells horizontally when consecutive cells have identical values. Text of formatted values are used to compare values.

Usage

merge_h(x, i = NULL, part = "body")

Arguments

x 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 )
ft_merge <- flextable(dummy_df)
ft_merge <- merge_h(x = ft_merge)
ft_merge

merge_h_range

rowwise merge of a range of columns

Description

Merge flextable columns into a single one for each selected rows. All columns must be consecutive.

Usage

merge_h_range(x, i = NULL, j1 = NULL, j2 = NULL, part = "body")

Arguments

x 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.
merge_none

Illustrations

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.

Illustrations

See Also

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

Examples

typology <- data.frame(
  measure = c("Length", "Width", "Length", "Width", "Species"),
  stringsAsFactors = FALSE )

ft <- flextable( head( iris ) )
```
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 orignal 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.

Illustrations

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", "172525"),
```
```
minibar

mini barplots chunk wrapper

Description

This function is used to insert bars into flextable with function `compose()`. It should be used inside a call to `as_paragraph()`.

Usage

```r
minibar(
  value,
  max = NULL,
  barcol = "#CCCCCC",
  bg = "transparent",
  width = 1,
  height = 0.2,
  unit = "in"
)
```

Arguments

- `value`: values containing the bar size
ncol_keys

<table>
<thead>
<tr>
<th>max</th>
<th>max bar size</th>
</tr>
</thead>
<tbody>
<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>

Illustrations

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(), 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
```

---

ncol_keys  Number of columns

Description

returns the number of columns displayed

Usage

ncol_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

```r
library(flextable)
ft <- qflextable(head(cars))  
nrow_part(ft, part = "body")
```

<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

```r
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')

Illustrations

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

```r
ft_1 <- flextable(head(iris))
ft_1 <- theme_vader(ft_1)
ft_1 <- padding(ft_1, padding.top = 4, part = "all")
ft_1 <- padding(ft_1, j = 1, padding.right = 40)
ft_1 <- padding(ft_1, i = 3, padding.top = 40)
ft_1 <- padding(ft_1, padding.top = 10, part = "header")
ft_1 <- padding(ft_1, padding.bottom = 10, part = "header")
ft_1 <- autofit(ft_1)
ft_1
```

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

save a flextable as an image and display the result in a new R graphics window.

Usage

```r
## S3 method for class 'flextable'
plot(x, zoom = 2, expand = 2, ...)
```

Arguments

- `x` a flextable object
- `zoom`, `expand` parameters used by `webshot` function.
- `...` additional parameters sent to `as_raster()` function

Note

This function requires packages: webshot and magick.

See Also

Other flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `htmltools_value()`, `knit_print.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`

Examples

```r
ftab <- flextable( head( mtcars ) )
ftab <- autofit(ftab)
## Not run:
if( require("webshot") ){
  plot(ftab)
}
## End(Not run)
```
Description

This function is used to insert mini plots into flextable with function `compose()`. It should be used inside a call to `as_paragraph()`.

Available plots are 'box', 'line', 'points', 'density'.

Usage

```r
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()`)

Illustrations

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_l()`, `as_sub()`, `as_sup()`, `as_word_field()`, `colorize()`, `gg_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
`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("coucou "), color = "red")
)
x
```

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", ...)
```

Arguments

- `x`: flextable object
- `preview`: preview type, one of c("html", "pptx", "docx", "pdf", "log"). When "log" is used, a description of the flextable is printed.
- `...`: 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()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`
**proc_freq**

frequency table as flextable

**Description**

This function computes a two-way contingency table and makes a flextable with the result.

**Usage**

```r
proc_freq(
  x, row, col,
  main = "", include.row_percent = TRUE,
  include.column_percent = TRUE,
  include.table_percent = TRUE,
  include.column_total = TRUE,
  include.row_total = TRUE,
  include.header_row = TRUE,
  weight = NULL
)
```

**Arguments**

- `x` data.frame object
- `row` character column names for row
- `col` character column names for column
- `main` character title
- `include.row_percent` boolean whether to include the row percents; defaults to TRUE
- `include.column_percent` boolean whether to include the column percents; defaults to TRUE
- `include.table_percent` boolean whether to include the table percents; defaults to TRUE
- `include.column_total` boolean whether to include the row of column totals; defaults to TRUE
- `include.row_total` boolean whether to include the column of row totals; defaults to TRUE
- `include.header_row` boolean whether to include the header row; defaults to TRUE
- `weight` character column name for weight

**Author(s)**

Titouan Robert
Examples

```r
proc_freq(mtcars, "vs", "gear")
proc_freq(mtcars, "gear", "vs")
proc_freq(mtcars, "gear", "vs", weight = "wt")
proc_freq(mtcars, "gear", "vs", "My title")
```

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 "lrtb" 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.

Note that PDF does not yet support vertical alignments when text is rotated.

Usage

```r
rotate(x, i = NULL, j = NULL, rotation, align = NULL, part = "body")
```

Arguments

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

Details

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

Illustrations
See Also

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

Examples

```r
library(flextable)

ft_1 <- flextable(head(iris))
ft_1 <- rotate(ft_1, j = 1:4, align = "bottom", rotation = "tbrl", part = "header")
ft_1 <- rotate(ft_1, j = 5, align = "bottom", rotation = "btlr", part = "header")

# if output is docx or pptx, think about (1) set header heights  
# and (2) set rule "exact" for rows heights because Word  
# and PowerPoint don't handle auto height with rotated headers
ft_1 <- height(ft_1, height = 1.2, part = "header")
ft_1 <- hrule(ft_1, i = 1, rule = "exact", part = "header")

ft_1

dat <- data.frame(
  a = c("left-top", "left-middle", "left-bottom"),
  b = c("center-top", "center-middle", "center-bottom"),
  c = c("right-top", "right-middle", "right-bottom")
)

ft_2 <- flextable(dat)
ft_2 <- theme_box(ft_2)
ft_2 <- height_all(x = ft_2, height = 1.3, part = "body")
ft_2 <- hrule(ft_2, rule = "exact")
ft_2 <- rotate(ft_2, rotation = "tbrl")
ft_2 <- width(ft_2, width = 1.3)
ft_2 <- align(ft_2, j = 1, align = "left")
ft_2 <- align(ft_2, j = 2, align = "center")
ft_2 <- align(ft_2, j = 3, align = "right")
ft_2 <- valign(ft_2, i = 1, valign = "top")
ft_2 <- valign(ft_2, i = 2, valign = "center")
ft_2 <- valign(ft_2, i = 3, valign = "bottom")

ft_2
```

---

**save_as_docx**  
**save flextable objects in an Word file**

**Description**

sugar function to save flextable objects in an Word file.
save_as_html

Usage

```r
save_as_docx(..., 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.

See Also

Other flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_html()`, `save_as_image()`, `save_as_pptx()`

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",
    width = 8.3, height = 11.7),
  type = "continuous",
  page_margins = page_mar()
)
save_as_docx("iris table" = ft1, "mtcars table" = ft2,
  path = tf, pr_section = sect_properties)
```

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).
save_as_image

Usage

save_as_html(
  ..., 
  values = NULL, 
  path, 
  encoding = "utf-8", 
  title = deparse(sys.call())
)

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
encoding encoding to be used in the HTML file
title page title

See Also

Other flextable print function: as_raster(), df_printer(), flextable_to_rmd(), htmltools_value(), knit_print.flextable(), plot.flextable(), print.flextable(), save_as_docx(), save_as_image(), save_as_pptx()

Examples

ft1 <- flextable( head( iris ) )
tf1 <- tempfile(fileext = ".html")
save_as_html(ft1, path = tf1)
# browseURL(tf1)

ft2 <- flextable( head( mtcars ) )
tf2 <- tempfile(fileext = ".html")
save_as_html( 
  `iris table` = ft1, 
  `mtcars table` = ft2, 
  path = tf2, 
  title = "rhoooo")
# browseURL(tf2)

save_as_image save a flextable as an image
Description

save a flextable as a png, pdf or jpeg image.

Image generated with package ‘webshot’ or package ‘webshot2’. **Package ‘webshot2’ should be preferred** as ‘webshot’ can have issues with some properties (i.e. bold are not rendered for some users).

The image is coming from a screenshot of the ‘HTML’ output. save_as_image() is a tool to make life easier for users. Nevertheless, the features have some limitations that can’t be solved with flextable because they are not related to flextable:

- png does support transparency,
- jpeg does not support transparency,
- webshot2 does not allow transparent background,
- webshot does allow transparent background.

Usage

```r
save_as_image(x, path, zoom = 3, expand = 10, webshot = "webshot")
```

Arguments

- `x`: a flextable object
- `path`: image file to be created. It should end with .png, .pdf, or .jpeg.
- `zoom, expand`: parameters used by webshot function.
- `webshot`: webshot package as a scalar character, one of "webshot" or "webshot2".

Note

This function requires package webshot or webshot2. The screenshot process is rather slow because it is managed by an external program (see webshot or webshot2 documentation).

See Also

Other flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_pptx()`

Examples

```r
ft <- flextable( head( mtcars ) )
ft <- autofit(ft)
tf <- tempfile(fileext = ".png")
## Not run:
if( require("webshot") ){
  save_as_image(x = ft, path = "myimage.png")
}
## End(Not run)
```
**save_as_pptx**

save flextable objects in an PowerPoint file

---

**Description**

sugar function to save flextable objects in an PowerPoint file.

**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

**See Also**

Other flextable print function: `as_raster()`, `df_printer()`, `flextable_to_rmd()`, `htmltools_value()`, `knit_print.flextable()`, `plot.flextable()`, `print.flextable()`, `save_as_docx()`, `save_as_html()`, `save_as_image()`

**Examples**

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

ft2 <- flextable( head( mtcars ) )
tf <- tempfile(fileext = ".pptx")
save_as_pptx('iris table' = ft1, 'mtcars table' = ft2, path = tf)
```

---

**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
- **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.

Illustrations

See Also

Other functions to add rows in header or footer: `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_lines()`, `add_header_row()`, `add_header()`, `set_header_footer_df`, `set_header_labels()`

Examples

```r
library(dplyr)

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(0.18539, 0.19775, 0.27465)
)
```
```r
ft_1 <- flextable(x)
ft_1 <- colformat_double(ft_1, digits = 2)
ft_1 <- theme_box(ft_1)
ft_1 <- separate_header(
  x = ft_1,
  opts = c("span-top", "bottom-vspan")
)
ft_1
```

---

**set_caption**

**Set Caption**

**Description**

Set caption value in a flextable.

- The caption will be associated with a paragraph style when the output is Word. It can also be numbered as an auto-numbered Word computed value.
- The PowerPoint format ignores captions.

**Usage**

```r
set_caption(
  x,
  caption,
  autonum = NULL,
  style = "Table Caption",
  html_escape = TRUE
)
```

**Arguments**

- `x`: flextable object
- `caption`: caption value
- `autonum`: an autonum representation. See `officer::run_autonum()`. This has only an effect when output is Word. If used, the caption is preceded by an auto-number sequence. In this case, the caption is preceded by an auto-number sequence that can be cross referenced.
- `style`: caption paragraph style name. These names are available with function `officer::styles_info()` when output is Word; if HTML, the value is set as class value in the caption tag.
- `html_escape`: should HTML entities be escaped so that it can be safely included as text or an attribute value within an HTML document.

**R Markdown**

Flextable captions can be defined from R Markdown documents by using `knitr::opts_chunk$set()`. The following options are available with `officedown::rdocx_document` and/or `bookdown`:
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.

<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>
<tr>
<td>caption table sequence identifier.</td>
<td>tab.lp</td>
<td>&quot;tab:&quot;</td>
</tr>
</tbody>
</table>

The following options are only available when used with `officedown::rdocx_document`:

<table>
<thead>
<tr>
<th>label</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix for numbering chunk (default to &quot;Table &quot;).</td>
<td>tab.cap.pre</td>
<td>Table</td>
</tr>
<tr>
<td>suffix for numbering chunk (default to &quot;: &quot;).</td>
<td>tab.cap.sep</td>
<td>&quot;:&quot;</td>
</tr>
<tr>
<td>title number depth</td>
<td>tab.cap.tnd</td>
<td>0</td>
</tr>
<tr>
<td>separator to use between title number and table number.</td>
<td>tab.cap.tns</td>
<td>&quot;:&quot;</td>
</tr>
<tr>
<td>caption prefix formatting properties</td>
<td>tab.cap.fp_text</td>
<td>fp_text_lite(bold = TRUE)</td>
</tr>
</tbody>
</table>

See `knit_print.flextable` for more details.

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
```


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,
  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,
  na_str = NULL,
  nan_str = NULL,
  fmt_date = NULL,
  fmt_datetime = NULL,
  extra_css = NULL,
  fonts_ignore = NULL,
  theme_fun = 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)
`set_flextable_defaults` 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").

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()`.

fmt_date, fmt_datetime formats for date and datetime columns as documented in `strptime()`. Default to '%Y-%m-%d' and '%Y-%m-%d %H:%M:%S'.

extra_css css instructions to be integrated with the table.

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_pdf, post_process_docx, post_process_html, post_process_pptx Post-processing functions that will allow you to customize the display by output type (pdf, html, docx, pptx). They are executed just before printing the table.

**Value**

a list containing previous default values.

**Illustrations**

**See Also**

Other functions related to themes: `get_flextable_defaults()`, `theme_alafoli()`, `theme_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`, `theme_zebra()`
Examples

```r
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**

Define formatter functions associated to each column key. Functions have a single argument (the vector) and are returning the formatted values as a character vector.

**Usage**

```r
set_formatter(x, ..., values = NULL, part = "body")
```

```r
set_formatter_type(
  x,
  fmt_double = "%03f",
  fmt_integer = "%0f",
  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**
  - a list of name-value pairs of functions, names should be existing col_key values. If values is supplied argument ... is ignored.
- **part**
  - partname of the table (one of 'body' or 'header' or 'footer')
- **fmt_double, fmt_integer**
  - arguments used by `sprintf` to format double and integer columns.
- **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()

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.
set_header_labels

Illustrations

See Also

Other functions to add rows in header or footer: `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_lines()`, `add_header_row()`, `add_header()`, `separate_header()`, `set_header_labels()`

Examples

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

ft_1 <- flextable(head(iris))
ft_1 <- set_header_df(ft_1, mapping = typology, key = "col_keys")
ft_1 <- merge_h(ft_1, part = "header")
ft_1 <- merge_v(ft_1, j = "Species", part = "header")
ft_1 <- theme_vanilla(ft_1)
ft_1 <- fix_border_issues(ft_1)
ft_1

tytopology <- data.frame(
               "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

```r
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.

**Illustrations**

**See Also**

Other functions to add rows in header or footer: `add_footer_lines()`, `add_footer_row()`, `add_footer()`, `add_header_lines()`, `add_header_row()`, `add_header()`, `separate_header()`, `set_header_footer_df`

**Examples**

```r
t <- flextable(head(iris))
t <- set_header_labels(t,
  Sepal.Length = "Sepal length",
  Sepal.Width = "Sepal width", Petal.Length = "Petal length",
  Petal.Width = "Petal width"
)

t <- flextable(head(iris))
t <- set_header_labels(t,
  values = list(,
    Sepal.Length = "Sepal length",
    Sepal.Width = "Sepal width",
    Petal.Length = "Petal length",
    Petal.Width = "Petal width"
  )
)
t
```

**Description**

Set table layout and table width. Default to fixed algorithm.

If layout is fixed, column widths will be used to display the table; width is ignored.

If layout is autofit, column widths will not be used; table width is used (as a percentage).
Usage

`set_table_properties(x, layout = "fixed", width = 0)`

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.

Illustrations

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 <- qflextable(head(cars))
ft_2 <- set_table_properties(ft_1, width = .5, layout = "autofit")
ft_2
```

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

```r
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')

Illustrations

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 `tabulator()` function.

**Usage**

```r
summarizor(x, by = character(), overall_label = NULL)
```

**Arguments**

- `x`: dataset
- `by`: columns names to be used as grouping columns
- `overall_label`: label to use as overall label

**Illustrations**

- `ft_1` appears as:
- `ft_2` appears as:

**Note**

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

**See Also**

- `fmt_2stats`

**Examples**

```r
z <- summarizor(CO2[-c(1, 4)],
    by = "Treatment",
    overall_label = "Overall"
)
```

# version 1 ----
```
# tab_1 <- tabulator(
#   x = z,
#   rows = c("variable", "stat"),
#   columns = "Treatment",
#   blah = as_paragraph(
#     as_chunk(
#       fmt_2stats(
#         num1 = stat, num2 = value, cts = cts, pcts = percent
#     )))
```
surround

\)
\)
\)

\)
l愉

# version 2 ----
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)
\)

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

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

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)) {

tabulator

Tabulation of aggregations

Description

It tabulates a data.frame representing an aggregation which is then transformed as a 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: 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.

Illustrations
ft_1 appears as:

ft_2 appears as:

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

n_format <- function(z){
  x <- sprintf("%.0f", z)
  x[is.na(z)] <- "-"
  x
}

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), formatter = n_format ))
  )

  ft_1 <- as_flextable(cft_1)
  ft_1
}

if(require("data.table") && require("ggplot2")){

  multi_fun <- function(x) {
    list(mean = mean(x),
         sd = sd(x))
  }

  myformat <- function(z){
    x <- sprintf("%.1f", z)
    x[is.na(z)] <- ""
    x
  }

  grey_txt <- fp_text_default(color = "gray")

  dat <- as.data.table(ggplot2::diamonds)
  dat <- dat[cut %in% c("Fair", "Good", "Very Good")]
  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_2 <- tabulator(
    x = dat, rows = c("cut", "color"),
    columns = "clarity",
    'z stats' = as_paragraph(
      as_chunk(z.mean, formatter = myformat)),
    'y stats' = as_paragraph(
      as_chunk(y.mean, formatter = myformat),
      as_chunk("(\w0B1 ", props = grey_txt),
      as_chunk(y.sd, formatter = myformat, props = grey_txt),
      as_chunk(")", props = grey_txt)
  )

  ft_2 <- as_flextable(tab_2)
ft_2 <- autofit(x = ft_2, add_w = .05)
ft_2
}

if(require("data.table")) {
  # data.table version
  dat <- melt(as.data.table(iris),
    id.vars = "Species",
    variable.name = "name", value.name = "value"),
    list(avg = mean(value, na.rm = TRUE),
        sd = sd(value, na.rm = TRUE)),
    by = c("Species", "name")
  ]
  # dplyr version
  # library(dplyr)
  # dat <- iris %>%
  # pivot_longer(cols = -c(Species)) %>%
  # group_by(Species, name) %>%
  # summarise(avg = mean(value, na.rm = TRUE),
  #   sd = sd(value, na.rm = TRUE),
  #   .groups = "drop")

  tab_3 <- tabulator(
    x = dat, rows = c("Species"),
    columns = "name",
    `mean (sd)` = as_paragraph( as_chunk(avg),
        " (", as_chunk(sd), ")")
  )
  ft_3 <- as_flextable(tab_3, separate_with = character(0))
  ft_3
}
init_flextable_defaults()

---

**tabulator_colnames**  
*column keys of tabulator objects*

**Description**

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

... any filter conditions that use variables names, the same than the argument columns of function `tabulator()` (tabulator(columns = c("col1", "col2"))).

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, 2L, 0L, 3L,
            2L, 1L, 2L, 0L, 12L, 4L, 1L, 7L, 3L, 1L, 5L, 5L, 3L, 10L,
            4L, 1L, 4L, 2L, 0L, 3L, 1L, 0L, 4L, 4L, 2L, 42L, 28L, 19L, 26L,
            19L, 11L, 12L, 10L, 7L, 10L, 5L, 5L, 0L, 3L, 4L, 3L, 3L,
            1L, 2L, 3L,
  risktime = c(157L, 77L, 21L, 139L, 68L, 17L, 126L, 63L, 14L, 102L, 55L,
               12L, 88L, 50L, 10L, 82L, 45L, 8L, 76L, 42L, 6L, 134L, 71L,
               22L, 110L, 63L, 18L, 96L, 58L, 14L, 86L, 42L, 10L, 66L,
               35L, 8L, 59L, 32L, 8L, 51L, 28L, 6L, 212L, 130L, 101L,
               136L, 72L, 63L, 90L, 42L, 43L, 64L, 21L, 32L, 47L, 14L,
               21L, 39L, 13L, 14L, 29L, 7L, 10L,
  time = rep(as.character(1:7), 3),
  histology = rep(as.character(1:3), 21),
  stage = rep(as.character(1:3), each = 21)
)

datasup_first <- data.frame(
  time = factor(1:7, levels = 1:7),
  zzz = runif(7)
)

z <- tabulator(cancer_dat,
```
```r
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_alafoli**  
*Apply alafoli theme*

**Description**

Apply alafoli theme

**Usage**

```r
theme_alafoli(x)
```

**Arguments**

- `x`  
  a flextable object

**Illustrations**
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_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
```

### Illustrations

<table>
<thead>
<tr>
<th>theme_booktabs</th>
<th>Apply booktabs theme</th>
</tr>
</thead>
</table>

### Description

Apply theme booktabs to a `flextable`

### Usage

```r
tHEME_BOOKTABLES(x, bold_header = FALSE, ...)
```

### Arguments

- `x` a `flextable` object
- `bold_header` header will be bold if `TRUE`
- `...` unused

### Illustrations
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_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**

```r
theme_box(x)
```

**Arguments**

- `x` a flextable object

**Illustrations**
**theme_tron**

**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_flextab_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_flextab_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_flextab_defaults()`, `set_flextab_defaults()`, `theme_alafoli()`, `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**

```r
theme_tron(x)
```

**Arguments**

- `x` a flextable object

**Illustrations**
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_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

### Illustrations
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_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

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

Arguments

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

Illustrations
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_booktabs()`, `theme_box()`, `theme_tron_legacy()`, `theme_tron()`, `theme_vanilla()`, `theme_zebra()`

Examples

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

---

### theme_vanilla

**Apply vanilla theme**

**Description**

Apply theme vanilla to a `flextable`: The external horizontal lines of the different parts of the table (body, header, footer) are black 2 points thick, the external horizontal lines of the different parts are black 0.5 point thick. Header text is bold, text columns are left aligned, other columns are right aligned.

**Usage**

`theme_vanilla(x)`

**Arguments**

- `x` a `flextable` object
behavior

Theme functions are not like 'ggplot2' themes. They are applied to the existing table immediately. If you add a row in the footer, the new row is not formatted with the theme. The theme function applies the theme only to existing elements when the function is called.

That is why theme functions should be applied after all elements of the table have been added (mainly additionnal header or footer rows).

If you want to automatically apply a theme function to each flextable, you can use the theme_fun argument of set_flextable_defaults(); be aware that this theme function is applied as the last instruction when calling flextable() - so if you add headers or footers to the array, they will not be formatted with the theme.

You can also use the post_process_html argument of set_flextable_defaults() (or post_process_pdf, post_process_docx, post_process_pptx) to specify a theme to be applied systematically before the flextable() is printed; in this case, don't forget to take care that the theme doesn't override any formatting done before the print statement.

Illustrations

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_zebra()

Examples

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

theme_zebra Apply zebra theme

Description

Apply theme zebra to a flextable

Usage

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

- `x` a flextable object
- `odd_header`, `odd_body`, `even_header`, `even_body`
  odd/even colors for table header and body

Illustrations

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_booktabs()`, `theme_box()`, `theme_tronLegacy()`, `theme_tron()`, `theme_vader()`, `theme_vanilla()`

Examples

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

Description

Define `df_printer()` as data.frame print method in an R Markdown document.

In a setup run chunk:

```r
flextable::use_df_printer()
```
**use_model_printer**

**Usage**

```r
use_df_printer()
```

**See Also**

```r
df_printer(), flextable()
```

---

**Description**

Define `as_flextab()` 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:

```r
flextable::use_model_printer()
```

**Usage**

```r
use_model_printer()
```

**See Also**

```r
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')

Illustrations

See Also

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

Examples

ft_1 <- flextable(iris[c(1:3, 51:53, 101:103), ])
ft_1 <- theme_box(ft_1)
ft_1 <- merge_v(ft_1, j = 5)
ft_1

ft_2 <- valign(ft_1, j = 5, valign = "top", part = "all")
ft_2
vline

set vertical borders

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')

Illustrations

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

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

# add vertical borders
ft <- vline(ft, border = std_border)
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')

Illustrations

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

```r
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
```
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')

Illustrations

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**

*Delete flextable content*

**Description**

Set content display as a blank " ".

**Usage**

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

**Arguments**

- `x`: 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.

Illustrations

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, 20
  as_flextable.gam, 21
  as_flextable.glm, 22
  as_flextable.grouped_data, 22
  as_flextable.htest, 24
  as_flextable.lm, 25
  as_flextable.tabulator, 27
  as_flextable.xtable, 29

* borders management
  border_inner, 46
  border_inner_h, 47
  border_inner_v, 48
  border_outer, 49
  border_remove, 50
  hline, 87
  hline_bottom, 88
  hline_top, 89
  surround, 133
  vline, 151
  vline_left, 152
  vline_right, 153

* cells formatters
  colformat_char, 50
  colformat_date, 51
  colformat_datetime, 52
  colformat_double, 53
  colformat_image, 55
  colformat_int, 56
  colformat_lgl, 57
  colformat_num, 58
  set_formatter, 126

* chunk elements for paragraph
  as_b, 16
  as_bracket, 17
  as_chunk, 18
  as_equation, 19
  as_highlight, 32
  as_i, 33
  as_image, 33
  as_sub, 37
  as_sup, 38
  as_word_field, 39
  colorize, 61
  gg_chunk, 83
  hyperlink_text, 91
  linerange, 97
  lollipop, 99
  minibar, 105
  plot_chunk, 111

* flextable dimensions
  autofit, 40
  dim.flextable, 65
  dim.pretty, 66
  fit_to_width, 68
  flextable_dim, 71
  height, 84
  hrule, 90
  ncol_keys, 106
  nrow_part, 107
  set_table_properties, 129
  width, 154

* flextable merging function
  merge_at, 101
  merge_h, 102
  merge_h_range, 102
  merge_none, 103
  merge_v, 104

* flextable print function
  as_raster, 36
  df_printer, 64
  flextable_to_rmd, 72
  htmltools_value, 91
  knit_print.flextable, 93
  plot.flextable, 110
  print.flextable, 113
  save_as_docx, 116
  save_as_html, 117

156
save_as_image, 118
save_as_pptx, 120

* functions for defining formatting properties
  fp_border_default, 79
  fp_text_default, 80

* functions for mixed content paragraphs
  append_chunks, 15
  as_paragraph, 35
  compose, 61
  prepend_chunks, 112

* functions related to themes
  get_flextable_defaults, 82
  set_flextable_defaults, 123
  theme_alafoli, 140
  theme_booktabs, 141
  theme_box, 142
  theme_tron, 143
  theme_tron_legacy, 144
  theme_vader, 145
  theme_vanilla, 146
  theme_zebra, 147

* functions that add lines in the table
  add_body, 5
  add_body_row, 6
  add_footer, 7
  add_footer_lines, 8
  add_footer_row, 9
  add_header, 10
  add_header_row, 13

* functions that add rows in the table
  add_header_lines, 12

* functions to add rows in header or footer
  add_footer, 7
  add_footer_lines, 8
  add_footer_row, 9
  add_header, 10
  add_header_lines, 12
  add_header_row, 13
  separate_header, 120
  set_header_footer_df, 127
  set_header_labels, 128

* sugar functions for table style
  align, 14
  bg, 43
  bold, 46
  color, 59
  empty_blanks, 67
  font, 75
  fontsize, 76
  highlight, 86
  italic, 92
  line_spacing, 98
  padding, 108
  rotate, 115
  valign, 150
  add_body, 5, 7–11, 13
  add_body_row, 6, 6, 8–11, 13
  add_footer, 6, 7, 7–9–13, 121, 128, 129
  add_footer_lines, 6–8, 8, 10–13, 121, 128, 129
  add_footer_row, 6–9, 9, 11–13, 121, 128, 129
  add_header, 6–10, 10, 12, 13, 121, 128, 129
  add_header_lines, 8–11, 12, 13, 121, 128, 129
  add_header_row, 6–12, 13, 121, 128, 129
  add_header_row(), 7
  add_latex_dep, 14
  add_latex_dep(), 95
  align, 14, 43, 46, 60, 67, 76, 77, 86, 93, 99, 108, 116, 150
  align_nottext_col (align), 14
  align_text_col (align), 14
  append_chunks, 15, 35, 62, 112
  append_chunks(), 16–19, 32, 33, 37–39, 61, 81, 91
  as_b, 16, 18, 19, 32–34, 37–39, 61, 83, 92, 98, 100, 106, 111
  as_b(), 62
  as_bracket, 17, 17, 18, 19, 32–34, 37–39, 61, 83, 92, 98, 100, 106, 111
  as_chunk, 17, 18, 19, 32–34, 37–39, 61, 83, 92, 98, 100, 106, 111
  as_chunk(), 15, 16, 35, 61, 62, 81, 112
  as_equation, 17, 18, 19, 32–34, 37–39, 61, 83, 92, 98, 100, 106, 111
  as_flextable, 20, 21–25, 28, 30
  as_flextable(), 27, 149
  as_flextable.brmsfit
    (as_flextable.merMod), 26
  as_flextable.gam, 20, 21, 22–25, 28, 30
  as_flextable.glm, 20, 21, 22, 23–25, 28, 30
  as_flextable.glmmadmb
    (as_flextable.merMod), 26
  as_flextable.glmmTMB
    (as_flextable.merMod), 26
dim_pretty, 41, 66, 66, 68, 71, 85, 90, 107, 115, 130, 155
dim_pretty(), 40, 109
div(), 91
fit_to_width, 41, 66, 68, 71, 85, 90, 107, 130, 155
fix_border_issues, 69
flextable, 69
flextable(), 5–7, 123, 149, 154
flextable-package, 5
flextable_dim, 41, 66, 68, 71, 85, 90, 107, 130, 155
flextable_html_dependency, 72
flextable_to_rmd, 36, 65, 72, 91, 96, 110, 113, 117–120
flextable_to_rmd(), 94
fmt_2stats, 74
fmt_2stats(), 132
font, 15, 43, 46, 60, 67, 75, 77, 86, 93, 99, 108, 116, 150
fontsize, 15, 43, 46, 60, 67, 76, 86, 93, 99, 108, 116, 150
footers_flextable_at_bkm, 77
footnote, 78
footnote(), 71
format(), 56, 58, 59
formatC, 125
formatC(), 54, 59
fp_border(), 28, 47–49, 79, 87–89, 151–153
fp_border_default, 79, 81
fp_border_default(), 28
fp_text(), 80
fp_text_default, 80, 80
fp_text_default(), 39, 62
get_flextable_defaults, 82, 125, 141–148
get_flextable_defaults(), 70, 123
gg_chunk, 17–19, 32–34, 37–39, 61, 83, 92, 98, 100, 106, 111
gg_chunk(), 16, 112
headers_flextable_at_bkm, 84
height, 41, 66, 68, 71, 84, 90, 107, 130, 155
height(), 109, 115
height_all(height), 84
hline, 47–50, 87, 88, 89, 134, 151–153
hline(), 42, 80, 133
hline_bottom, 47–50, 87, 88, 89, 134, 151–153
hline_top, 47–50, 87, 88, 89, 134, 151–153
hline_top(), 133
hrule, 41, 66, 68, 71, 85, 90, 107, 130, 155
hrule(), 84, 85, 115
HTML, 91
htmltools_value, 36, 65, 74, 91, 96, 110, 113, 117–120
hyperlink_text, 17–19, 32–34, 37–39, 61, 83, 91, 98, 100, 106, 111
hyperlink_text(), 35
init_flextable_defaults
(set_flextable_defaults), 123
init_flextable_defaults(), 70
kmeans, 25
knit_meta_add, 14
knit_print.flextable, 36, 65, 72, 74, 91, 93, 110, 113, 117–120, 123
knit_print.flextable(), 69, 71, 113
line_spacing, 15, 43, 46, 60, 67, 76, 77, 86, 93, 98, 108, 116, 150
linerange, 17–19, 32–34, 37–39, 61, 83, 92, 97, 100, 106, 111
merge_at, 101, 102–104
merge_h, 101, 102, 103, 104
merge_h(), 11
merge_h_range, 101, 102, 102, 103, 104
merge-none, 101–103, 103, 104
merge-v, 101–103, 104
merge-v(), 11
minibar, 17–19, 32–34, 37–39, 61, 83, 92, 98, 100, 105, 111
minibar(), 35
mk-par (compose), 61
mk-par(), 75, 136
width(), 109

xtab_to_flextab
   (as_flextab.xtable), 29