Package ‘huxtable’

October 13, 2022

Type Package

Title Easily Create and Style Tables for LaTeX, HTML and Other Formats

Version 5.5.0

Author David Hugh-Jones [aut, cre]

Maintainer David Hugh-Jones <davidhughjones@gmail.com>

Description Creates styled tables for data presentation. Export to HTML, LaTeX, RTF, 'Word', 'Excel', and 'PowerPoint'. Simple, modern interface to manipulate borders, size, position, captions, colours, text styles and number formatting. Table cells can span multiple rows and/or columns. Includes a 'huxreg' function for creation of regression tables, and 'quick_*' one-liners to print data to a new document.

License MIT + file LICENSE

URL https://hughjonesd.github.io/huxtable/

BugReports https://github.com/hughjonesd/huxtable/issues

Imports assertthat, commonmark, fansi, generics, glue, htmltools, memoise, R6, rlang, stats, stringi, stringr (>= 1.2.0), tidyselect, utils, xml2

Suggests AER, bookdown, broom (>= 0.5.1), broom.mixed, covr, crayon, devtools, dplyr (>= 0.7.0), flextable (>= 0.5.5), fitExtra (>= 0.0.2), ggplot2, httr, knitr, lme4, lmerTest, nlme, nnet, officer, openxlsx, psych, quarto, rmarkdown, sandwich, scales, testthat, tibble, tinytex

SystemRequirements LaTeX packages: adjustbox, array, calc, caption, colortbl, fontspec, graphicx, hhline, hyperref, multirow, siunitx, tabularx, threeparttable, ulem, wrapfig

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

Depends R (>= 2.10)
NeedsCompilation: no
Repository: CRAN
Date/Publication: 2022-06-15 11:30:02 UTC

R topics documented:

- huxtable-package ........................................ 4
- add_colnames ........................................... 5
- add_footnote ........................................... 6
- add_rows .................................................. 7
- align ...................................................... 8
- as_flextable ............................................. 9
- as_huxtable ............................................. 10
- as_Workbook ............................................ 11
- background_color ...................................... 12
- bold ....................................................... 15
- border-colors ........................................... 16
- border-styles .......................................... 17
- borders .................................................. 18
- brdr ....................................................... 20
- brdr_thickness .......................................... 21
- by_cases .................................................. 21
- by_colorspace .......................................... 22
- by_function ............................................. 23
- by_quantiles ............................................. 24
- by_ranges ............................................... 26
- by_regex .................................................. 27
- by_rows ................................................... 28
- by_values ............................................... 29
- caption .................................................... 30
- caption_pos ............................................... 31
- caption_width ............................................ 32
- cbind.huxtable .......................................... 32
- column_to_header ...................................... 34
- col_width ............................................... 35
- escape_contents ........................................ 36
- final ....................................................... 37
- fmt_percent ............................................. 37
- fmt_pretty .............................................. 38
- font ....................................................... 39
- font_size ................................................ 40
- guess_knitr_output_format ............................ 41
- header_cols ............................................. 42
- height ..................................................... 43
- huxreg .................................................... 43
- huxtable ................................................ 46
- huxtable-FAQ .......................................... 48
<table>
<thead>
<tr>
<th>R topics documented</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>huxtable-news</td>
<td>49</td>
</tr>
<tr>
<td>huxtable-options</td>
<td>62</td>
</tr>
<tr>
<td>hux_logo</td>
<td>63</td>
</tr>
<tr>
<td>insert_column</td>
<td>64</td>
</tr>
<tr>
<td>jams</td>
<td>65</td>
</tr>
<tr>
<td>knit_print.data.frame</td>
<td>66</td>
</tr>
<tr>
<td>knit_print.huxtable</td>
<td>67</td>
</tr>
<tr>
<td>label</td>
<td>67</td>
</tr>
<tr>
<td>latex_float</td>
<td>68</td>
</tr>
<tr>
<td>mapping-functions</td>
<td>69</td>
</tr>
<tr>
<td>markdown</td>
<td>71</td>
</tr>
<tr>
<td>merge_across</td>
<td>72</td>
</tr>
<tr>
<td>merge_cells</td>
<td>73</td>
</tr>
<tr>
<td>merge_repeated_rows</td>
<td>74</td>
</tr>
<tr>
<td>mutate.huxtable</td>
<td>75</td>
</tr>
<tr>
<td>na_string</td>
<td>76</td>
</tr>
<tr>
<td>number_format</td>
<td>77</td>
</tr>
<tr>
<td>padding</td>
<td>79</td>
</tr>
<tr>
<td>position</td>
<td>80</td>
</tr>
<tr>
<td>print.huxtable</td>
<td>81</td>
</tr>
<tr>
<td>print_html</td>
<td>82</td>
</tr>
<tr>
<td>print_latex</td>
<td>83</td>
</tr>
<tr>
<td>print_md</td>
<td>84</td>
</tr>
<tr>
<td>print_rtf</td>
<td>85</td>
</tr>
<tr>
<td>print_screen</td>
<td>86</td>
</tr>
<tr>
<td>quick-output</td>
<td>87</td>
</tr>
<tr>
<td>report_latex_dependencies</td>
<td>89</td>
</tr>
<tr>
<td>restack-across-down</td>
<td>90</td>
</tr>
<tr>
<td>rotation</td>
<td>92</td>
</tr>
<tr>
<td>rowspecs</td>
<td>93</td>
</tr>
<tr>
<td>row_height</td>
<td>94</td>
</tr>
<tr>
<td>rtf_fc_tables</td>
<td>95</td>
</tr>
<tr>
<td>sanitize</td>
<td>96</td>
</tr>
<tr>
<td>set-multiple</td>
<td>97</td>
</tr>
<tr>
<td>set-outer</td>
<td>99</td>
</tr>
<tr>
<td>set_contents</td>
<td>100</td>
</tr>
<tr>
<td>set_default_properties</td>
<td>100</td>
</tr>
<tr>
<td>set_markdown_contents</td>
<td>101</td>
</tr>
<tr>
<td>spans</td>
<td>102</td>
</tr>
<tr>
<td>split-across-down</td>
<td>103</td>
</tr>
<tr>
<td>stripe</td>
<td>104</td>
</tr>
<tr>
<td>stripes</td>
<td>105</td>
</tr>
<tr>
<td>style-functions</td>
<td>106</td>
</tr>
<tr>
<td>t.huxtable</td>
<td>107</td>
</tr>
<tr>
<td>table_environment</td>
<td>108</td>
</tr>
<tr>
<td>tabular_environment</td>
<td>108</td>
</tr>
<tr>
<td>text_color</td>
<td>109</td>
</tr>
<tr>
<td>themes</td>
<td>110</td>
</tr>
</tbody>
</table>
**Description**

Huxtable is a package for creating HTML and LaTeX tables. It provides similar functionality to xtable, with a simpler interface.

**Quick start**

To create a huxtable object, use `huxtable()` or `as_huxtable()`:

```r
library(huxtable)
employees <- huxtable(
  Names = c("Hadley", "Yihui", "Dirk"),
  Salaries = c(1e5, 1e5, 1e5),
  add_colnames = TRUE
)
car_hux <- as_hux(mtcars)
```

You can then set properties which affect how the huxtable is displayed:

```r
# make the first row bold:
bold(employees)[1, ] <- TRUE

# change the font size everywhere:
font_size(employees) <- 10
```

Or you can use a tidyverse style with the pipe operator:

```r
library(magrittr)
employees <- employees %>%
  set_font_size(10) %>%
  set_bold(1, everywhere, TRUE)
```

For more information, see the website or read the vignette with `vignette("huxtable")`. See `huxtable-FAQ` for frequently asked questions, including ways to get help. To report a bug, or suggest an enhancement, visit github.
Description

Add a first row of column names, or a first column of row names, to the huxtable.

Usage

```r
add_colnames(ht, ...)  
## S3 method for class 'huxtable'
add_colnames(ht, rowname = NULL, ...)

add_rownames(ht, ...)
## S3 method for class 'huxtable'
add_rownames(ht, colname = "rownames", preserve_rownames = TRUE, ...)
```

Arguments

- `ht` A huxtable.
- `...` Arguments passed to methods.
- `rowname` Optional row name for the new row of column names.
- `colname` Column name for the new column of row names.
- `preserve_rownames` Preserve existing row names.

Details

Note that `add_colnames` will change the mode of all columns to character. Also note that it will move your rows down by one: what was row 1 will now be row 2, and the column names will now be row 1.

`add_colnames` preserves column names. `add_rownames` only preserves them if asked to.

Value

The modified object.

Examples

```r
ht <- huxtable(
  First = rnorm(5),
  Second = rnorm(5),
  add_rownames = FALSE
)
add_rownames(ht)
```
add_colnames(ht)

# Out by 1:
add_rownames(add_colnames(ht))

# Better:
add_colnames(add_rownames(ht))

# Alternatively:
add_colnames(add_rownames(ht, ""))

---

**add_footnote**  
*Add a row with a footnote*

**Description**

This adds a single row at the bottom. The first cell contains the footnote; it spans all table columns and has an optional border above.

**Usage**

```r
add_footnote(ht, text, border = 0.8, number_format = NA, ...)
```

**Arguments**

- **ht**: A huxtable.
- **text**: Text for the footnote.
- **border**: Width of the footnote's top border. Set to 0 for no border, or `NULL` to leave the border unchanged.
- **number_format**: Number format for the footnote cell.
- **...**: Other properties, passed to `set_cell_properties()` for the footnote cell.

**Value**

The modified huxtable

**Examples**

```r
jams <- add_footnote(jams,  
  "* subject to availability")
jams
```
add_rows

Insert one huxtable into another

Description

These functions combine two huxtables or similar objects and return the result.

Usage

add_rows(x, y, after = nrow(x), copy_cell_props = TRUE)
add_columns(x, y, after = ncol(x), copy_cell_props = TRUE)

Arguments

x, y Huxtables or objects that can be converted by as_hux
after Row or column after which y is inserted. Can be 0. Can be a row or column name. The default adds y to the end of x.
copy_cell_props Logical. Passed to rbind.huxtable() or cbind.huxtable().

Details

Arguments in ... can include copy_cell Props.

Value

A huxtable.

See Also

insert_row() and insert_column(), which insert multiple values into a single row.

Examples

ht <- hux("Gooseberry", 2.15)
add_rows(jams, ht)
add_rows(jams, ht, after = 1)

mx <- matrix(
c("Sugar", "50%", "60%", "40%", "Weight (g)", 300, 250, 300),
4, 2)
add_columns(jams, mx)
**align**

*Set the horizontal alignment of cell content*

**Description**

Values may be "left", "center", "right", NA or a single character. If value is a single character (e.g. a decimal point), then the cell is aligned on this character.

**Usage**

```r
align(ht)
align(ht) <- value
set_align(ht, row, col, value)
map_align(ht, row, col, fn)
```

**Arguments**

- `ht` A huxtable.
- `row` A row specifier. See `rowspecs` for details.
- `col` An optional column specifier.
- `fn` A mapping function. See `mapping-functions` for details.
- `value` A character vector or matrix. Set to NA to reset to the default, which is "left".

**Value**

`align()` returns the `align` property. `set_align()` returns the modified huxtable.

**Aligning on a decimal point**

To align cells on the decimal point, set `align` to "." or any other single character (e.g. "," in European languages).

By default, huxtable aligns these cells by padding with spaces. The mechanics of this were improved for LaTeX in version 5.3.0, but are still not perfect. Using a fixed-width font may help.

If `options("huxtable.latex_siunitx_align")` is set to TRUE, then in LaTeX output, numbers in these cells will be surrounded by `\tablenum{}`. See the siunitx documentation for more details.

Note that this may have other side-effects, for example `1e3` becomes `1 \times 10^3`.

To use non-default decimal points, set both `align(ht)` and `number_format()`. See the example.

**Examples**

```r
codes <- c(1, 1.5, 1.03, 10, 10.01)
number_hux <- as_hux(matrix(numbers, 5, 5))
number_format(number_hux) <- ".4g"
number_format(number_hux)[, 5] <- fmt_pretty(
```

...
number_hux <- map_align(number_hux,
  by_cols("left", "center", "right", ".", ",", "")
)

alignments <- c(
  "left",
  "centre",
  "right",
  "decimal (.)",
  "decimal (,)"
)

number_hux <- rbind(
  alignments,
  number_hux
)

align(number_hux)

number_hux

---

**as_flextable**

*Convert a huxtable for Word/Powerpoint*

**Description**

Huxtables can be converted to `flextable::flextable()` objects, for use in Word and Powerpoint documents.

**Usage**

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

```r
## S3 method for class 'huxtable'
as_flextable(x, colnames_to_header = FALSE, ...)
```

**Arguments**

- `x` A huxtable.
- `...` Not used.
- `colnames_to_header` Use huxtable column names as the header. If FALSE, the flextable will contain only a body and no header.
Details

With recent versions of "flextable" and Pandoc, huxtables can be automatically outputted from rmarkdown word_document and/or powerpoint_presentation documents. (Powerpoint presentations require pandoc version >= 2.4.0.)

Properties are supported, with the following exceptions:

- Rotation of 0, 90 or 270 is supported.
- Non-numeric widths and heights are not supported. Table heights are treated as a proportion of 9 inches; table widths are treated as a proportion of 6 inches. So e.g. height(ht) <- 0.5 will give a height of 4.5 inches.
- Table wrap and table position are not supported.
- Border style "double" is not supported and becomes "solid".
- Captions are supported with recent versions of flextable, but not caption_pos() or caption_width().

Value

an object of class flextable.

Challenge

Try to say as_flextable.huxtable ten times without pausing.

Examples

ht <- hux(a = 1:3, b = 1:3)
ft <- as_flextable(ht)
## Not run:
my_doc <- officer::read_docx()
my_doc <- flextable::body_add_flextable(my_doc, ft)
print(my_doc, target =
"path/to/my_doc.docx")
## End(Not run)

as_huxtable

Convert objects to huxtables

Description

as_huxtable or as_hux converts an object to a huxtable. Conversion methods exist for data frames and tibbles, tables, ftables, matrices and (most) vectors.
Usage

```r
as_huxtable(x, ...)
```

```r
as_hux(x, ...)
```

```r
## Default S3 method:
as_huxtable(
x,
  add_colnames = getOption("huxtable.add_colnames", TRUE),
  add_rownames = FALSE,
  autoformat = getOption("huxtable.autoformat", TRUE),
  ...
)
```

```r
is_huxtable(x)
```

```r
is_hux(x)
```

Arguments

- `x` Object to convert.
- `...` Arguments passed on to `huxtable()`.
- `add_colnames` If TRUE, add a first row of column names to the huxtable.
- `add_rownames` If TRUE or a character string, add a first column of row names to the huxtable. The string gives the name for the new column (or "rownames" for TRUE).
- `autoformat` If TRUE, automatically format columns by type. See below.

Details

`is_hux[table]` tests if an object is a huxtable.

For `table` objects, `add_colnames` and `add_rownames` are TRUE by default. For `matrix` objects, they are FALSE. Other classes use `options("huxtable.add_colnames")`, which is TRUE by default; `add_rownames` is FALSE.

For `dplyr::grouped_df()` objects, groups will be converted to header rows if `groups_to_header` is TRUE.

Value

An object of class "huxtable".

Examples

```r
dfr <- data.frame(a = 1:5,
          b = letters[1:5],
          stringsAsFactors = FALSE)
```
as_huxtable(dfr)

mx <- matrix(letters[1:12], 4, 3)
as_huxtable(mx, add_colnames = FALSE)

library(stats)
tbl <- table(
  Wool = warpbreaks$wool,
  Tension = warpbreaks$tension
)
as_huxtable(tbl) # adds row and column names by default

# adding rownames:
as_hux(mtcars[1:3,], add_colnames = TRUE,
  add_rownames = "Car")

if (requireNamespace("dplyr")) {
  iris_grp <- dplyr::group_by(iris[c(1:4, 51:54, 101:104), ], Species)
as_hux(iris_grp, groups_to_headers = TRUE)
}

---

as_Workbook

Convert a huxtable for Excel

Description

If the openxlsx package is installed, Huxtables can be converted to openxlsx::openxlsx() Workbook objects, for use in Excel documents.

Usage

as_Workbook(ht, ...)

## S3 method for class 'huxtable'
as_Workbook(
  ht,
  Workbook = NULL,
  sheet = "Sheet 1",
  write_caption = TRUE,
  start_row = 1,
  start_col = 1,
  ...
)

Arguments

ht  A huxtable.
...
Not used.
Workbook  An existing Workbook object. By default, a new workbook will be created.
sheet  Name for the worksheet where the huxtable will be created.
write_caption  If TRUE, print any caption in the row above or below the table.
start_row, start_col
Number. Write data starting at the given row and column.

Details

Use \texttt{openxlsx::saveWorkbook()} to save the resulting object to an Excel file. Properties are supported with the following exceptions:

- Non-numeric column widths and row heights, table width and height.
- Decimal padding.
- Cell padding.
- Table position.
- Caption width.

\texttt{Huxtable} tries to guess appropriate widths and height for rows and columns; numeric \texttt{width()} and \texttt{height()} are treated as scaling factors.

Contents are only stored as numbers if a whole column is "numeric", i.e. can be converted by \texttt{as.numeric()}). Otherwise, they are stored as text.

Value

An object of class \texttt{Workbook}.

Examples

\begin{verbatim}
wb <- as_Workbook(jams)

## Not run:
openxlsx::saveWorkbook(wb,  
"my-excel-file.xlsx")

## End(Not run)

# multiple sheets in a single workbook:
wb <- openxlsx::createWorkbook()  
wb <- as_Workbook(jams,  
   Workbook = wb, sheet = "sheet1")
wb <- as_Workbook(  
   hux("Another", "huxtable"),  
   Workbook = wb,  
   sheet = "sheet2")
\end{verbatim}
background_color

Description

Colors can be in any format understood by R:

- A color name like "darkred"
- A HTML string like "#FF0000"
- The result of a function like rgb(1, 0, 0) or grey(0.5)

Usage

background_color(ht)
background_color(ht) <- value
set_background_color(ht, row, col, value)
map_background_color(ht, row, col, fn)

Arguments

ht A huxtable.
row A row specifier. See rowspecs for details.
col An optional column specifier.
fn A mapping function. See mapping-functions for details.
value A character vector or matrix.

Details

Transparent colors are not guaranteed to work at present.

Value

background_color() returns the background_color property. set_background_color() returns the modified huxtable.

See Also

Other formatting functions: bold(), font_size(), font(), na_string(), number_format(), text_color()
**Examples**

```r
color(jams) <- grey(0.7)

color(jams)

set_background_color(jams, "yellow")
set_background_color(jams, 2:3, 1, "yellow")
map_background_color(jams, by_rows("yellow", grey(0.7)))
```

**Description**

Make cell text bold or italic

**Usage**

```r
bold(ht)
bold(ht) <- value
set_bold(ht, row, col, value = TRUE)
map_bold(ht, row, col, fn)

italic(ht)
italic(ht) <- value
set_italic(ht, row, col, value = TRUE)
map_italic(ht, row, col, fn)
```

**Arguments**

- `ht`: A huxtable.
- `row`: A row specifier. See `rowspecs` for details.
- `col`: An optional column specifier.
- `fn`: A mapping function. See `mapping-functions` for details.
- `value`: A logical vector or matrix.
  Set to NA to reset to the default, which is FALSE.

**Value**

`bold()` returns the `bold` property. `set_bold()` returns the modified huxtable.

**See Also**

Other formatting functions: `background_color()`, `font_size()`, `font()`, `na_string()`, `number_format()`, `text_color()`
Examples

```r
bold(jams) <- TRUE
bold(jams)

set_bold(jams, FALSE)
set_bold(jams,
  2:3, 1, FALSE)
map_bold(jams,
  by_rows(FALSE, TRUE))
```

**Description**

These functions set border colors.

**Usage**

```r
left_border_color(ht)
left_border_color(ht) <- value
set_left_border_color(ht, row, col, value )
map_left_border_color(ht, row, col, fn)

right_border_color(ht)
right_border_color(ht) <- value
set_right_border_color(ht, row, col, value )
map_right_border_color(ht, row, col, fn)

top_border_color(ht)
top_border_color(ht) <- value
set_top_border_color(ht, row, col, value )
map_top_border_color(ht, row, col, fn)

bottom_border_color(ht)
bottom_border_color(ht) <- value
set_bottom_border_color(ht, row, col, value )
map_bottom_border_color(ht, row, col, fn)
```

**Arguments**

- `ht` A huxtable.
- `row` A row specifier. See `rowspecs` for details.
- `col` An optional column specifier.
- `fn` A mapping function. See `mapping-functions` for details.
- `value` A valid R color, e.g. "red", "#FF0000".
Details

Borders are always "collapsed": right\_border\_color(ht)[, 1] is the same as left\_border\_color(ht)[, 2], and setting one sets the other.

Limitations

- Transparent borders with the alpha channel set are not guaranteed to work.

See Also

- set\_multiple, brdr()

Other border properties: border\_styles, borders

Examples

```r
jams <- set\_all\_borders(jams)
bottom\_border\_color(jams)[1, ] <- "red"
jams

set\_bottom\_border\_color(jams, "blue")
```

---

**border\_styles** Set border styles

Description

These functions set border styles.

Usage

```r
left\_border\_style(ht)
left\_border\_style(ht) <- value
set\_left\_border\_style(ht, row, col, value )
map\_left\_border\_style(ht, row, col, fn)

right\_border\_style(ht)
right\_border\_style(ht) <- value
set\_right\_border\_style(ht, row, col, value )
map\_right\_border\_style(ht, row, col, fn)

top\_border\_style(ht)
top\_border\_style(ht) <- value
set\_top\_border\_style(ht, row, col, value )
map\_top\_border\_style(ht, row, col, fn)
```
borders

```
bottom_border_style(ht)
bottom_border_style(ht) <- value
set_bottom_border_style(ht, row, col, value )
map_bottom_border_style(ht, row, col, fn)
```

**Arguments**

- **ht** A huxtable.
- **row** A row specifier. See `rowspecs` for details.
- **col** An optional column specifier.
- **fn** A mapping function. See `mapping-functions` for details.
- **value** One of "solid", "double", "dashed" or "dotted".

**Details**

Borders are always "collapsed": `right_border_style(ht)[, 1]` is the same as `left_border_style(ht)[, 2]`, and setting one sets the other.

**Limitations**

- In HTML, you will need to set a width of at least 3 to get a double border.
- Only "solid" and "double" styles are currently implemented in LaTeX.

**See Also**

- `set-multiple, brdr()`
- Other border properties: `border-colors, borders`

**Examples**

```
jams <- set_all_borders(jams)
bottom_border_style(jams)[1, ] <- "dotted"
jams

set_bottom_border_style(jams, "double")
```

---

<table>
<thead>
<tr>
<th>borders</th>
<th>\textit{Set borders}</th>
</tr>
</thead>
</table>

**Description**

These functions set borders between cells.
Usage

```r
left_border(ht)
left_border(ht) <- value
set_left_border(ht, row, col, value = 0.4)
map_left_border(ht, row, col, fn)
```

```r
right_border(ht)
right_border(ht) <- value
set_right_border(ht, row, col, value = 0.4)
map_right_border(ht, row, col, fn)
```

```r
top_border(ht)
top_border(ht) <- value
set_top_border(ht, row, col, value = 0.4)
map_top_border(ht, row, col, fn)
```

```r
bottom_border(ht)
bottom_border(ht) <- value
set_bottom_border(ht, row, col, value = 0.4)
map_bottom_border(ht, row, col, fn)
```

```r
## S3 replacement method for class 'huxtable'
left_border(ht) <- value
```

```r
## S3 replacement method for class 'huxtable'
right_border(ht) <- value
```

```r
## S3 replacement method for class 'huxtable'
top_border(ht) <- value
```

```r
## S3 replacement method for class 'huxtable'
bottom_border(ht) <- value
```

Arguments

- `ht` A huxtable.
- `value` A numeric thickness or a `brdr()` object.
- `row` A row specifier. See `rowspecs` for details.
- `col` An optional column specifier.
- `fn` A mapping function. See `mapping-functions` for details.

Details

Borders are always "collapsed": `right_border(ht)[, 1]` is the same as `left_border(ht)[, 2]`, and setting one sets the other.

Setting `left_border(ht) <- number` sets the border thickness. You can set multiple properties at once by using `brdr()`.
Currently in LaTeX, all non-zero border widths on a given line must be the same.

Limitations

- In HTML, you will need to set a width of at least 3 to get a double border.
- Only "solid" and "double" styles are currently implemented in LaTeX, and all non-zero horizontal border widths on a given line must be the same.

See Also

set-multiple

Other border properties: border-colors, border-styles

Examples

```r
bottom_border(jams)[1, ] <- 0.4
jams

bottom_border(jams)[1, ] <- brdr(0.4, "solid", "blue")
jams

set_bottom_border(jams, brdr(0.4, "solid", "green"))
```

---

**brdr**

Create a border object

Description

`brdr()` objects can be passed into `set_top_border()` and friends. They set multiple border properties simultaneously.

Usage

```r
brdr(thickness = 0.4, style = "solid", color = NA_character_)
```

Arguments

- **thickness**: Thickness of the border in points.
- **style**: "solid" (the default), "double", "dashed" or "dotted".
- **color**: String representing a valid color (either a color name or a hexadecimal string like "#00FF00").

Value

An object of class "brdr".
Examples

```r
set_bottom_border(jams, brdr(1, "solid", "red"))
```

---

**Description**

Get thickness of a `brdr()` object

**Usage**

```r
brdr_thickness(x)
```

**Arguments**

- `x` A `brdr()` object.

**Value**

A number or numeric matrix.

**Examples**

```r
brdr_thickness(left_border(jams))
brdr_thickness(brdr(1, "solid", "red"))
```

---

**by_cases**

*Map cell contents to properties using case_when*

**Description**

This function uses `dplyr::case_when()` to set cell properties.

**Usage**

```r
by_cases(..., ignore_na = TRUE)
```

**Arguments**

- `...` A list of two-sided formulas interpreted by `case_when`.
- `ignore_na` If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.
by_colorspace

Details

Within the formulas, the variable `.` will refer to the content of `ht[rows, cols]`, after conversion by `as.matrix()`.

case_when returns NA when no formula LHS is matched. To avoid this, set a default in the last formula: TRUE ~ default.

case_when can’t deal with `brdr()` objects, so you cannot use these in `by_cases()`.

Value

A function for use in `map_***` functions.

See Also

mapping-functions

Other mapping functions: `by_colorspace()`, `by_function()`, `by_quantiles()`, `by_ranges()`, `by_regex()`, `by_rows()`, `by_values()`

Examples

```r
if (! requireNamespace("dplyr")) {
  stop("Please install the 'dplyr' package to run this example")
}

ht <- hux(runif(5), letters[1:5])

map_background_color(ht, by_cases(  
  . == "a" ~ "red",  
  . %in% letters ~ "green",  
  . < 0.5 ~ "pink"  
))
```

by_colorspace Map numeric cell contents smoothly to colors

Description

by_colorspace() can be used to set background, border or text colors, visually differentiating high or low values.

Usage

```r
by_colorspace(  
  ...,  
  range = NULL,  
  na_color = NA,  
  ignore_na = TRUE,  
  colwise = FALSE
)  
```
by_function

Arguments

... Colors
range Numeric endpoints. If NULL, these are determined from the data.
na_color Color to return for NA values. Can be NA itself.
ignore_na If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.
colwise Logical. Calculate breaks separately within each column?

Details

by_colorspace requires the "scales" package.

Value

A function for use in map_*** functions.

See Also

mapping-functions

Other mapping functions: by_cases(), by_function(), by_quantiles(), by_ranges(), by_regex(), by_rows(), by_values()

Examples

if (! requireNamespace("scales")) {
  stop("Please install the \"scales\" package to run this example")
}
ht <- as_hux(matrix(rnorm(25), 5, 5))
map_background_color(ht,
  by_colorspace("red", "yellow", "blue"))
map_background_color(ht,
  by_colorspace("red", "yellow", "blue",
  colwise = TRUE))

by_function Map cell contents to cell properties using a function or scale

Description

This creates a simple wrapper around a function for use in map_***. Useful functions include scales and palettes from the scales package.

Usage

by_function(inner_fn, ignore_na = TRUE)
by_quantiles

Arguments

inner_fn  A one-argument function which maps cell values to property values.
ignore_na If TRUE, NA values in the result will be left unchanged from their previous values.

Otherwise, NA normally resets to the default.

Details

The argument of inner_fn will be as.matrix(ht[row, col]). Be aware how matrix conversion
affects the mode of cell data.

Value

A function for use in map_*** functions.

See Also

mapping-functions

Other mapping functions: by_cases(), by_colorspace(), by_quantiles(), by_ranges(), by_regex(),
by_rows(), by_values()

Examples

ht <- as_hux(matrix(runif(20), 5, 4))

map_background_color(ht,
    by_function(grey))

if (requireNamespace("scales")) {
    map_text_color(ht, by_function(
        scales::seq_gradient_pal()
    ))
}

by_quantiles  Map numeric quantiles to cell properties

Description

These functions split cell values by quantiles. Non-numeric cells are ignored.

Usage

by_quantiles(
    quantiles,
    values,
    right = FALSE,
    extend = TRUE,
by_quantiles

ignore_na = TRUE,
colwise = FALSE
)

by_equal_groups(n, values, ignore_na = TRUE, colwise = FALSE)

Arguments

quantiles  Vector of quantiles.
values     Vector of values. length(values) should be one greater than length(quantiles),
or one less if extend = FALSE.
right      If TRUE, intervals are closed on the right, i.e. if values are exactly equal to a
break, they go in the lower group. Otherwise, intervals are closed on the left,
so equal values go in the higher group. FALSE by default.
extend     Extend breaks to c(-Inf, breaks, Inf). i.e. include numbers below and
above the outermost breaks. TRUE by default.
ignore_na  If TRUE, NA values in the result will be left unchanged from their previous values.
Otherwise, NA normally resets to the default.
colwise    Logical. Calculate breaks separately within each column?
n         Number of equal-sized groups. length(values) should equal n.

Details

by_equal_groups(n, values) splits the data into n equal-sized groups (i.e. it is a shortcut for
by_quantiles(seq(1/n, 1 - 1/n, 1/n), values)).

Value

A function for use in map_*** functions.

See Also

mapping-functions

Other mapping functions: by_cases(), by_colors(), by_function(), by_ranges(), by_regex(),
by_rows(), by_values()

Examples

ht <- hux(rnorm(5), rnorm(5))

map_background_color(ht,
  by_quantiles(
    c(0.2, 0.8),
    c("red", "yellow", "green")
  ))

map_background_color(ht,
  by_quantiles(
by_ranges

Map numeric ranges to cell properties

Description
by_ranges() sets property values for cells falling within different numeric ranges.

Usage
by_ranges(breaks, values, right = FALSE, extend = TRUE, ignore_na = TRUE)

Arguments
breaks A vector of numbers in increasing order.
values A vector of property values. length(values) should be one greater than length(breaks) if extend = TRUE, or one less if extend = FALSE.
right If TRUE, intervals are closed on the right, i.e. if values are exactly equal to a break, they go in the lower group. Otherwise, intervals are closed on the left, so equal values go in the higher group. FALSE by default.
extend Extend breaks to c(-Inf, breaks, Inf), i.e. include numbers below and above the outermost breaks. TRUE by default.
ignore_na If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.

Details
Non-numeric cells return NA. The effects of this depend on ignore_na.

Value
A function for use in map_*** functions.

See Also
mapping-functions
Other mapping functions: by_cases(), by_colorspace(), by_function(), by_quantiles(), by_regex(), by_rows(), by_values()
Examples

```r
ht <- huxtable(c(1, 3, 5))
map_background_color(ht,
  by_ranges(
    c(2, 4),
    c("red", "yellow", "blue")
  ))
map_background_color(ht,
  by_ranges(
    c(2, 4),
    "pink",
    extend = FALSE
  ))
map_background_color(ht,
  by_ranges(
    c(1, 5),
    c("red", "yellow", "green"),
    right = TRUE
  ))
map_background_color(ht,
  by_ranges(
    c(1, 5),
    c("red", "yellow", "green"),
    right = FALSE
  ))
```

by_regex

Map cells matching a string or regex to cell properties

Description

by_regex() sets properties on cells which match a regular expression.

Usage

```r
by_regex(..., .grepl_args = list(), ignore_na = TRUE)
```

Arguments

- `...` A list of name-value pairs. The names are regular expressions. If there is a single unnamed argument, this is the default value for unmatched cells. More than one unnamed argument is an error.
- `.grepl_args` A list of arguments to pass to `grepl()`. Useful options include `fixed`, `perl` and `ignore.case`.
- `ignore_na` If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.
by_rows

Set cell properties by row or column

Description

by_rows and by_cols set properties in horizontal or vertical "stripes".

Usage

by_rows(..., from = 1, ignore_na = TRUE)

by_cols(..., from = 1, ignore_na = TRUE)

Arguments

... One or more cell property values.

from Numeric. Row or column to start at.

ignore_na If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.

Value

A function for use in map_*** functions.
by_values

See Also

mapping-functions

Other mapping functions: by_cases(), by_colors(), by_function(), by_quantiles(), by_ranges(), by_regex(), by_values()

Examples

ht <- as_hux(matrix(rnorm(25), 5, 5))
map_background_color(ht,
  by_rows("green", "grey"))
map_background_color(ht,
  by_cols("green", "grey"))

by_values Map specific cell values to cell properties

Description

Use by_values() to set properties for cells with specific, pre-determined contents.

Usage

by_values(..., ignore_na = TRUE)

Arguments

... Name-value pairs like name = value. Cells where contents are equal to name will have the property set to value. If there is a single unnamed argument, this is the default value for unmatched cells. More than one unnamed argument is an error.

ignore_na If TRUE, NA values in the result will be left unchanged from their previous values. Otherwise, NA normally resets to the default.

Value

A function for use in map_*** functions.

See Also

mapping-functions

Other mapping functions: by_cases(), by_colors(), by_function(), by_quantiles(), by_ranges(), by_regex(), by_rows()
Examples

```r
ht <- hux(letters[1:3])
map_background_color(ht,
    by_values(a = "red", c = "yellow"))
map_background_color(ht,
    by_values(a = "red", c = "yellow", "green"))
```

---

**caption**

*Set the table caption*

Description

By default, captions are displayed above the table. You can change this with `caption_pos()`.

Usage

```r
caption(ht)
caption(ht) <- value
set_caption(ht, value)
```

Arguments

- `ht` A huxtable.
- `value` A string. Set to `NA` to reset to the default, which is "NA".

Details

Captions are not escaped. See the example for a workaround.

Value

`caption()` returns the caption property. `set_caption()` returns the modified huxtable.

See Also

Other caption properties: `caption_pos()`, `caption_width()`

Examples

```r
set_caption(jams, "Pots of jam for sale")
# escape caption characters:
caption(jams) <- sanitize(
    "Make $$$ with jam",
    type = "latex")
```
caption_pos

Position the table’s caption

Description

If `caption_pos` is "top" or "bottom", then the horizontal position ("left", "center" or "right") will be determined by the huxtable’s `position()`.

Usage

```r
caption_pos(ht)
caption_pos(ht) <- value
set_caption_pos(ht, value)
```

Arguments

- **ht**: A huxtable.
- **value**: String: "top", "bottom", "topleft", "topcenter", "topright", "bottomleft", "bottomcenter" or "bottomright". Set to NA to reset to the default, which is "top".

Value

`caption_pos()` returns the `caption_pos` property. `set_caption_pos()` returns the modified huxtable.

See Also

Other caption properties: `caption_width()`, `caption()`

Examples

```r
caption_pos(jams) <- "topleft"
caption_pos(jams)

caption(jams) <- "Jam for sale"
jams
set_caption_pos(jams, "bottom")
```
caption_width  

Set the width of the table caption

Description

A numeric width is interpreted as a proportion of text width in LaTeX, or of width of the containing element in HTML. A character width must be a valid LaTeX or CSS dimension. The default, NA, makes the caption the same width as the table.

Usage

caption_width(ht)
caption_width(ht) <- value
set_caption_width(ht, value)

Arguments

ht  
A huxtable.

value  
Number or string. Set to NA to reset to the default, which is NA.

Value

caption_width() returns the caption_width property. set_caption_width() returns the modified huxtable.

See Also

Other caption properties: caption_pos(), caption()

Examples

caption_width(jams) <- 0.5
caption_width(jams)

---

cbind.huxtable  

Combine rows or columns

Description

These methods are called when one argument to cbind/rbind is a huxtable. As well as combining cell contents, they copy table, row, column and/or cell properties into the returned result.
Usage

## S3 method for class 'huxtable'
cbind(..., deparse.level = 1, copy_cell_props = TRUE)

## S3 method for class 'huxtable'
rbind(..., deparse.level = 1, copy_cell_props = TRUE)

Arguments

... Vectors, matrices, or huxtables.
deparse.level Unused.
copy_cell_props Cell properties to copy from neighbours (see below).

Details

Table properties will be taken from the first argument which is a huxtable. So will row properties
(for cbind) and column properties (for rbind).

If some of the inputs are not huxtables, and copy_cell_props is TRUE, then cell properties will be
copied to non-huxtables. Objects on the left or above get priority over those on the right or below.

If copy_cell_props is FALSE, cells from non-huxtable objects will get the default properties.

You cannot bind huxtables with data frames, since the R method dispatch will always call the data
frame method instead of the huxtable-specific code. For a solution, see add_columns().

Value

A huxtable.

Examples

sugar <- c("Sugar", "40\%", "35\%", "50\%")
jams <- set_bold(jams, 1, everywhere)
cbind(jams, sugar)
cbind(jams, sugar,
    copy_cell_props = FALSE)

jams <- set_text_color(jams,
    everywhere, 1, "red")
rbind(jams, c("Damson", 2.30))
rbind(jams, c("Damson", 2.30),
    copy_cell_props = FALSE)
column_to_header

Convert a column to header rows

Description

Convert a column to header rows

Usage

```
column_to_header(
  ht,  
  col,  
  ...,
  glue = "{value}"
  start_col = 1,
  ignore_headers = TRUE,
  set_headers = TRUE
)
```

Arguments

- `ht`: A huxtable.
- `col`: A column specifier for a single column.
- `...`: Properties to set on new rows
- `glue`: Glue string. "{value}" will be replaced by the column value.
- `start_col`: Integer. New header text will start at this column.
- `ignore_headers`: Logical. Ignore existing headers?
- `set_headers`: Logical. Set new rows as headers?

Examples

```
column_to_header(jams, "Type")
column_to_header(jams, "Type", text_color = "red")
column_to_header(jams, "Price",
  number_format = 2,
  italic = TRUE,
  glue = "Price: {value}")
```

```
iris_hux <- as_hux(iris[c(1:4, 51:54, 101:104),])
column_to_header(iris_hux, "Species",
  markdown = TRUE,
  glue = "Species: **{value}**"
)
**Description**

Numeric column widths are treated as proportions of the table width. Character widths must be valid CSS or LaTeX dimensions.

**Usage**

```r
col_width(ht)
col_width(ht) <- value
set_col_width(ht, col, value)
```

**Arguments**

- `ht` A huxtable.
- `col` A col specifier. See `rowspecs` for details.
- `value` Numeric or character vector. Set to `NA` to reset to the default, which is `NA`.

**Details**

In LaTeX, if you specify a column width, but set `wrap` to `FALSE` and have cells which overrun, then you may have problems with table position and with background colours in other cells. The workaround is to adjust the width, so that your cells no longer overrun.

**Value**

`col_width()` returns the `col_width` property. `set_col_width()` returns the modified huxtable.

**See Also**

Other table measurements: `height()`, `row_height()`, `width()`

**Examples**

```r
col_width(jams) <- c(.2, .8)
col_width(jams)
jams$Notes <- c("Notes",
            "This year's finest", "", "")
jams
set_col_width(jams, c(.4, .5, .1))
```
escape_contents

Escape or unescape text in cells

Description

Setting escape_contents to FALSE allows you to include raw HTML or TeX code in your cells.

Usage

```
escape_contents(ht)
escape_contents(ht) <- value
set_escape_contents(ht, row, col, value )
map_escape_contents(ht, row, col, fn)
```

Arguments

- `ht`: A huxtable.
- `row`: A row specifier. See `rowspecs` for details.
- `col`: An optional column specifier.
- `fn`: A mapping function. See `mapping-functions` for details.
- `value`: A logical vector or matrix.
  Set to NA to reset to the default, which is TRUE.

Details

If `markdown()` is TRUE for a cell, the escape_contents property will be ignored.

Value

- `escape_contents()` returns the escape_contents property.
- `set_escape_contents()` returns the modified huxtable.

See Also

- `sanitize()` for escaping text manually.

Examples

```r
ht <- huxtable(
  Text  = "x squared",
  Maths = "$x^2$"
)
ht <- set_escape_contents(ht, FALSE)
## Not run:
quick_pdf(ht)
```
final

Return the last n rows or columns

Description
This is a convenience function to use in row and column specifications. In that context, it returns the last n row or column numbers of the huxtable.

Usage
final(n = 1)

Arguments
n Number of rows to return.

Details
Technically, final returns a two-argument function - see rowspecs for more details.

Examples
set_bold(jams, final(2), final(1), TRUE)

fmt_percent
Format numbers as percent

Description
fmt_ functions are designed to work with number_format().

Usage
fmt_percent(digits = 1, format = "f", ...)

Arguments
digits How many digits to print.
format, ... Passed into formatC().

Value
An object you can pass into number_format().
See Also

Other format functions: `fmt_pretty()`

Examples

```r
jams$Sugar <- c("Sugar content", 0.4, 0.35, 0.45)
set_number_format(jams, -1, "Sugar", fmt_percent(1))
```

```r
fmt_pretty

Use prettyNum to format numbers

Description

Use prettyNum() to format numbers

Usage

```r
fmt_pretty(big.mark = ",", ..., scientific = FALSE)
```

Arguments

```r
big.mark, scientific, ...

Passed to prettyNum()
```

Value

An object you can pass into number_format().

See Also

Other format functions: `fmt_percent()`

Examples

```r
jams$Sales <- c("Sales", 35000, 55500, 20000)
set_number_format(jams, -1, "Sales", fmt_pretty())
```

font

Set the font for cell text

Description

Set the font for cell text

Usage

```r
font(ht)
fnt(ht) <- value
set_font(ht, row, col, value )
map_font(ht, row, col, fn)
```

Arguments

- `ht`: A huxtable.
- `row`: A row specifier. See `rowspecs` for details.
- `col`: An optional column specifier.
- `fn`: A mapping function. See `mapping-functions` for details.
- `value`: A character vector or matrix.
  Set to NA to reset to the default, which is "NA".

Details

To find out what fonts are on your system, `systemfonts::match_font()` is useful.

For HTML, you can use comma-separated lists of font names like "Times New Roman, Times, Serif". This is not portable, though.

LaTeX and HTML use different font names. To use the same font names across document formats, see `options("huxtable.latex_use_fontspec")` in `huxtable-options`.

Value

`font()` returns the font property. `set_font()` returns the modified huxtable.

See Also

Other formatting functions: `background_color()`, `bold()`, `font_size()`, `na_string()`, `number_format()`, `text_color()`
Examples

```r
font(jams) <- "times"
font(jams)

jams2 <- set_font(jams,
    "arial")
font(jams2)

jams3 <- set_font(jams,
    2:3, 1, "arial")
font(jams3)

jams4 <- map_font(jams,
    by_rows(
        "arial",
        "times")
)
font(jams4)
```

---

### font_size

**Make text larger or smaller**

**Description**

Font size is in points.

**Usage**

```r
font_size(ht)
font_size(ht) <- value
set_font_size(ht, row, col, value )
map_font_size(ht, row, col, fn)
```

**Arguments**

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.
- **fn**: A mapping function. See `mapping-functions` for details.
- **value**: A numeric vector.
  - Set to `NA` to reset to the default, which is `NA`.

**Value**

- `font_size()` returns the `font_size` property.
- `set_font_size()` returns the modified huxtable.
guess_knitr_output_format

See Also
Other formatting functions: background_color(), bold(), font(), na_string(), number_format(), text_color()

Examples

```r
font_size(jams) <- 14
font_size(jams)

jams2 <- set_font_size(jams, 12)
font_size(jams2)

jams3 <- set_font_size(jams, 2:3, 1, 12)
font_size(jams3)

jams4 <- map_font_size(jams,
  by_rows(12,
           14))
font_size(jams4)
```

Description
Convenience function which tries to guess the ultimate output from knitr and rmarkdown.

Usage
guess_knitr_output_format()

Value
"html", "latex", or something else. If we are not in a knitr document, returns an empty string.

Examples
```r
## Not run:
# in a knitr document
guess_knitr_output_format()

## End(Not run)
```
header_cols

Mark rows or columns as headers

Description

Arbitrary rows and columns can be headers: they do not have to be at the top or left of the table.

Usage

header_cols(ht)
header_cols(ht) <- value
set_header_cols(ht, col, value)

header_rows(ht)
header_rows(ht) <- value
set_header_rows(ht, row, value)

Arguments

ht A huxtable.
col A col specifier. See rowspecs for details.
value Logical vector. Set to NA to reset to the default, which is FALSE.
row A row specifier. See rowspecs for details.

Details

By default header rows and columns are not shown differently from other rows, but you can change this with style_headers(). Various themes may set properties on headers. Lastly, headers are treated differently when restacking.

Value

header_cols() returns the header_cols property. set_header_cols() returns the modified huxtable.

Examples

jams <- set_header_rows(jams, 1, TRUE)
jams <- set_header_cols(jams, 1, TRUE)
style_headers(jams,

  bold = TRUE,
  text_color = "purple"
)

)
**height**  

*Set the table height*

---

**Description**

`height()` sets the height of the entire table, while `[row_height()]` sets the height of individual rows. Height is measured as a proportion of the containing block (HTML) or `\text{theight}` (LaTeX). A character height must be a valid CSS or LaTeX dimension.

**Usage**

```r
height(ht)  
height(ht) <- value  
set_height(ht, value)
```

**Arguments**

- **ht**  
  A huxtable.

- **value**  
  A number or string. Set to `NA` to reset to the default, which is `NA`.

**Value**

`height()` returns the height property. `set_height()` returns the modified huxtable.

**See Also**

Other table measurements: `col_width()`, `row_height()`, `width()`

**Examples**

```r
height(jams) <- 0.4  
height(jams)
```

---

**huxreg**  

*Create a huxtable to display model output*

---

**Description**

Create a huxtable to display model output
Usage

huxreg(
  ..., 
  error_format = "((std.error))",
  error_pos = c("below", "same", "right"),
  number_format = "%.3f",
  align = ",",
  ci_level = NULL,
  tidy_args = NULL,
  glance_args = NULL,
  stars = c("***" = 0.001, "**" = 0.01, "*" = 0.05),
  bold_signif = NULL,
  borders = 0.4,
  outer_borders = 0.8,
  note = if (is.null(stars)) NULL else "{stars}",
  statistics = c(N = "nobs", R2 = "r.squared", "logLik", "AIC"),
  coefs = NULL,
  omit_coefs = NULL
)

Arguments

... Models, or a single list of models. Names will be used as column headings.
error_format How to display uncertainty in estimates. See below.
error_pos Display uncertainty "below", to the "right" of, or in the "same" cell as estimates.
number_format Format for numbering. See number_format() for details.
align Alignment for table cells. Set to a single character to align on this character.
ci_level Confidence level for intervals. Set to NULL to not calculate confidence intervals.
tidy_args List of arguments to pass to generics::tidy(). A list without names will be treated as a list of argument lists, one for each model.
glance_args List of arguments to pass to generics::glance(). A list without names will be treated as a list of argument lists, one for each model.
stars Levels for p value stars. Names of stars are symbols to use. Set to NULL to not show stars.
bold_signif Where p values are below this number, cells will be displayed in bold. Use NULL to turn off this behaviour.
borders Thickness of inner horizontal borders. Set to 0 for no borders.
outer_borders Thickness of outer (top and bottom) horizontal borders. Set to 0 for no borders.
note Footnote for bottom cell, which spans all columns. {stars} will be replaced by a note about significance stars. Set to NULL for no footnote.
statistics A vector of summary statistics to display. Set to NULL to show all available statistics. To change display names, name the statistics vector: c("Displayed title" = "statistic_name", ...)


coefs A vector of coefficients to display. Overrules `omit_coefs`. To change display names, name the coef vector: `c("Displayed title" = "coefficient_name", ...)

`omit_coefs` Omit these coefficients.

Details
Models must have a `generics::tidy()` method defined, which should return "term", "estimate", "std.error", "statistic" and "p.value". The "broom" package provides methods for many model objects. If the tidy method does not have a `conf.int` option, `huxreg` will calculate confidence intervals itself, using a normal approximation.

If `...` has names or contains a single named list, the names will be used for column headings. Otherwise column headings will be automatically created.

If the coef and/or statistics vectors have names, these will be used for row headings. If different values of coef have the same name, the corresponding rows will be merged in the output.

Statistics should be column names from `generics::glance()`. You can also use "nobs" for the number of observations. If `statistics` is `NULL` then all columns from glance will be used. To use no columns, set `statistics = character(0)`.

`error_format` is a string to be interpreted by `glue::glue()`. Terms in parentheses will be replaced by computed values. You can use any columns returned by `tidy`: typical columns include statistic, p.value, std.error, as well as conf.low and conf.high if you have set `ci_level`. For example, to show confidence intervals, you could write `error_format = "{conf.low} to {conf.high}"`.

Value
A huxtable object.

Fixing p values manually
If you wish to use e.g. robust standard errors, you can pass results from e.g. `lmtest::coeftest()` into `huxreg`, since these objects have tidy methods. Alternatively, to manually insert your own statistics, see `tidy_override()`.

Examples
```r
if (! requireNamespace("broom")) {
  stop("Please install 'broom' to run this example.")
}

lm1 <- lm(mpg ~ cyl, mtcars)
lm2 <- lm(mpg ~ cyl + hp, mtcars)
glm1 <- glm(I(mpg > 20) ~ cyl, mtcars,
  family = binomial)

huxreg(lm1, lm2, glm1)
```

```r
if (requireNamespace("sandwich") && requireNamespace("lmtest")) {
```
lm_robust <- lmtest::coeftest(lm1,
  vcov = sandwich::vcovHC)
# coeftest() has no "glance" method:
huxreg(lm_robust,
  statistics = character(0))

---

**huxtable**  
*Create a huxtable*

**Description**

huxtable, or hux, creates a huxtable object.

**Usage**

```r
huxtable(
  ..., 
  add_colnames = getOption("huxtable.add_colnames", TRUE),
  add_rownames = FALSE,
  autoformat = getOption("huxtable.autoformat", TRUE)
)
```

```r
hux(
  ..., 
  add_colnames = getOption("huxtable.add_colnames", TRUE),
  add_rownames = FALSE,
  autoformat = getOption("huxtable.autoformat", TRUE)
)
```

```r
tribble_hux(
  ..., 
  add_colnames = getOption("huxtable.add_colnames", TRUE),
  autoformat = getOption("huxtable.autoformat", TRUE)
)
```

**Arguments**

- `...`: For huxtable, named list of values as in `data.frame()`. For tribble_hux, data values as in `tibble::tribble()`.
- `add_colnames`: If TRUE, add a first row of column names to the huxtable.
- `add_rownames`: If TRUE or a character string, add a first column of row names to the huxtable. The string gives the name for the new column (or "rownames" for TRUE).
- `autoformat`: If TRUE, automatically format columns by type. See below.
Details

If you use add_colnames or add_rownames, be aware that these will shift your rows and columns along by one: your old row/column 1 will now be row/column 2, etc.

add_colnames defaults to TRUE. You can set the default globally by setting `options("huxtable.add_colnames")` to TRUE or FALSE.

tribble_hux is a simple wrapper around `tibble::tribble()` which lets you create data in a readable format. It requires the "tibble" package to be installed.

Value

An object of class `huxtable`.

Automatic formatting

If `autoformat` is TRUE, then columns will have `number_format()` and `align()` properties set automatically, as follows:

• Integer columns will have `number_format` set to 0.
• Other numeric columns will have `number_format` set to "%.3g".
• All other columns will have `number_format` set to NA (no formatting).
• Integer, Date and date-time (i.e. POSIXct and POSIXlt) columns will be right-aligned.
• Other numeric columns will be aligned on `options("OutDec")`, usually ".".
• Other columns will be left aligned.

You can change these defaults by editing `options("huxtable.autoformat_number_format")` and `options("huxtable.autoformat_align")`. See `huxtable-package` for more details.

Automatic alignment also applies to column headers if `add_colnames` is TRUE; headers of columns aligned on a decimal point will be right-aligned. Automatic number formatting does not apply to column headers.

See Also

`huxtable-options`

Examples

```r
ht <- huxtable(
  column1 = 1:5,
  column2 = letters[1:5]
)
ht
tribble_hux(
  ~ Name, ~ Salary,
  "John Smith", 50000,
  "Jane Doe", 50000,
  "David Hugh-Jones", 50000,
  add_colnames = TRUE
)
```
huxtable-FAQ

Frequently Asked Questions, including how to get help

Description

A FAQ of common issues.

Details

• I get a LaTeX error when I try to compile my document!
  Have you installed the LaTeX packages you need? LaTeX packages are different from R
  packages. Run `check_latex_dependencies()` to find out if you are missing any. Then install
  them using your system’s LaTeX management application. Or you can try `install_latex_dependencies()`.
  In some rmarkdown and LaTeX formats, you also need to add LaTeX dependencies manually. Run `report_latex_dependencies()` and add the output to your LaTeX preamble, or in
  Rmarkdown formats, add it to the rmarkdown header like this:

  ```r
  header-includes:
    - \usepackage{array}
    - \usepackage{caption}
    ... et cetera
  ```

• Huxtable isn’t working in my Rmarkdown beamer_presentation slides.
  You may need to set the beamer "fragile" option, like this:

  ```r
  # Slide title {.fragile}
  ```

• Numbers in my cells look weird!
  You can change numeric formatting using `number_format()`. Base R options like `scipen`
  usually have no effect.

• I ran `caption(ht) <- "Something"` and got an error message:

  ```r
  Error in UseMethod("caption<-") :
  no applicable method for 'caption<-' applied to an object of class "c('huxtable', 'data.frame')"
  ```
  You may have loaded another package with a caption method, e.g. "xtable". Try loading
  huxtable after xtable.

• How can I get line breaks in my cells?
  Just insert a line break "\n" in the cell contents. Then make sure that `width()` is set and
  `wrap()` is TRUE (it is by default).

• How can I change the font size, font etc. of captions?
  There are no direct commands for this. You have to use raw HTML/TeX/other commands
  within the caption itself. For example to have a bold caption in HTML, you might do some-thing like:

  ```r
  set_caption(jams, "<b>Jam Prices</b>")
  ```
• How do I refer to tables in bookdown?
   As of version 4.3.0, this is handled automatically for you. Just set the label using `label()`,
   then in markdown text do e.g.:

   \@ref(tab:my-table-label).

• I called `library(huxtable)` and now my `data.table` objects are getting printed!
   Set `options(huxtable.knit_print_df = FALSE)`.

• How can I set a property on an arbitrary group of cells?
   If you can't use the `mapping-functions` interface, and you want to set a property for multiple
   cells that aren't all in the same rows and/or columns, you could use a little-known fact about
   R subsetting. If you subset `ht[x]` where `x` is two-column numeric matrix, then each row of `x`
   indexes a single `(row, column)` cell. So, for example, here's how to set the background color
   of cells `(2,1)`, `(1, 3)` and `(4, 2)` of a huxtable:

   ```r
   indices <- matrix(c(2, 1, 1, 3, 4, 2), ncol = 2, byrow = TRUE)
   background_color(jams)[indices] <- "orange"
   ```

   Another useful trick sets properties on the diagonal, using `diag()`:

   ```r
   diag(background_color(jams)) <- "grey"
   ```

• I have another problem.
   If you have a bug - i.e. there is something wrong with the software - or a feature request,
   please report it to [https://github.com/hughjonesd/huxtable/issues](https://github.com/hughjonesd/huxtable/issues). Otherwise, ask a
   question on StackOverflow or [https://community.rstudio.com](https://community.rstudio.com). That way, other people
   will benefit from the answers you get.

• Can I email you directly?
   I'd rather you asked on a public website. If you then email me a link, I may be able to help.

---

**huxtable-news**

**Changes to the huxtable package**

**Description**

This help page simply gives the contents of NEWS.md.

**huxtable 5.5.0**

- Huxtable should work with Quarto documents.
  - Quarto labels and captions will override huxtable-provided ones.
  - Quarto style references like `@table-label` only work with quarto labels.
  - Please report any bugs!
- New `column_to_header()` function converts a column to header rows. New `as_hux()` method for
  grouped_df objects optionally converts groups to header rows.
- New convenience functions `stripe_rows()` and `stripe_columns()`.
• Add format and ... options to fmt_percent() to allow flexible formatting via formatC().
• add_footnote() gets an explicit number_format argument which is NA by default.
• Bugfix: infinite loop with wide characters in to_screen().
• Bugfix: duplicate colnames when exporting huxreg(..., error_pos = "right") to flextable.
• Bugfix: bookdown-style references weren’t working in blogdown.

huxtable 5.4.0

• New behaviour: setting colspan() or rowspan() overwrites the content of cells that have been shadowed.
  
  ht <- hux(c(1, 1), c(2, 2), c(3, 3))
  ht <- set_allBorders(ht)
  colspan(ht)[1, 1] <- 3

  # old behaviour
  ht[, c(2, 1, 3)]
  ## +--------------------------+
  ## | 2 | 1 | 3 |
  ## +--------+--------+--------+
  # new behaviour
  ht[, c(2, 1, 3)]
  ## +--------------------------+
  ## | 1 | 2 | 1 | 3 |
  ## +--------+--------+--------+

  • New option huxtable.latex_siunitx_align allows you to use the LaTeX siunitx package to handle decimal point alignment. This is FALSE by default.
  • Bugfix: centre alignment was not working in print_screen().
  • Bugfix: failure in to_md() with recent versions of stringi package.
  • Bugfix: repeating a single row in a subset, like ht[c(1, 1, 2, 3), ], was setting colspan = 2 on the repeated row.
  • Bugfix: zero-argument subset replacement like ht[] <- ... wasn’t working.

huxtable 5.3.0

• Improve decimal alignment in LaTeX when align(ht) == ".". This may change the appearance of some documents.
• Allow tidy_override() to extend columns of tidy and glance.
• Bugfix: #196 ^ was giving errors in LaTeX.
huxtable 5.2.0

- Add table_environment property so you can use e.g. "table*" in TeX.
- Bugfix: print_screen(h, colnames = FALSE) didn’t print a final newline.
- Bugfix: italic from markdown was being printed as underlined in TeX.
- Minor test update for compatibility with broom.

huxtable 5.1.1

- Minor test update for compatibility with broom.
- Fixes for R 4.1.0.

huxtable 5.1.0

- as_flextable() now exports markdown in cells to RTF, and to Word with the help of the optional ftExtra package. Thanks @atusy for adding this feature.
- Improvements to markdown screen export. This now uses the optional fansi package.
- New feature: as_Workbook() gains start_row and start_col arguments, to write a huxtable into an Excel worksheet starting at a particular row or column.
- New feature: huxreg() gains a glance_args argument to pass arguments to glance().
- New feature: options(huxtable.long_minus = TRUE) will try to use long minus signs before numbers. The default is FALSE. It will probably become TRUE in a future version.
- Bugfix: insert_row/column(..., after = 0) was unsetting table properties.
- Bugfix: unicode characters above 32767 were incorrectly represented in RTF. Thanks @kaigu1990.
- Bugfix: columns were being collapsed in as_Workbook().
- Bugfix: style_cells didn’t work unless huxtable was on the search path.
- Bugfix: merge_repeated_rows merged NA rows incorrectly.
- Bugfix: number format was not set correctly in huxreg()’s note.
- Bugfix: in huxreg(), tidy_args threw an error if the first argument to tidy() was a named list.
- Bugfix: tidy_replace() was broken.
- Clearer error messages for tidy_override() when extend = FALSE. In future, extend will probably default to TRUE.

Other news:

- Huxtable received its first Patreon sponsor! Thanks to Ross Mattheis.

huxtable 5.0.0

Huxtable 5.0.0 brings numerous changes. For a more user-friendly introduction, see https://hughjonesd.github.io/whats-new-in-huxtable-5.0.0.html.

Breaking changes:

- There are changes to LaTeX output.
- LaTeX \tabcolsep is now set to 0 within huxtable tables, while left and right padding should now take effect even when wrap is FALSE.
- The default LaTeX table environment is now “tabular” unless width is set. If width is set, it is “tabularx”.
- wrap only matters if width is set. Otherwise, cell wrapping is off.
- the \centerbox macro from the LaTeX “adjustbox” package is used to centre tables. This should improve centring when tables are too wide. You may need to update the LaTeX “adjustbox” package to a recent version. check_latex_dependencies() can inform you about this.

- As previously signalled, add_colnames has now become TRUE by default in huxtable() and as_huxtable(). Set options(huxtable.add_colnames = FALSE) to go back to the old behaviour.
- Newlines in cell contents are now respected (in LaTeX, so long as wrap = TRUE and width has been set).
- Huxtable borders have been reworked, fixing some longstanding bugs and adding new features.
  - Borders are now automatically collapsed. For example:
    ```r
    jams %>%
      set_right_border(everywhere, 1, 1) %>%
      set_left_border(everywhere, 2, 0.4)
    ```
    will set the border in between the columns of jams to 0.4, overwriting the previous value. This is more in line with what you would expect. For example, the following code now does what you probably want:
    ```r
    jams %>%
      set_rowspan(2, 1, 3) %>%
      set_bottom_border(4, everywhere, 1)
    ```
    Instead of the old behaviour:
    ```r
    jams %>%
      set_rowspan(2, 1, 3) %>%
      set_bottom_border(4, everywhere, 1)
    ```
    ```r
    Type  Price
    Strawberry  1.90
    2.10
    1.80
    -----------------------------
    ```
  - set_left_border(), set_all_borders() and friends all use a default value of 0.4. So to set a default border, write e.g.
    ```r
    as_hux(head(iris)) %>%
    set_bottom_border(1, everywhere)
    ```
  - A new brdr() class encapsulates border thickness, style and colour. You can set all properties at once by writing, e.g.
as_hux(jams) %>%
  set_bottom_border(1, everywhere, brdr(1, "dotted", "darkgreen"))
left_border(ht) and friends return a brdr object. To access the border thickness, write
brdr_thickness(left_border(ht)).

• Various deprecated items have been removed:
  – The 3-argument form of set_* instead, use map_*.
  – The byrow argument to set_* Instead, use map_ and by_cols().
  – error_style and pad_decimal arguments in huxreg. Use error_format and align(hx)
    <- ".".
  – The where(), is_a_number() and pad_decimal() functions. Use map_* functions, !
    is.na(as.numeric(x)), and align(ht) <- ".".

• Default padding has been increased to 6 points.
• By default, width() is now unset.
• By default, wrap() is now TRUE.
• every() has been renamed to stripe(), to avoid a clash with purrr::every(). everywhere,
  evens and odds are still the same.
• The little-used ability to set copy_cell_props to a character vector in rbind.huxtable and
  cbind.huxtable has been removed. You can still set it to FALSE.
• add_rows() and add_columns() now always call rbind.huxtable or cbind.huxtable() and
  return a huxtable.
• Huxtable no longer supports dplyr versions less than 0.7.0 (released mid-2017).
• set_cell_properties() has been renamed style_cells(). It is retained as a soft-deprecated
  alias.
• Various themes have been tweaked:
  – theme_basic() now has bold headers and no header column by default.
  – theme_plain() defaults to position = "centre".
  – theme_striped() uses grey stripes, a white border, and subtler headers.
  – theme_article() has thinner borders.

Other changes:
• You can now use markdown within table cells.
  – Use set_markdown(ht, rows, cols) to turn this on.
  – Or use the convenience function set_markdown_contents() to set cell contents that
    will be interpreted as markdown.
  – Markdown works for HTML and LaTeX. There’s basic support for on-screen display.
• Huxtable now has the concept of header row and columns.
  – By default, data frame column names will be headers.
  – To set other rows to be headers, use set_header_rows(ht, row_numbers, TRUE). For
    columns, use header_cols() or set_header_cols().
  – New functions style_headers(), style_header_cols(), and style_header_rows() to
    set multiple properties on headers.
  – In themes, header_row/col = TRUE set the first row/col to a header, and style all header
    rows/cols.
• set_bold() and set_italic() now use a default value of TRUE. So you can write e.g.
as_hux(head(iris)) %>%
  set_bold(1, everywhere)
• Console output in R now shows table position and caption position.
• By default, huxtable now sets labels from the current knitr chunk label, if there is one. This is consistent with kable(). In bookdown, you can then do e.g.

Some iris species are shown in @ref(tab:mytable):
```
r as_hux(iris)
```
Set options(huxtable.autolabel = FALSE) to turn off this behaviour.
• The one-argument form of [ now works for huxtables just as it does for data frames. For example, ht[2:3] selects columns 2 and 3.
• New functions fmt_percent() and fmt_pretty() for passing into number_format():

```r
jams$Sugar <- c("Sugar content", 0.4, 0.35, 0.45)
set_number_format(jams, -1, "Sugar", fmt_percent(1))
```
• split_across() and split_down() split a huxtable into a list of sub-tables. Headers can be automatically included.
• restack_across() and restack_down() split a huxtable, then join it back up. This is useful for making a table fit on a page.
• merge_across() and merge_down() merge an area of cells horizontally across rows, or vertically down columns.
• New functions set_lr_borders()/_border_colors()/_border_styles()/_padding() set left and right borders and padding simultaneously. New functions set_tb_borders() etc. set top and bottom properties simultaneously. There are map_ equivalents of all of these.
• set_outer_padding() sets padding around a range of cells, similarly to set_outer_borders().
• A new table-level property, caption_width(), allows you to set the width of the caption. The default, NA, sets the width equal to the table width.
• There are two new themes: theme_compact() and theme_bright().
• For huxreg(), a new function tidy_replace() allows you to replace the output of tidy(x) entirely.
• huxtable now only sets options(huxtable.knit_print_df = TRUE) if it is attached, not if it is loaded.
• huxtable supports dplyr::relocate(), new in dplyr 1.0.0.
• Improvements to as_flextable().
• Improvements to quick_pptx() (thanks @davidgohel).
• Bugfixes for options(huxtable.use_fontspec = TRUE).
• Bugfix: add_rownames = "string" now works as promised.
• Bugfix: non-ASCII characters are now supported in RTF.

Other news:
• New versions of the gtsummary package will have an as_huxtable() method.
• Package texreg on CRAN includes a huxtablereg() function for creating a table of regression outputs.
huxtable 4.7.1

- The `expss` package now supports export to huxtables.
- `by_quantiles()`, `by_equal_groups()` and `by_colors_space()` have gained a `colwise` argument, which calculates quantiles or colors separately for each column.
- Add caption support for `as_flextable()` (thanks @sjewo).

huxtable 4.7.0

- Better error messages.
- New `merge_repeated_rows()` function: merge repeated rows into a single cell.
- New `fill` and `colspan/rowspan` arguments for `insert_row()/insert_column()`:
  - `insert_row(ht, "blah", "", "", "", "", ...)` can be written `insert_row(ht, "blah", fill = "")`.
  - `colspan/rowspan` set `colspan/rowspan` of the first cell in the inserted row/column.

huxtable 4.6.1

- Bugfix: right borders in wrong place when cells were merged.
- Bugfix: chinese characters were displaying wrongly in `to_screen()`.

huxtable 4.6.0

- Set `options('huxtable.latex_use_fontspec')` to TRUE to use portable font names in TeX documents, with the LaTeX “fontspec” package.
- Bugfix: attributes were being copied wrongly in subset assignment of huxtables.
- Bugfix: text colors in `hux_logo()`.
- Bugfix: rbind of huxtable and matrix wasn’t setting `row_height` correctly.

huxtable 4.5.0

- Add `quick_latex()` function.
- The `texreg` package now includes a `huxtablereg` function, analogous to `huxreg`, which outputs a huxtable from a list of regressions. This will be available from the next version of `texreg`.

huxtable 4.4.0

- Huxtables can now be printed directly in Word documents and Powerpoint presentations, thanks to the `flextable` package and recent versions of Pandoc. (Powerpoint printing requires Pandoc >= 2.4.0.)
- New “wrapleft” and “wrapright” options to `position()` allow text wrapping around tables.
- New `set_outer_border_colors()` and `set_outer_border_styles()` functions, like `set_outer_borders()`.
- Huxtable no longer requires the `broom` package, instead using the `generics` package. If you use `huxreg()`, you will still need e.g. `broom` or `broom.mixed` to provide `tidy()` and `glance()` methods for specific models.
• Bugfix: tidy.tidy_override() and glance.tidy_override() should work even if underlying object has no tidy() or glance() method.
• Bugfix: huxtables had option clash when echo = TRUE in Rmd pdf_document format.
• Bugfix: caption() and height() weren’t playing nicely.
• Bugfix: mutate(..., copy_cell_props = FALSE) was adding a column named copy_cell_props.
• Bugfix: check_latex_dependencies and install_latex_dependencies gave misleading errors.
• Enhancement: when stars is NULL in huxreg, don’t print a note by default.
• Enhancement: use tinytex when available, allowing autoinstallation of latex packages.

huxtable 4.3.0
• More work on TeX. Tables should now compile when raw_attributes is not set.
• New map_xxx functions to set properties variably by cell values.
• Functions for mapping properties variably: by_rows, by_values, by_ranges, by_quantiles etc.
• Correct bookdown labels are now automatically created.
• New grey, blue, green and orange themes.
• New “themes” vignette.
• New tidy_override function to override p values etc. in huxreg.
• New set_contents function to change huxtable contents within dplyr pipes.
• Enhancement: left- and right-aligned captions are now set above the table in LaTeX, using the “threeparttable” package. You will need to install this using e.g. install_latex_dependencies() or tlmgr if it is not already on your system.
• Enhancement: in huxtable() and friends, add_rownames = "Colname" now sets the name for the new column.
• Improvements to the vignettes and help files.
• Bugfix: to_md could hang with bold/italic cells.

Deprecated:
• The 3 argument form of set_xxx functions is deprecated, as is the where function. Use map_xxx instead.
• Argument byrow is soft-deprecated. Use by_cols() instead.

huxtable 4.2.1
• Bugfix: wrap=TRUE caused squeezed text in RTF.

Important:
• TeX code was getting escaped by pandoc. To avoid this, if possible, huxtable now adds fenced code blocks round latex tables (see https://pandoc.org/MANUAL.html#extension-raw_attribute). You must add md_extensions: +raw_attribute to your YAML header for this to work, and you will need a recent (> 2.0.0) version of Pandoc.
huxtable 4.2.0

- More speedups: LaTeX 2-3x faster, as_Workbook 2-3x faster.
- Simplify LaTeX output using our own LaTeX commands.
- RTF support: new print_rtf, to_rtf and quick_rtf functions.
- New border_style properties to set “solid”, “double”, “dotted” or “dashed” borders. (At present, LaTeX only allows “solid” or “double”.)
- New merge_cells function, an alternative interface to colspan and rowspan.
- New quick_pptx function to print data frames and huxtables into Powerpoint.
- New install_latex_dependencies and check_latex_dependencies utility functions.
- add_rows and add_columns now accept data frames as arguments.
- New theme_mondrian theme :-D
- Enhancement: print_md now handles bold and italic cells.
- Enhancement: quick_pdf has new width and height options to change paper size.
- Use CSS writing-mode where possible for text rotation. Note that this may break on non-LTR languages. If this affects you, please file an issue.
- Bugfix: LaTeX didn’t compile when height and caption were both set.
- Bugfix: print_screen and print_md would hang with a wide huxtable.
- Tweaks to documentation.

huxtable 4.1.0

- dplyr, knitr, rmarkdown and some other packages have moved to “Suggests:”, lowering the dependency load considerably. All the functionality is still present. huxtable gives an informative warning if a needed package is not installed.
- Code rewrites for better performance and maintainability: HTML is up to 10x faster, LaTeX is up to 4x faster.
- Documentation improvements.
- New tribble_hux function wrapping tibble::tribble() for readable data input.
- New add_rows and add_columns functions to insert one or more rows into the middle of a huxtable.
- New option “huxtable.knitr_output_format” to override the default output format in knitr documents.
- Numeric row heights and column widths are rescaled to 1 when huxtables are cbinded/rbinded.
- LaTeX: at points where borders cross, priority is given to the horizontal border color.
- Bugfix: property accessors had the wrong environment. Thanks to Iñaki Úcar.
- Bugfix: row heights and column widths weren’t being copied with cbind/rbind.
- Bugfixes for 0-row or 0-column huxtables:
  - Output works, usually with a warning.
  - cbind and rbind work.
- Bugfix: HTML cols were printed with ‘width: NA’.
• Bugfix: width, col_width etc. can be reset to a number after setting them to a string.
  – The (undocumented) ability to mix numeric and non-numeric values for padding and/border
    widths has been removed. If you want a number, set a number and not a string.
• Bugfix: HTML tables with position “right” weren’t right-aligned.
• Nicer error messages when rbinding objects with different numbers of rows.
• Vignette improvements.
• is_a_number is deprecated.
• ... and a cool new randomized hux_logo() ;-

huxtable 4.0.1
• Improved formatting in Excel output.
• New format method which returns the result of to_html, to_latex etc. as appropriate.
• Bugfix: to_html printing e.g. “left-border: NA;” in cell CSS.
• Bugfix: set_all_* not working when huxtable is not attached.
• Bugfix: asWorkbook failing with non-numeric width.
• Bugfix: hux_logo was using multiple fonts, fails with Excel output.
• Bugfix: as_flextable borders not working in cells with colspan > 1.
• Documentation bugfixes.
• Compatibility with broom 5.0.0 - thanks @alexpghayes

huxtable 4.0.0
• New theme_plain theme.
• The default value for add_colnames is going to become TRUE. At present it remains FALSE. Set
  options("huxtable.add_colnames") to TRUE or FALSE to set the default and avoid warn-
  ings in future.
• quick_* functions now automatically open documents if used interactively. Use open = FALSE
to avoid.
• Tweak top and bottom margins for HTML tables.
• pad_decimal is deprecated in favour of align(ht) <- "".
• huxreg continues with a warning if statistics are unavailable for some models.

Breaking changes:
• huxtable now provides knit_print.data.frame methods. This means that bare data frames
  will be pretty-printed via huxtable if the package is loaded.
  – Set options("huxtable.knit_print_df") to FALSE if you don’t want this.
  – By default data frames are printed using the theme_plain theme. Set options("huxtable.knit_print_df_theme")
    to a different one-argument function if you want to use a different theme.
• The new autoformat argument lets huxtable() and as_huxtable() automatically choose
  alignment and number format based on column type. Set options("huxtable.autoformat")
  to FALSE to turn off this feature by default.
• The default value of `number_format` has changed from “%5.3g” to “%.3g”, which no longer space-pads numbers.

• `as_flextable` now does not print column names in the header. This matches the standard huxtable behaviour whereby headers are “just another row/column”. To get the old behaviour, use `colnames_to_header = TRUE`.

**Bugfixes:**

• Bugfix: Date and datetime columns were converted to numbers by `add_colnames`.

• LaTeX bugfix: background colors were printing an extra space.

• `huxreg` was never using built-in confidence intervals.

• Screen bugfixes:
  – set `max_width` to screen width (thanks @jacob-long)
  – misaligned decimal points

• Various bugfixes for `number_format`, `huxreg`, `as_hux.table`, `as_flextable`.

**huxtable 3.0.0**

• Output to Excel workbooks using the `openxlsx` package.

• New `quick_xlsx` function.

• `dplyr` select helpers now work inside `set_*` column specifications: e.g. `set_bold(ht, 1:3, matches("ab"), TRUE)`

• Column names can now be used for the `after` argument to `insert_column`.

• `quick_*` functions: when the `file` argument is not explicitly specified, confirm overwrites manually, or fail if called non-interactively.

• Add pointless quote marks in Description and Title... I don’t make the rules.

• Don’t apply `number_format` to negative exponents (e.g. `1.12e-3`).

• New `tidy_args` argument to `huxreg` allows per-model customization of the call to `tidy`.

**Breaking changes:**

• `quick_xxx` functions without an explicit `file` argument throw an error if called non-interactively, and prompt before overwriting files if called interactively.

**huxtable 2.0.2**

• Don’t apply `number_format` to exponents in scientific notation.

• Turn off some tests on CRAN, as they fail there but not elsewhere.

**huxtable 2.0.1**

• Fix `quick_pdf` error when moving output across filesystems.
huxtable 2.0.0

- New `quick_html`, `quick_pdf` and `quick_docx` functions to print table-like objects to a new document.
- `to_screen` only shows colnames if there are any non-zero-length column names.

Breaking changes:
- `number_format` now applies to any number-like substrings in cells. This means you can include e.g. significance stars in a cell and still use `number_format` to format the content.
- If `number_format` is NA, numbers are unchanged.
- Default value of `number_format` has changed from “%5.2f” to “%5.3g”, which plays nicer with integers but may surprise you by using scientific format for large numbers.

huxtable 1.2.0

- New `outerBorders` argument for `huxreg`. This changes default behaviour slightly.
- New border argument for `add_footnote` to choose width of footnote’s top border.
- Added guard assertions to many exported functions.
- Bugfix: captions and colnames are wrapped in `to_screen` to respect `max_width`.

huxtable 1.1.0

- No more ugly autocreated column names.
- Allow huxtable to have invalid or empty column names in general.
- LaTeX should now be much faster on large tables.
- `set_outer_borders` now accepts the same row/column arguments as other `set_` functions.
- Better handling in LaTeX of horizontal borders which don’t cross the entire table. (But not varying positive border widths. . . .)
- Bugfix: flextable didn’t like `huxreg`’s syntactically invalid column names.
- Accept, but silently change, English spelling of ‘centre’ in `align`, `position` and `caption_pos`.

huxtable 1.0.0

- LaTeX implements different thicknesses for vertical and horizontal borders (but only one horizontal thickness per row).
- LaTeX border colors now collapse nicely: set colors override unset ones.
- React gracefully to lack of p values in `huxreg`.
- New `set_outer_borders` function to set borders round a rectangle of cells.
- `to_screen` and `to_md` now respect `wrap` and `col_widths` properties.
- Screen and markdown wrap respect word boundaries.
- `to_screen` and `to_md` gain a `min_width` argument; `to_md` gains a logical `header` argument; `to_screen` gains a `compact` argument replacing `blank = NULL`.
- On screen colour and bold support, if the `crayon` package is installed. New `huxtable.color_screen` option.
• Move from ReporteRs to officer and flextable. No more RJava horror.
• New error_format argument to huxreg for flexible control over uncertainty estimates.
• Infrastructure improvements: slightly less ugly code in screen.R and LaTeX.R.

**Breaking changes:**
• Removed options collapse, borders, blank and colname_color from to_screen/print_screen.
• as_FlexTable is deprecated and calls as_flextable with a warning. header_rows and footer_rows arguments are ignored. If you need this feature, tell me.
• HTML border sizes are now set in points, not pixels.
• In huxreg:
  – ci_level is NULL by default. Set it to a number to calculate confidence intervals.
  – error_style is deprecated with a warning in favour of error_format.
  – Use {stars} not %stars% to display significance levels in the note argument.
  – borders becomes a number specifying border width. Set to 0 for no borders.

**huxtable 0.3.1**
• New convenience functions insert_row and insert_column.
• latex_float property allows you to change positioning in LaTeX.
• (Semantic versioning fail: this should have been 0.4.0.)

**huxtable 0.3.0**
• New borders argument for huxreg. gives borders in sensible places.
• Allow more flexible caption positioning with caption_pos.
• New set_default_properties function to set default properties for new huxtables.
• Fix compatibility with dplyr 0.6.0.

**huxtable 0.2.2**
• Fix a bug that could lead to wrong significance stars in huxreg.

**huxtable 0.2.1**
• Compatibility with dplyr 0.6.0.
• Use ~ for decimal padding in LaTeX.

**huxtable 0.2.0**
• New huxreg function to convert a list of models to a huxtable.
• New set_* interface allowing column ranges, expressions a la subset, and filling in values by row.
• Replacement methods $<-, [<- and [[<- now work better.
• New function set_cell_properties to set multiple properties on cells.
• evens, odds, everywhere, every(n, from), final(n), where(cond): convenience functions to select rows, columns and cells.
• Export to Word/Powerpoint via ReporteRs.
• Huxtable now supports dplyr verbs like filter and select.
• Exported function guess_knitr_output_format.
• Ability to set border colors.
• Prevent overlapping row/colspans.
• Expanded introduction and new vignette for huxreg.
• Numerous bugs have been fixed and replaced with new, more advanced bugs.

Breaking changes:
• theme_minimal has been renamed theme_basic to avoid a name clash with ggplot2.

huxtable 0.1.0
• Added a NEWS.md file to track changes to the package.
• First CRAN release.

---

### huxtable-options

<table>
<thead>
<tr>
<th>Package options</th>
</tr>
</thead>
</table>

<huxtable-options>

#### Description

Huxtable has several options.

#### Details

- options('huxtable.add_colnames') sets the default value for add_colnames in huxtable() and as_huxtable(). As of version 5.0.0, this defaults to TRUE.
- options('huxtable.print') sets the print method for huxtable objects. See print.huxtable().
- options('huxtable.knitr_output_format') overrides the default output format when huxtable objects are printed by knitr. Set to "html", "latex", "md" or "screen". If NULL (the default), huxtable guesses the format using guess_knitr_output_format().
- options('huxtable.autolabel'). If TRUE, (the default) automatically sets label() from the knitr chunk label, if there is one.
- options('huxtable.color_screen'). If TRUE and package crayon is available, huxtables will be printed in color on screen.
- options('huxtable.bookdown'). Set to TRUE within a bookdown document to automatically print bookdown-style labels. If unset, huxtable will try to guess if we are in a bookdown document.
- options('huxtable.knit_print_df'). If TRUE, data frames in knitr will be pretty-printed using huxtable. This option defaults to TRUE only if huxtable is attached to the search path using library(); not if huxtable is merely loaded (e.g. imported by another package).
• **options('huxtable.knit_print_df_theme').** A function applied to data frames before printing in knitr. The function should take one argument (a data frame) and return a huxtable. Defaults to *theme_plain()*.

• **options('huxtable.autoformat')** sets the default value for `autoformat` in `huxtable()` and `as_huxtable()`. It defaults to TRUE.

• **options('huxtable.latex_use_fontspec').** If TRUE, use the "fontspec" package, which allows you to use the same font names in TeX and HTML. This requires the xetex or xelatex engine, which can be set using an .rmd header option. Note that `quick_pdf()` may use pdflatex. The default is FALSE.

• **options('huxtable.long_minus').** If TRUE, prints long minus signs for numbers. The default is FALSE. In LaTeX output, this option is overridden by `options('huxtable.latex_siunitx_align')`.

• **options('huxtable.latex_siunitx_align').** If TRUE, uses the \tablenum macro from the "siunitx" package to align numbers when `align(ht)` is "." or similar. See `align()` for details. The default is FALSE.

• **options('huxtable.autoformat_number_format')** and **options('huxtable.autoformat_align')** are lists. The list names are base R classes. `huxtable()` with `autoformat = TRUE` will set `number_format()` and `align()` for data columns according to the corresponding list values. For example, to center-align Date objects you could set "huxtable.autoformat_align" to something like `list(\ldots, Date = "center", \ldots)`.

---

**hux_logo**

---

**Description**

Returns a randomized huxtable logo, inspired by Mondrian.

**Usage**

```
hux_logo(latex = FALSE, html = FALSE)
```

**Arguments**

- **latex**
  Style for LaTeX.

- **html**
  Style for HTML.

**Value**

The huxtable logo.

**Examples**

```
print_screen(hux_logo())
```
**Description**

These convenience functions wrap `cbind` or `rbind` for huxtables, to insert a single row or column.

**Usage**

```r
insert_column(
  ht,
  ..., after = 0, fill = NULL,
  colspan = 1, copy_cell_props = TRUE
)

insert_row(
  ht,
  ..., after = 0, fill = NULL,
  colspan = 1, copy_cell_props = TRUE
)
```

**Arguments**

- `ht` A huxtable.
- `...` Cell contents.
- `after` Insert the row/column after this position. 0 (the default) inserts as the first row/column.
- `fill` String. If `...` contains fewer elements than there are columns/rows to fill, the remaining cells will be filled with this.
- `rowspan, colspan` Scalar integer. Sets the rowspan or colspan of the first cell only. The default `NULL` throws an error if there are too few elements.
- `copy_cell_props` Copy cell properties from the previous row or column (if `after > 0`). See `cbind.huxtable()`.

**Details**

In `insert_column` only, you can use a column name for `after`.

Even if `colspan` or `rowspan` are greater than 1, you must still provide values for the hidden cells. Use `fill = ""` for this.
Value

The modified huxtable

See Also

`add_rows()` and `add_columns()`, which insert multiple rows/columns at once.

Examples

```r
insert_row(jams,
  c("Gooseberry", 2.15),
  after = 1
)

insert_column(jams,
  c("Sugar", "50\%", "60\%", "40\%"),
  after = "Price"
)

insert_column(jams,
  "Sugar",
  after = "Price",
  fill = "50\%"
)

# don't forget to use `fill`:
insert_row(jams,
  "Jams and prices",
  fill = "",
  colspan = 2
)
```

---

<table>
<thead>
<tr>
<th>jams</th>
<th>Prices of 3 jams</th>
</tr>
</thead>
</table>

Description

A huxtable of jams.

Usage

```r
jams
```

Format

A huxtable with 4 rows and 2 columns ("Type" and "Price").
knit_print.data.frame  *Print data frames in knitr using huxtable*

**Description**

Print data frames in knitr using huxtable

**Usage**

```r
knit_print.data.frame(x, options, ...)
```

**Arguments**

- `x` A huxtable.
- `options` Not used.
- `...` Not used.

**Details**

huxtable defines a knit_print method for data.frames. This converts the data frame to a huxtable, with `add_colnames = TRUE`, themes it using `theme_plain()` and prints it. It also tries to set a few intelligent defaults, e.g. wrapping long columns and setting an appropriate width. To turn this behaviour off, set `options(huxtable.knit_print_df = FALSE)`. To change the theme, set `options("huxtable.knit_print_df_theme")` to a one-argument function which should return the huxtable.

**See Also**

- `huxtable-options`
- Other knit_print: `knit_print.huxtable()`

**Examples**

```r
## Not run:
# in your knitr document
mytheme <- function (ht) {
  ht <- set_allBorders(ht, 0.4)
  ht <- set_allBorderColors(ht, "darkgreen")
  ht <- set_background_color(ht,
                              evens, odds, "salmon")
  ht
}

options(huxtable.knit_print_df_theme = mytheme)
# groovy!
data.frame(
```
## End(Not run)

---

**knit_print.huxtable**  
*Print a huxtable within knitr*

### Description

Print a huxtable within knitr

### Usage

```r
knit_print.huxtable(x, options, ...)
```

### Arguments

- **x**: A huxtable.
- **options**: Not used.
- **...**: Not used.

### Details

knitr calls `knitr::knit_print()` on objects when they are printed in a knitr (or RMarkdown) document. The method for huxtable objects guesses the appropriate output format and prints itself out appropriately. You can override the output format by setting `options("huxtable.knitr_output_format")`.

### See Also

- **huxtable-options**
- Other knit_print: `knit_print.data.frame()`

---

**label**  
*Set a table label for external referencing*

### Description

The label is used as the table’s label in LaTeX, and as the “id” property of the table element in HTML.

### Usage

```r
label(ht)
label(ht) <- value
set_label(ht, value)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ht</td>
<td>A huxtable.</td>
</tr>
<tr>
<td>value</td>
<td>A string. Set to \texttt{NA} to reset to the default, which is &quot;NA&quot;.</td>
</tr>
</tbody>
</table>

Details

LaTeX table labels typically start with "tab:".

Within knitr, huxtable labels will default to the same as the knitr chunk label. To turn off this behaviour, set \texttt{options(huxtable.autolabel = FALSE)}.

If you use \texttt{bookdown}, and set a label on your table, the table \texttt{caption()} will automatically be prefixed with \texttt{(#label)}. You can then refer to the table using \texttt{@ref(label)}. \texttt{label} needs to start with "tab:"; if it doesn’t, the "tab:" prefix will be added automatically. To turn off this behaviour, set \texttt{options(huxtable.bookdown = FALSE)}.

Value

\texttt{label()} returns the \texttt{label} property. \texttt{set_label()} returns the modified huxtable.

See Also

huxtable-options

Examples

```r
label(jams) <- "tab:mytable"
label(jams)
```

---

\textit{latex_float} \hspace{1cm} \textit{Set the position of the table float in LaTeX}

Description

Possible values include:

- "h": here
- "h!": definitely here
- "t": top of page
- "ht": here or at top of page
- "b": bottom of page
- "p": page of floats

Usage

\texttt{latex_float(ht)}
\texttt{latex_float(ht) <- value}
\texttt{set_latex_float(ht, value)}
mapping-functions

Arguments

- **ht**: A huxtable.
- **value**: A string. Set to `NA` to reset to the default, which is "ht".

Details

See LaTeX documentation for more details.

Value

latex_float() returns the latex_float property. set_latex_float() returns the modified huxtable.

Examples

```r
latex_float(jams) <- "b"
l latex_float(jams)
```

Description

This help page explains how to set properties differently for cells, depending on their contents. For example, in a table of p-values, you could bold cells where p < 0.05:

```r
map_bold(pval_hux, by_ranges(0.05, c(TRUE, FALSE)))
```

Or you can use red text for a particular value:

```r
hxtbl %>% map_text_color(by_values("Warning" = "red"))
```

There is a `map_...` function for each huxtable cell property. The syntax is:

```r
map_property(ht, row, col, fn)
```

where *property* is the property name.

*row* and *col* specify ranges of rows and columns. See `rowspecs` for details. To set properties for the whole table, omit `row` and `col`:

```r
map_property(ht, fn)
```

The *fn* argument is a *mapping function* which maps cell contents to property values.

- To set property values in "stripes" by rows or by columns, use `by_rows()` and `by_cols()`.
• To set property values for cells with specific contents, use \texttt{by_values()}.  
• To set property values for cells within a numeric range, use \texttt{by_ranges()}.  
• To set property values for cells by quantiles, use \texttt{by_quantiles()} or \texttt{by_equal_groups()}.  
• To set property values for cells that match a string or regular expression, use \texttt{by_regex()}.  
• To map numeric values to a colorspace, use \texttt{by_colorspace()}.  
• For a more general solution, use \texttt{by_function()} or \texttt{by_cases()}.  

\textbf{Caveat}

Most functions convert the huxtable to a matrix using \texttt{as.matrix()}. This can have unexpected results if you mix character and numeric data. See the example.

\textbf{Technical details}

\texttt{fn} takes four arguments: the entire original huxtable \texttt{ht}, a numeric vector of \texttt{rows}, a numeric vector of \texttt{cols}, and the current property values for \texttt{ht[rows, cols]}, as a matrix. It should return the new property values for \texttt{ht[rows, cols]}, as a matrix.

\textbf{Examples}

\begin{verbatim}
ht <- hux(Condition = c("OK", "Warning", "Error"))
ht <- map_text_color(ht, by_values(  
    OK     = "green",  
    Warning = "orange",  
    Error  = "red"  
))
ht

# Leaving NA values alone:
map_text_color(ht, by_values(  
    "OK" = "blue", NA, ignore_na = TRUE))

# Resetting values:
map_text_color(ht, by_values(  
    "OK" = "blue", NA, ignore_na = FALSE))

ht <- as_hux(matrix(rnorm(15), 5, 3))
map_background_color(ht, by_ranges(  
    c(-1, 1),  
    c("blue", "yellow", "red")  
))
map_background_color(ht,  
    by_equal_groups(2, c("red", "green")))

ht <- hux(  
    Coef = c(3.5, 2.4, 1.3),  
    Pval = c(0.04, 0.01, 0.07),  
    add_colnames = TRUE
)
map_bold(ht, everywhere, "Pval",
\end{verbatim}


```r
by_ranges(0.05, c(TRUE, FALSE))
```

# Problems with as.matrix:

```r
ht <- hux(c(-1, 1, 2), letters[1:3])
as.matrix(ht)  # look at the spaces...
as.matrix(ht) > 0  # uh oh
map_text_color(ht,
   by_cases(. < 0 ~ "red", TRUE ~ "blue"))
```

# To avoid this, only look at the truly numeric columns:

```r
map_text_color(ht, row = 1:3, col = 1,
   by_cases(. < 0 ~ "red", TRUE ~ "blue"))
```

---

**markdown**

**Interpret cell content as markdown**

**Description**

Cells where the markdown property is TRUE will be interpreted as markdown.

**Usage**

```r
markdown(ht)
markdown(ht) <- value
set_markdown(ht, row, col, value = TRUE)
map_markdown(ht, row, col, fn)
```

**Arguments**

- **ht**: A huxtable.
- **row**: A row specifier. See rowspecs for details.
- **col**: An optional column specifier.
- **fn**: A mapping function. See mapping-functions for details.
- **value**: A logical vector or matrix.
  Set to NA to reset to the default, which is FALSE.

**Details**

Markdown is currently implemented for HTML, Word, Powerpoint, RTF, LaTeX and on-screen display. Word requires the ftExtra package.

Most formats use commonmark, with the "strikethrough" extension enabled.

The following features are intended to work:

- bold and italic text
- strikethrough (write ~~text~~ to strike through text).
merge_across

- hyperlinks

There are some quirks:

- Paragraph-level properties (e.g. lists) won’t work in Word.
- Strikethrough will probably not work in Word.
- To make lists work in LaTeX, set width() and ensure wrap() is TRUE.
- Inline images in RTF work using the INCLUDEPICTURE field type.

If you try to use markdown tables within a table cell, then seek psychiatric help.

Value

markdown() returns the markdown property. set_markdown() returns the modified huxtable.

Note

Markdown content in cells is completely separate from printing the whole table as markdown using print_md(). When you set markdown to TRUE, huxtable itself interprets the cell contents as markdown, and spits out HTML, TeX or whatever.

See Also

set_markdown_contents(), a shortcut function.

Examples

jams[3, 2] <- "~2.10~ **Sale!** 1.50"
set_markdown(jams, 3, 2)

merge_across Merge cells across rows or down columns

Description

merge_across() creates multicolumn cells within each row. merge_down() creates multirow cells within each column.

Usage

merge_across(ht, row, col)
merge_down(ht, row, col)
merge_cells

Arguments

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.

Value

The `ht` object.

Cell content

In merged cell ranges, only the top left cell’s content is displayed. In addition, when you merge cells (either by setting `colspan()` or `rowspan()`, or using `merge_cells()` and friends) the content of the top left cell is copied to other cells. This prevents unexpected changes to content if you reorder or subset rows and columns.

See Also

Other cell merging: `merge_cells()`, `merge_repeated_rows()`

Examples

```r
ht <- as_hux(matrix(1:12, 4, 3, byrow = TRUE))
ht <- set_all_borders(ht, 1)
merge_across(ht, 2:4, 2:3)
merge_down(ht, 2:4, 2:3)
```

---

merge_cells Merge a range of cells

Description

merge_cells() merges a rectangle of cells into a single displayed cell, by setting `colspan()` and `rowspan()`.

Usage

merge_cells(ht, row, col)

Arguments

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.
merge_repeated_rows

Details
merge_cells(ht, c(min_row, max_row), c(min_col, max_col)) is equivalent to

colspan(ht)[min_row, min_col] <- max_col - min_col + 1
rowspan(ht)[min_row, min_col] <- max_row - min_row + 1

Value
The ht object.

Cell content
In merged cell ranges, only the top left cell’s content is displayed. In addition, when you merge cells (either by setting colspan() or rowspan(), or using merge_cells() and friends) the content of the top left cell is copied to other cells. This prevents unexpected changes to content if you reorder or subset rows and columns.

See Also
Other cell merging: merge_across(), merge_repeated_rows()

Examples

ht <- hux(a = 1:3, b = 1:3)
ht <- set_all_borders(ht, 1)
merge_cells(ht, 2:3, 1:2)

merge_repeated_rows Merge repeated rows into multirow cells

Description
merge_repeated_rows() looks within each column for contiguous groups of identical cells. These are merged by setting rowspan(). Doing this helps remove redundant information from the table.

Usage
merge_repeated_rows(ht, row, col)

Arguments
ht A huxtable.
row A row specifier. See rowspecs for details.
col An optional column specifier.
mutate.huxtable

Details

If row contains gaps, results may be unexpected (and a warning is given).

Value

The `ht` object.

Cell content

In merged cell ranges, only the top left cell’s content is displayed. In addition, when you merge cells (either by setting `colspan()` or `rowspan()`, or using `merge_cells()` and friends) the content of the top left cell is copied to other cells. This prevents unexpected changes to content if you reorder or subset rows and columns.

See Also

Other cell merging: `merge_across()`, `merge_cells()`

Examples

```r
ht <- as_hux(jams[c(1, 2, 2, 3, 3, 4), ])
ht <- add_columns(ht, c("Sugar", "30%", "40%", "30%", "40%", "30%"),
                    after = 1)
ht
merge_repeated_rows(ht)
merge_repeated_rows(ht, everywhere, "Type")
```

Description

Huxtable can be used with dplyr verbs `dplyr::select()`, `dplyr::rename()`, `dplyr::relocate()`, `dplyr::slice()`, `dplyr::arrange()`, `dplyr::mutate()` and `dplyr::transmute()`. These will return huxtables. Other verbs like `dplyr::summarise()` will simply return data frames as normal; `dplyr::pull()` will return a vector. `mutate` has an extra option, detailed below.

Usage

```r
mutate.huxtable(.data, ..., copy_cell_props = TRUE)
```

Arguments

- `.data` A huxtable.
- `...` Arguments passed to `dplyr::mutate()`.
- `copy_cell_props` Logical: copy cell and column properties from existing columns.
Details

If `mutate` creates new columns, and the argument `copy_cell_props` is missing or `TRUE`, then cell and column properties will be copied from existing columns to their left, if there are any. Otherwise, they will be the standard defaults. Row and table properties, and properties of cells in existing columns, remain unchanged.

Examples

```r
ht <- hux(a = 1:5, b = 1:5, c = 1:5, d = 1:5, add_colnames = FALSE)
bold(ht)[c(1, 3), ] <- TRUE
bold(ht)[, 1] <- TRUE
ht2 <- dplyr::select(ht, b:c)
ht2
bold(ht2)
ht3 <- dplyr::mutate(ht, x = a + b)
ht3
bold(ht3)
ht4 <- dplyr::mutate(ht, x = a + b,
                      copy_cell_props = FALSE)
bold(ht4)
```

---

### na_string

*Change how NA values are printed*

**Description**

NA values in the huxtable are printed as the value of `na_string`.

**Usage**

```r
na_string(ht)
na_string(ht) <- value
set_na_string(ht, row, col, value )
map_na_string(ht, row, col, fn)
```

**Arguments**

- **ht**
  A huxtable.
- **row**
  A row specifier. See `rowspecs` for details.
- **col**
  An optional column specifier.
- **fn**
  A mapping function. See `mapping-functions` for details.
- **value**
  A character vector or matrix.
  Set to `NA` to reset to the default, which is `""`.

**Value**

`na_string()` returns the `na_string` property. `set_na_string()` returns the modified huxtable.
See Also

Other formatting functions: `background_color()`, `bold()`, `font_size()`, `font()`, `number_format()`, `text_color()`

Examples

```r
ejams[3, 2] <- NA
ejams
set_na_string(jams, "---")
```

---

**number_format**

Set how numbers are formatted in cells

---

Description

If `number_format` is:

- numeric, numbers will be rounded to that many decimal places;
- character, it will be used as an argument to `sprintf()`;
- a function, the function will be applied to the numbers;
- `NA`, then numbers will not be formatted (except by conversion with `as.character`).

Usage

```r
number_format(ht)
number_format(ht) <- value
set_number_format(ht, row, col, value )
map_number_format(ht, row, col, fn)
```

Arguments

- `ht` A huxtable.
- `row` A row specifier. See `rowspecs` for details.
- `col` An optional column specifier.
- `fn` A mapping function. See `mapping-functions` for details.
- `value` A character or integer vector.

Note that setting to `NA` does not reset to the default.
Details

Number formatting is applied to any parts of cells that look like numbers. The exception is exponents in scientific notation; huxtable attempts to detect and ignore these.

The default value is "%.3g", which rounds numbers if they have more than 3 significant digits, and which may use scientific notation for large numbers.

Note that if your cells are of type numeric, a number format of NA doesn’t guarantee you get back what you typed in, since R’s default conversion may apply scientific notation and rounding.

To set number_format to a function, enclose the function in list. The function should take one argument and return a string. fmt_pretty() and fmt_percent() are useful shortcuts for common formatting functions.

Value

number_format() returns the number_format property. set_number_format() returns the modified huxtable.

See Also

fmt_pretty() and fmt_percent(). options("huxtable.long_minus") in huxtable-options for pretty-printing minus signs.

Other formatting functions: background_color(). bold(). font_size(). font(). na_string(). text_color()

Examples

ht <- huxtable(
  number_format = c(
    "Default",
    "NA",
    "2",
    "\%5.2f\",
    "Pretty",
    "Sign"
  ),
  a = rep(1000, 6),
  b = rep(1000.005, 6),
  c = rep(0.0001, 6),
  d = rep(-1, 6),
  e = rep("3.2 (s.e. 1.4)", 6)
)

number_format(ht)[3, -1] <- NA
number_format(ht)[4, -1] <- 2
number_format(ht)[5, -1] <- "\%5.2f"

number_format(ht)[6, -1] <- fmt_pretty()

number_format(ht)[7, -1] <- list(
  function(x) if (x > 0) "+" else "-"
right_border(ht) <- 1
bottom_border(ht)[1, ] <- 1

ht

ht_bands <- huxtable("10000 Maniacs", autoformat = FALSE)
# probably not what you want:
ht_bands
# fixed:
set_number_format(ht_bands, NA)

padding  Set padding

Description
These functions set the space around the edges of cells, within the borders.

Usage

left_padding(ht)
left_padding(ht) <- value
set_left_padding(ht, row, col, value)
map_left_padding(ht, row, col, fn)

right_padding(ht)
right_padding(ht) <- value
set_right_padding(ht, row, col, value)
map_right_padding(ht, row, col, fn)

top_padding(ht)
top_padding(ht) <- value
set_top_padding(ht, row, col, value)
map_top_padding(ht, row, col, fn)

bottom_padding(ht)
bottom_padding(ht) <- value
set_bottom_padding(ht, row, col, value)
map_bottom_padding(ht, row, col, fn)

Arguments

ht  A huxtable.
row  A row specifier. See rowspecs for details.
col  An optional column specifier.
fn A mapping function. See mapping-functions for details.
value Numeric: padding width/height in points.

See Also
set-multiple, set-outer.

Examples

```r
left_padding(jams) <- 2
left_padding(jams)

jams <- set_left_padding(jams, 2)
left_padding(jams)
```

---

**position**

*Set the table’s position with respect to surrounding content*

Description

Table position may be "left", "right" or "center". If you want text to wrap around the table, use "wrapleft" or "wrapright".

Usage

```r
position(ht)
position(ht) <- value
set_position(ht, value)
```

Arguments

- **ht**
  A huxtable.
- **value**
  String. "left", "center", "right", "wrapleft" or "wrapright". Set to NA to reset to the default, which is "center".

Details

"wrapleft" and "wrapright" position the table to the left or right, and allow text to wrap around the table.

Value

position() returns the position property. set_position() returns the modified huxtable.
Examples

```r
position(jams) <- "right"
position(jams)

set_position(jams, "left")
set_position(jams, "right")
set_position(jams, "center")
```

Description

By default huxtables are printed using `print_screen()`. In certain cases, for example in Sweave documents, it may be useful to change this. You can do so by setting `options("huxtable.print")`.

Usage

```r
## S3 method for class 'huxtable'
print(x, ...)

## S3 method for class 'huxtable'
format(x, ..., output = c("latex", "html", "md", "screen", "rtf"))
```

Arguments

- `x` A huxtable.
- `...` Options passed to other methods.
- `output` Output format. One of "html", "latex", "md", "screen", or "rtf".

Value

- `print` prints the huxtable and returns NULL invisibly.
- `format` returns a string representation from `to_latex()`, `to_html()` etc.

See Also

To change how huxtables are printed within knitr, see `options("huxtable.knitr_output_format")` in `huxtable-options`
print_html

Create HTML representing a huxtable

Description

These functions print or return an HTML table.

Usage

print_html(ht, ...)

to_html(ht, ...)

print_notebook(ht, ...)

## S3 method for class 'huxtable'
to_html(ht, ...)

Arguments

ht A huxtable.

... Arguments to pass to methods. Not currently used.

Value

to_html returns an HTML string. print_html prints the string and returns NULL.

print_notebook prints HTML output suitable for use in an RStudio interactive notebook.

See Also

Other printing functions: print_latex(), print_md(), print_rtf(), print_screen()

Examples

ht <- hux(a = 1:3, b = letters[1:3])
to_html(ht)
print_latex

Create LaTeX representing a huxtable

Description

Create LaTeX representing a huxtable

Usage

print_latex(ht, ...)

to_latex(ht, ...)

## S3 method for class 'huxtable'
to_latex(ht, tabular_only = FALSE, ...)

Arguments

ht A huxtable.

... Arguments to pass to methods.

tabular_only Return only the LaTeX tabular, not the surrounding float.

Details

If we appear to be in a rmarkdown document with the Pandoc markdown +raw_attribute extension available, to_latex will return LaTeX surrounded by a "raw attribute code block" (see https://pandoc.org/MANUAL.html#extension-raw_attribute). This helps protect against pandoc accidentally escaping the TeX code.

Value

to_latex returns a string. print_latex prints the string and returns NULL.

See Also

Other printing functions: print_html(), print_md(), print_rtf(), print_screen()

Examples

ht <- huxtable(
  a = 1:3,
  b = letters[1:3]
)

print_latex(ht)
Create Markdown representing a huxtable

Description

Create Markdown representing a huxtable

Usage

print_md(ht, ...)
to_md(ht, ...)

## S3 method for class 'huxtable'
to_md(ht, header = TRUE, min_width =getOption("width")/4, max_width = 80, ...)

Arguments

ht A huxtable.
...
Arguments passed to methods.
header Logical. Print the first row as a header?
min_width Minimum width in on-screen characters of the result.
max_width Maximum width in on-screen characters of the result. Overrides min_width.

Details

Only align and caption properties are used. The markdown format is multiline_tables, see the pandoc documentation.

Value

to_md() returns a string. print_md() prints the string and returns NULL.

See Also

Other printing functions: print_html(), print_latex(), print_rtf(), print_screen()

Examples

print_md(jams)
print_rtf

Create RTF representing a huxtable

Description
These functions print or return an RTF character string.

Usage

print_rtf(ht, fc_tables = rtf_fc_tables(ht), ...)
to_rtf(ht, ...)

## S3 method for class 'huxtable'
to_rtf(ht, fc_tables = rtf_fc_tables(ht), ...)

Arguments

ht
A huxtable.

fc_tables
See rtf_fc_tables().

... Arguments to pass to methods.

Details
RTF files use a single per-document table for colors, and one for fonts. If you are printing multiple huxtables in a document, you need to make sure that the font and color table is set up correctly and that the RTF tables refer back to them. See rtf_fc_tables().

1. Prepare all the huxtables;
2. Call rtf_fc_tables(), passing in all the huxtables;
3. Print the rtfFCTables object in the RTF document header;
4. Pass in the rtfFCTables object to each call to print_rtf.

Value
to_rtf returns a string representing an RTF table. The fc_tables attribute of the returned string will contain the fc_tables object that was passed in (or autocreated). print_rtf prints the string and returns NULL.

Limitations

- rmarkdown"s rtf_document can"t yet print out customized color tables, so custom fonts and colors won"t work in this context.
- col_width() and width() can only be numeric or "pt".
- wrap() has no effect: cell contents always wrap.
- rotation() can only be 90 or 270, i.e. text going up or down.
See Also

Other printing functions: \texttt{print_html()}, \texttt{print_latex()}, \texttt{print_md()}, \texttt{print_screen()}

Examples

\begin{verbatim}
print_rtf(jams)
\end{verbatim}

---

\textbf{print_screen} \hspace{3cm} \textit{Print a huxtable on screen}

Description

Print a huxtable on screen

Usage

\begin{verbatim}
print_screen(ht, ...)
to_screen(ht, ...)

## S3 method for class 'huxtable'
to_screen(
  ht,
  min_width = ceiling(getOption("width")/6),
  max_width = getOption("width", Inf),
  compact = TRUE,
  colnames = TRUE,
  color = getOption("huxtable.color_screen", default = TRUE),
  ...
)
\end{verbatim}

Arguments

\begin{verbatim}
ht \hspace{1cm} A huxtable.
... \hspace{1cm} Passed on to \texttt{to_screen}.
min_width \hspace{1cm} Minimum width in on-screen characters of the result.
max_width \hspace{1cm} Maximum width in on-screen characters of the result. Overrides \texttt{min_width}.
compact \hspace{1cm} Logical. To save space, don't print lines for empty horizontal borders.
colnames \hspace{1cm} Logical. Whether or not to print column names.
color \hspace{1cm} Logical. Whether to print the huxtable in color (requires the \texttt{crayon} package).
\end{verbatim}
Details

Screen display shows the following features:

- Table and caption positioning
- Merged cells
- Cell alignment
- Borders
- Cell background and border color (if the "crayon" package is installed)
- Text color, bold and italic (if the "crayon" package is installed)

Cell padding, widths and heights are not shown, nor are border styles.

Value

to_screen returns a string. print_screen prints the string and returns NULL.

See Also

Other printing functions: print_html(), print_latex(), print_md(), print_rtf()

Examples

```r
bottom_border(jams)[1, 1:2] <- 1
bold(jams)[1, 1:2] <- TRUE
jams <- map_text_color(jams,
    by_regex("berry" = "red"))

print_screen(jams)
```

Description

These functions use huxtable to print objects to an output document. They are useful as one-liners for data reporting.

Usage

```r
quick_latex(
    ...,
    file = confirm("huxtable-output.tex"),
    borders = 0.4,
    open = interactive()
)
```
quick_pdf(
    ..., 
    file = confirm("huxtable-output.pdf"), 
    borders = 0.4, 
    open = interactive(), 
    width = NULL, 
    height = NULL 
)

quick_html(
    ..., 
    file = confirm("huxtable-output.html"), 
    borders = 0.4, 
    open = interactive() 
)

quick_docx(
    ..., 
    file = confirm("huxtable-output.docx"), 
    borders = 0.4, 
    open = interactive() 
)

quick_pptx(
    ..., 
    file = confirm("huxtable-output.pptx"), 
    borders = 0.4, 
    open = interactive() 
)

quick_xlsx(
    ..., 
    file = confirm("huxtable-output.xlsx"), 
    borders = 0.4, 
    open = interactive() 
)

quick_rtf(
    ..., 
    file = confirm("huxtable-output.rtf"), 
    borders = 0.4, 
    open = interactive() 
)

Arguments

... One or more huxtables or R objects with an as_huxtable method.

file File path for the output.
**report_latex_dependencies**

Borders

Border width for members of ... that are not huxtables.

Open

Logical. Automatically open the resulting file?

Width

String passed to the LaTeX geometry package's paperwidth option. Use NULL for the default width.

Height

String passed to geometry's paperheight option. Use NULL for the default height.

Details

Objects in ... will be converted to huxtables, with borders added.

If ‘file’ is not specified, the command will fail in non-interactive sessions. In interactive sessions, the default file path is "huxtable-output.xxx" in the working directory; if this already exists, you will be asked to confirm manually before proceeding.

Value

Invisible NULL.

Examples

```r
## Not run:
m <- matrix(1:4, 2, 2)
quick_pdf(m, jams)
quick_latex(m, jams)
quick_html(m, jams)
quick_docx(m, jams)
quick_xlsx(m, jams)
quick_pptx(m, jams)
quick_rtf(m, jams)
## End(Not run)
```

**report_latex_dependencies**

Manage LaTeX dependencies for huxtables

Description

**report_latex_dependencies** prints out and/or returns a list of LaTeX dependencies for adding to a LaTeX preamble.

**check_latex_dependencies** checks whether the required LaTeX packages are installed.

**install_latex_dependencies** is a utility function to install and/or update the LaTeX packages that huxtable requires. It calls `tinytex::tlmgr_install()` if possible, or `tlmgr install` directly.
Usage

```r
report_latex_dependencies(quiet = FALSE, as_string = FALSE)
```

```r
check_latex_dependencies(quiet = FALSE)
```

```r
install_latex_dependencies()
```

Arguments

- **quiet**: Logical. For `report_latex_dependencies`, suppress printing of dependencies. For `check_latex_dependencies`, suppress messages.
- **as_string**: Logical: return dependencies as a string.

Value

If `as_string` is TRUE, `report_latex_dependencies` returns a string of "\\usepackage\{...\}" statements; otherwise it returns a list of `rmarkdown::latex_dependency` objects, invisibly.

`check_latex_dependencies()` returns TRUE or FALSE.

`install_latex_dependencies` returns TRUE if tlmgr returns 0.

Examples

```r
report_latex_dependencies()
```

```r
## Not run:
check_latex_dependencies()
```

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

```r
## Not run:
install_latex_dependencies()
```

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

**Description**

- `restack_across()` splits a huxtable horizontally, then joins the parts up side by side.
- `restack_down()` splits a huxtable vertically, then joins the parts up top to bottom.
restack-across-down

Usage

restack_across(
    ht,
    rows,
    headers = TRUE,
    on_remainder = c("warn", "stop", "fill")
)

restack_down(
    ht,
    cols,
    headers = TRUE,
    on_remainder = c("warn", "stop", "fill")
)

Arguments

ht A huxtable
rows, cols How many rows/columns the new result should have.
headers Logical. Take account of header rows/columns?
on_remainder String. "warn", "stop" or "fill". See below.

Details

If headers is TRUE, header rows/columns will be repeated across/down the restacked huxtable as necessary.
on_remainder determines what happens if the huxtable could not be evenly divided for restacking:
  • "stop": stop with an error.
  • "fill": fill the remainder with empty cells.
  • "warn" (the default): issue a warning, then fill the remainder with empty cells.

Value

A new huxtable.

See Also

split-across-down

Examples

ht <- as_hux(matrix(LETTERS[1:4], 2, 2))
ht <- set_allBorders(ht)
ht

restack_down(ht, 1)
restack_across(ht, 1)

# headers:
restack_across(jams, 2)
restack_across(jams, 2,
    headers = FALSE)

# on_remainder:
restack_across(jams, 3,
    on_remainder = "fill")

---

rotation | Rotate text within cells

### Description
Numbers represent degrees to rotate text anti-clockwise:

### Usage
rotation(ht)
rotation(ht) <- value
set_rotation(ht, row, col, value)
map_rotation(ht, row, col, fn)

### Arguments
- **ht** A huxtable.
- **row** A row specifier. See rowspecs for details.
- **col** An optional column specifier.
- **fn** A mapping function. See mapping-functions for details.
- **value** A numeric vector or matrix. Set to NA to reset to the default, which is 0.

### Details
- 0 is the default;
- 90 is going upwards, for left-to-right languages;
- 270 is going downwards.

You will probably need to set col_width() and row_height() explicitly to achieve a nice result, in both HTML and LaTeX.

### Value
rotation() returns the rotation property. set_rotation() returns the modified huxtable.
Examples

rotation(jams) <- 90
rotation(jams)

jams2 <- set_rotation(jams, 270)
rotation(jams2)

jams3 <- set_rotation(jams, 2:3, 1, 270)
rotation(jams3)

jams4 <- map_rotation(jams, by_rows(270, 90))
rotation(jams4)

Description

This help page describes how to use the row and col arguments in set_* functions.

The basics

The set_* functions for cell properties all have arguments like this: set_property(ht, row, col, value).

You can treat row and col arguments like arguments for data frame subsetting. For example, you can use row = 1:3 to get the first three rows, col = "salary" to specify the column named "salary", or row = ht$salary >= 50000 to specify rows where a condition is true.

There are also a few extra tricks you can use:

- Write set_property(ht, x), omitting row and col, to set the property to x for all cells.
- Use everywhere to refer to all rows or all columns.
- Use final(n) to refer to the last n rows or columns.
- Use evens to get only even rows/columns and odds for only odd ones.
- Use stripe(n, from = m) to get every nth row/column starting at row/column m.
- Use dplyr functions like starts_with, contains and matches to specify columns (but not rows). See tidyselect::language for a full list.
The gory details

How the row and col arguments are parsed depends on the number of arguments passed to the set_* function.

- If there are two arguments then the second argument is taken as the value and is set for all rows and columns.
- If there are four arguments:
  - If row or col is numeric, character or logical, it is evaluated just as in standard subsetting. col will be evaluated in a special context provided by tidyselect::with_vars() to allow the use of dplyr functions.
  - If row or col is a function, it is called with two arguments: the huxtable, and the dimension number being evaluated, i.e. 1 for rows, 2 for columns. It must return a vector of column indices. evens(), odds(), stripe() and final() return functions for this purpose.

Examples

```r
set_bold(jams, 2:4, 1:2, TRUE)
set_background_color(jams, evens, everywhere, "grey95")
set_bold(jams, everywhere,
tidyselect::matches("yp"), TRUE)

set_text_color(jams, 2:4, 1:2,
c("red", "violetred", "purple"))
```

---

**row_height**

*Set the height of table rows*

---

Description

Numeric heights are scaled to 1 and treated as proportions of the table height in HTML, or of the text height (\textheight) in LaTeX. Character row heights must be valid CSS or LaTeX dimensions.

Usage

```r
row_height(ht)
row_height(ht) <- value
set_row_height(ht, row, value)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ht</td>
<td>A huxtable.</td>
</tr>
<tr>
<td>row</td>
<td>A row specifier. See rowspecs for details.</td>
</tr>
<tr>
<td>value</td>
<td>Numeric or character vector. Set to NA to reset to the default, which is NA.</td>
</tr>
</tbody>
</table>
Value

row_height() returns the row_height property. set_row_height() returns the modified huxtable.

See Also

Other table measurements: col_width(), height(), width()

Examples

```r
row_height(jams) <- c(.4, .2, .2, .2)
row_height(jams)
```

---

rtf_fc_tables  
Create RTF font and color tables

Description

Create RTF font and color tables

Usage

```r
rtf_fc_tables(..., extra_fonts = "Times", extra_colors = character(0))
```

Arguments

...  
One or more objects of class huxtable.

extra_fonts  
Extra fonts to include. These will be first in the fonts table.

extra_colors  
Extra colors to include, as R color names.

Details

RTF documents have a single table of fonts, and a table of colors, in the RTF header. To create font and color tables for multiple huxtables, use this command. You can print the returned object in the RTF header. Pass it to print_rtf() or to_rtf() to ensure that huxtables print out the correct colour references.

Value

An object of class rtfFCTables. This is a list containing two items: "fonts" is a character vector of unique font names; "colors" is a character vector of unique color names.
\textbf{Examples}

\begin{verbatim}
# Printing multiple huxtables:

ht <- huxtable("Blue with red border")
ht <- set_all_borders(ht, 1)
ht <- set_all_border_colors(ht, "red")
background_color(ht) <- "blue"

ht2 <- huxtable("Dark green text")
text_color(ht2) <- "darkgreen"

fc_tbls <- rtf_fc_tables(ht, ht2)

# In the document header:
print(fc_tbls)

# In the document body:
print_rtf(ht, fc_tables = fc_tbls)
print_rtf(ht2, fc_tables = fc_tbls)
\end{verbatim}

\textbf{sanitize} \hspace{1cm} \textit{Escape text for various formats}

\textbf{Description}

This escapes a string for LaTeX, HTML or RTF.

\textbf{Usage}

\texttt{sanitize(str, type = c("latex", "html", "rtf"))}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{str} \hspace{1cm} A character object.
  \item \texttt{type} \hspace{1cm} "latex", "html" or "rtf".
\end{itemize}

\textbf{Details}

HTML and LaTeX code was copied over from \texttt{xtable::sanitize()}.

\textbf{Value}

The sanitized character object.

\textbf{Examples}

\begin{verbatim}
  txt <- "Make $$ with us"
sanitize(txt, type = "latex")
\end{verbatim}
set-multiple

**Set left, right, top and bottom properties**

**Description**

These functions set left, right, top and/or bottom properties simultaneously for the specified cells.

**Usage**

```r
set_allBorders(ht, row, col, value = 0.4)
map_allBorders(ht, row, col, fn)
set_allBorderColors(ht, row, col, value)
map_allBorderColors(ht, row, col, fn)
set_allBorderStyle(ht, row, col, value)
map_allBorderStyle(ht, row, col, fn)
set_allPadding(ht, row, col, value)
map_allPadding(ht, row, col, fn)
set_tbPadding(ht, row, col, value)
map_tbPadding(ht, row, col, fn)
set_lrPadding(ht, row, col, value)
map_lrPadding(ht, row, col, fn)
set_tbBorders(ht, row, col, value)
map_tbBorders(ht, row, col, fn)
set_lrBorders(ht, row, col, value)
map_lrBorders(ht, row, col, fn)
set_tbBorderColor(ht, row, col, value)
map_tbBorderColor(ht, row, col, fn)
set_lrBorderColor(ht, row, col, value)
```
map_lr_border_colors(ht, row, col, fn)
set_tb_border_styles(ht, row, col, value)
map_tb_border_styles(ht, row, col, fn)
set_lr_border_styles(ht, row, col, value)
map_lr_border_styles(ht, row, col, fn)

Arguments

ht  A huxtable.
row  A row specifier. See rowspecs for details.
col  An optional column specifier.
value  Value(s) to set. Set to NA to reset to the default.
fn  A mapping function. See mapping-functions for details.

Details

- set_all_* functions set top, bottom, left and right properties.
- set_tb_* functions set top and bottom properties.
- set_lr_* functions set left and right properties.

Value

The modified huxtable.

See Also

borders, border-colors, border-styles, padding.

Examples

ht <- as_hux(jams)
ht <- set_allBorders(ht)
ht <- set_all_border_colors(ht, "red")
ht <- set_all_border_styles(ht, "double")
ht <- set_all_padding(ht, 1:3, 1:2, "20px")
ht <- set_tb_padding(ht, 10)
ht <- set_tbBorders(ht)
set_tb_border_colors(ht, "red")
set_tb_border_styles(ht, "double")
set-outer

Set borders and padding around a rectangle of cells

Description

Set borders and padding around a rectangle of cells

Usage

set_outer_borders(ht, row, col, value = 0.4)
set_outer_border_colors(ht, row, col, value)
set_outer_border_styles(ht, row, col, value)
set_outer_padding(ht, row, col, value)

Arguments

ht
A huxtable.
row
A row specifier. See rowspecs for details.
col
An optional column specifier.
value
Border width, color, style or a brdr() object. See borders. For padding, padding width in points.

Details

set_outer_borders sets borders round the top, bottom, left and right of a group of cells. Behaviour is undefined unless row and col specify contiguous sequences. set_outer_border_colors and set_outer_border_styles set border colors and styles. set_outer_padding sets padding, i.e. top padding on the top row of cells, etc.

Examples

ht2 <- huxtable(a = 1:3, b = 1:3)
set_outer_borders(ht2)
set_outer_borders(ht2, 2:3, 1:2)
set_contents

Set cell contents

Description

set_contents() is a convenience function to change the cell contents of a huxtable within a dplyr chain. set_contents(ht, x, y, foo) just calls ht[x, y] <- foo and returns ht.

Usage

contents(ht)
contents(ht) <- value
set_contents(ht, row, col, value )
map_contents(ht, row, col, fn)

Arguments

ht     A huxtable.
row    A row specifier. See rowspecs for details.
col    An optional column specifier.
fn      A mapping function. See mapping-functions for details.
value  Cell contents.

Examples

set_contents(jams, 2, 1, "Blackcurrant")
map_contents(jams, by_regex(".*berry" = "Snodberry"))

set_default_properties

Default huxtable properties

Description

Defaults are used for new huxtables, and also when a property is set to NA.

Usage

set_default_properties(...)   
get_default_properties(names = NULL)
Arguments

... Properties specified by name, or a single named list.
names Vector of property names. If NULL, all properties are returned.

Details

Note that autoformat = TRUE in huxtable() overrides some defaults.
To set default border styles, use the pseudo-properties border/border_style/border_color. You cannot set defaults separately for different sides.

Value

For set_default_properties, a list of the previous property values, invisibly.
For get_default_properties, a list of the current defaults.

See Also

Options for autoformat in huxtable-options.

Examples

```r
old <- set_default_properties(
  text_color = "red",
  border = 0.4
)

hux(a = 1:2, b = 1:2)
set_default_properties(old)
get_default_properties("bold")
```

Description

This convenience function calls set_contents() and set_markdown().

Usage

```r
set_markdown_contents(ht, row, col, value)
```

Arguments

- `ht` A huxtable.
- `row` A row specifier. See rowspecs for details.
- `col` An optional column specifier.
- `value` Cell contents, as a markdown string.
Value

The modified huxtable.

Note

Markdown content in cells is completely separate from printing the whole table as markdown using `print_md()`. When you set markdown to TRUE, huxtable itself interprets the cell contents as markdown, and spits out HTML, TeX or whatever.

See Also

`markdown()`.

Examples

```r
set_markdown_contents(jams, 1, 1, 
  "**Type** of jam")
```

---

<table>
<thead>
<tr>
<th>spans</th>
<th>Extend cells over multiple rows and/or columns</th>
</tr>
</thead>
</table>

Description

A cell with rowspan of 2 covers the cell directly below it. A cell with colspan of 2 covers the cell directly to its right. A cell with rowspan of 2 and colspan of 2 covers a 2 x 2 square, hiding three other cells.

Usage

```r
rowspan(ht)
rowspan(ht) <- value
set_rowspan(ht, row, col, value)
map_rowspan(ht, row, col, fn)
```

```r
colspan(ht)
colspan(ht) <- value
set_colspan(ht, row, col, value)
map_colspan(ht, row, col, fn)
```

Arguments

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.
- **fn**: A mapping function. See `mapping-functions` for details.
- **value**: An integer vector or matrix.
Cell content

In merged cell ranges, only the top left cell’s content is displayed. In addition, when you merge cells (either by setting \texttt{colspan()} or \texttt{rowspan()}, or using \texttt{merge_cells()} and friends) the content of the top left cell is copied to other cells. This prevents unexpected changes to content if you reorder or subset rows and columns.

See Also

\texttt{merge_cells()}, \texttt{merge_across()} and \texttt{merge_down()} for a higher-level interface.

Examples

```r
letter_hux <- as_hux(matrix(LETTERS[1:9], 3, 3))
letter_hux <- set_all_borders(letter_hux)
letter_hux
set_rowspan(letter_hux, 1, 1, 2)
set_colspan(letter_hux, 1, 1, 2)
```

split-across-down  \hspace{1em} \textit{Split a huxtable into multiple huxtables}

Description

These functions split a huxtable horizontally or vertically, and return the new sub-tables in a list.

Usage

```r
split_across(ht, after, height, headers = TRUE)

split_down(ht, after, width, height, headers = TRUE)
```

Arguments

- \texttt{ht} \hspace{1em} A huxtable.
- \texttt{after} \hspace{1em} Rows/columns after which to split. See \texttt{rowspecs} for details. Note that \texttt{tidyselect} semantics are allowed in \texttt{split_down()} but not \texttt{split_across()}.
- \texttt{height, width} \hspace{1em} Maximum height/width for the result.
- \texttt{headers} \hspace{1em} Logical. Take account of header rows/columns?

Details

Only one of \texttt{after} and \texttt{width} or \texttt{height} must be given. If \texttt{width} or \texttt{height} is given, the huxtable will be split by \texttt{col_width()} or \texttt{row_height()}, which must be numeric with no \texttt{NA} values.

If \texttt{headers} is \texttt{TRUE}, all previous headers will be added to each new table.
Value

A list of huxtables.

See Also

restack-across-down

Examples

ht <- as_hux(matrix(LETTERS[1:16], 4, 4))
ht <- set_all_borders(ht)
split_across(ht, after = 2)
split_down(ht, after = c(1, 3))

col_width(ht) <- c(0.15, 0.1, 0.25, 0.3)
split_down(ht, width = 0.3)

# split by column name:
split_down(jams, "Type")

# headers are repeated:
split_across(jams, 3)

stripe(n, from)
everywhere(ht, dimension)
evens(ht, dimension)
odds(ht, dimension)
stripes

Arguments

- **n**  A number (at least 1)
- **from**  A number (at least 1)
- **ht**  An object with a `dim` attribute like a matrix or data frame.
- **dimension**  Number of the dimension to use.

Details

Technically, `stripe` returns a 2-argument function which can be called like `f(ht, dimension)`. See `rowspecs` for details.

Until `huxtable` 5.0.0, `stripe` was called `every`. It was renamed to avoid a clash with `purrr::every`.

Examples

```r
ht <- huxtable(a = 1:10, b = 1:10)
set_background_color(ht, evens, everywhere, "grey95")
set_background_color(ht, stripe(3), everywhere, "grey95")
```

---

strips  

*Set background color stripes*

Description

These convenience functions call `map_background_color` with `by_rows` or `by_cols`.

Usage

- `stripe_rows(ht, stripe1 = "white", stripe2 = "grey90")`
- `stripe_columns(ht, stripe1 = "white", stripe2 = "grey90")`

Arguments

- **ht**  A huxtable.
- **stripe1**  Color for rows/columns 1, 3, 5, ...
- **stripe2**  Color for rows/columns 2, 4, 6, ...
Examples

stripe_rows(jams)
stripe_columns(jams)
stripe_rows(jams, "red", "blue")

Description

These functions set arbitrary cell properties on cells in header rows and/or columns.

Usage

style_headers(ht, ...)
style_header_rows(ht, ...)
style_header_cols(ht, ...)
style_cells(ht, row, col, ...)
set_cell_properties(ht, row, col, ...)

Arguments

ht A huxtable.
... Named list of cell properties.
row A row specifier. See rowspecs for details.
col An optional column specifier.

Details

- style_headers sets properties on all header cells.
- style_header_rows sets properties on header rows.
- style_header_cols sets properties on header columns.
- style_cells sets properties on all selected cells.
set_cell_properties is a deprecated alias for style_cells. Don't use it.
Examples

```r
colored <- c(NA, "red", "darkred", "purple")

jams <- set_header_cols(jams, 1, TRUE)
style_header_cols(jams, text_color = colored)

style_cells(jams, everywhere, 2, bold = TRUE)
```

Description

`t()` switches a huxtable so rows become columns and columns become rows.

Usage

```r
## S3 method for class 'huxtable'
t(x)
```

Arguments

- `x`: A huxtable.

Details

Row and column spans of `x` will be swapped, as will column widths and row heights, table width and height, and cell borders (bottom becomes right, etc.). Other properties - in particular, alignment, vertical alignment and rotation - will be preserved.

Value

The transposed huxtable.

Examples

```r
ht <- huxtable(
a = 1:3,
b = letters[1:3],
autoformat = FALSE
)
bottom_border(ht)[3,] <- 1
ht
t(ht)
```
tabular_environment

Set the "table" environment in LaTeX

Description

By default this is "table".

Usage

    table_environment(ht)
    table_environment(ht) <- value
    set_table_environment(ht, value)

Arguments

    ht            A huxtable.
    value         A string. Set to NA to reset to the default, which is "table".

Details

No features are guaranteed to work if you set this to a non-default value. Use at your own risk! In particular, you may need to set latex_float() to a non-default value.
If position() is set to "wrapleft" or "wrapright", this value is overridden.

Value

    table_environment() returns the table_environment property. set_table_environment() returns the modified huxtable.

Examples

    table_environment(jams) <- "table*
    table_environment(jams)

tabular_environment

Set the table's tabular environment in LaTeX

Description

By default this is either "tabular" or "tabularx".

Usage

    tabular_environment(ht)
    tabular_environment(ht) <- value
    set_tabular_environment(ht, value)
text_color

Arguments

- **ht**: A huxtable.
- **value**: A string. Set to NA to reset to the default, which is "NA".

Details

No features are guaranteed to work if you set this to a non-default value. Use at your own risk!

Value

- `tabular_environment()` returns the `tabular_environment` property. `set_tabular_environment()` returns the modified huxtable.

Examples

```r
tabular_environment(jams) <- "longtable"
tabular_environment(jams)
```

Description

Colors can be in any format understood by R:

- A color name like "darkred"  
- A HTML string like "#FF0000"  
- The result of a function like `rgb(1, 0, 0)` or `grey(0.5)`

Usage

```r
text_color(ht)
text_color(ht) <- value
set_text_color(ht, row, col, value)
map_text_color(ht, row, col, fn)
```

Arguments

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.
- **fn**: A mapping function. See `mapping-functions` for details.
- **value**: A character vector or matrix. Set to NA to reset to the default, which is "NA".
Value

text_color() returns the text_color property. set_text_color() returns the modified huxtable.

See Also

Other formatting functions: background_color(), bold(), font_size(), font(), na_string(), number_format()

Examples

text_color(jams) <- "blue"
text_color(jams)

set_text_color(jams, "red")
set_text_color(jams,
  2:3, 1, "red")
map_text_color(jams,
  by_rows("red", "blue"))

themes Theme a huxtable

Description

These functions quickly set default styles for a huxtable.

Usage

theme_plain(ht, header_rows = TRUE, position = "center")

theme_bright(
  ht,
  header_rows = TRUE,
  header_cols = FALSE,
  colors = c("#7eabf2", "#e376e3", "#fcbb03", "#7aba59", "#fc0356")
)

theme_basic(ht, header_rows = TRUE, header_cols = FALSE)

theme_compact(ht, header_rows = TRUE, header_cols = FALSE)

theme_striped(
  ht,
  stripe = "grey90",
  stripe2 = "grey95",
  header_rows = TRUE,
  header_cols = TRUE)
themes

)

theme_grey(ht, header_rows = TRUE, header_cols = TRUE)
theme_blue(ht, header_rows = TRUE, header_cols = TRUE)
theme_orange(ht, header_rows = TRUE, header_cols = TRUE)
theme_green(ht, header_rows = TRUE, header_cols = TRUE)
theme_article(ht, header_rows = TRUE, header_cols = TRUE)
theme_mondrian(ht, prop_colored = 0.1, font = NULL)

Arguments

ht  A huxtable object.
header_rows Logical: style header rows?
position "left", "center" or "right"
header_cols Logical: style header columns?
colors  Colors for header rows. Can also be a palette function.
stripe  Background colour for odd rows
stripe2  Background colour for even rows
prop_colored Roughly what proportion of cells should have a primary-color background?
font  Font to use. For LaTeX, try "cmss".

Details

• theme_plain is a simple theme with a bold header, a grey striped background, and an outer border.
• theme_basic sets header rows/columns to bold, and adds a border beneath them.
• theme_compact is like theme_basic but with minimal padding.
• theme_striped uses different backgrounds for alternate rows, and for headers.
• theme_article is similar to the style of many scientific journals. It sets horizontal lines above and below the table.
• theme_bright uses thick white borders and a colourful header. It works nicely with sans-serif fonts.
• theme_grey, theme_blue, theme_orange and theme_green use white borders and subtle horizontal stripes.
• theme_mondrian mimics the style of a Mondrian painting, with thick black borders and randomized colors.

Value

The huxtable object, appropriately styled.
Examples

```r
theme_plain(jams)
theme_basic(jams)
theme_compact(jams)
theme-striped(jams)
theme_article(jams)
theme_bright(jams)
theme_grey(jams)
theme_blue(jams)
theme_orange(jams)
theme_green(jams)
theme_mondrian(jams)
```

```r
## Not run:
quick_pdf(
  themePlain(jams),
  themeBasic(jams),
  themeCompact(jams),
  themeStriped(jams),
  themeArticle(jams),
  themeBright(jams),
  themeGrey(jams),
  themeBlue(jams),
  themeOrange(jams),
  themeGreen(jams),
  themeMondrian(jams)
)
```

## End(Not run)

---

**tidy_override**  
*Change a model’s tidy output*

**Description**

Use `tidy_override` and `tidy_replace` to provide your own p values, confidence intervals etc. for a model.

**Usage**

```r
tidy_override(x, ..., glance = list(), extend = FALSE)
```

```r
tidy_replace(x, tidied, glance = list())
```

```r
## S3 method for class 'tidy_override'
tidy(x, ...)
```

```r
## S3 method for class 'tidy_override'
```
`tidy_override`

glance(x, ...)

## S3 method for class 'tidy_override'
nobs(object, ...)

### Arguments

- **x**: A model with methods defined for `generics::tidy()` and/or `generics::glance()`.
- **...**: In `tidy_override`, columns of statistics to replace tidy output. In tidy and glance methods, arguments passed on to the underlying model.
- **glance**: A list of summary statistics for glance.
- **extend**: Logical: allow adding new columns to `tidy(x)` and `glance(x)`?
- **tidied**: Data frame to replace the result of `tidy(x)`.
- **object**: A `tidy_override` object.

### Details

tidy_override allows you to replace some columns of `tidy(x)` with your own data.
tidy_replace allows you to replace the result of `tidy(x)` entirely.

### Value

An object that can be passed in to huxreg.

### Examples

```r
if (!requireNamespace("broom", quietly = TRUE)) {
  stop("Please install 'broom' to run this example.")
}

lm1 <- lm(mpg ~ cyl, mtcars)
fixed_lm1 <- tidy_override(lm1, 
p.value = c(.04, .12),
  glance = list(r.squared = 0.99))
huxreg(lm1, fixed_lm1)

if (requireNamespace("nnet", quietly = TRUE)) {
  mn1 <- nnet::multinom(gear ~ mpg, mtcars)
tidied <- broom::tidy(mn1)
mnl4 <- tidy_replace(mn1, tidied[tidied$y.level == 4, ])
mnl5 <- tidy_replace(mn1, tidied[tidied$y.level == 5, ])
huxreg(mnl4, mnl5, statistics = "nobs")
}
```
valign  

*Set the vertical alignment of cell content*

---

**Description**

Allowed values are "top", "middle", "bottom" or NA.

**Usage**

```r
valign(ht)
valign(ht) <- value
set_valign(ht, row, col, value)
map_valign(ht, row, col, fn)
```

**Arguments**

- `ht`  
  A huxtable.
- `row`  
  A row specifier. See `rowspecs` for details.
- `col`  
  An optional column specifier.
- `fn`  
  A mapping function. See `mapping-functions` for details.
- `value`  
  A character vector or matrix.
  Set to NA to reset to the default, which is "top".

**Details**

Vertical alignment may not work for short text in LaTeX. Defining row heights with `row_height()` may help.

**Value**

`valign()` returns the `valign` property. `set_valign()` returns the modified huxtable.

**Examples**

```r
valign(jams) <- "top"
valign(jams)

jams2 <- set_valign(jams, "bottom")
valign(jams2)

jams3 <- set_valign(jams, 2:3, 1, "bottom")
valign(jams3)

jams4 <- map_valign(jams,
```
**width**

by_rows(
  "bottom",
  "top"
)
valign(jams4)

<table>
<thead>
<tr>
<th>width</th>
<th>Set the table width</th>
</tr>
</thead>
</table>

**Description**

`width()` sets the width of the entire table, while `col_width()` sets the width of individual columns. A numeric width is treated as a proportion of the surrounding block width (HTML) or text width (LaTeX). A character width must be a valid CSS or LaTeX dimension.

**Usage**

```
width(ht)
width(ht) <- value
set_width(ht, value)
```

**Arguments**

- **ht**  
  A huxtable.
- **value**  
  A number or string. Set to NA to reset to the default, which is NA.

**Value**

`width()` returns the width property. `set_width()` returns the modified huxtable.

**See Also**

Other table measurements: `col_width()`, `height()`, `row_height()`

**Examples**

```
width(jams) <- 0.8
width(jams)
```
wrap

Wrap cell content over multiple lines

Description

Text wrapping only works when the table `width()` has been set. In particular, if you want to insert newlines in cells, then you should set a value for `width()` and set `wrap` to TRUE.

Usage

```r
wrap(ht) 
wrap(ht) <- value 
set_wrap(ht, row, col, value ) 
map_wrap(ht, row, col, fn)
```

Arguments

- **ht**: A huxtable.
- **row**: A row specifier. See `rowspecs` for details.
- **col**: An optional column specifier.
- **fn**: A mapping function. See `mapping-functions` for details.
- **value**: A logical vector or matrix. Set to `NA` to reset to the default, which is TRUE.

Value

`wrap()` returns the `wrap` property. `set_wrap()` returns the modified huxtable.

Examples

```r
long_text <- paste( 
  rep("Some long text.", 10), 
  collapse = " 
)
ht <- huxtable(Long = long_text) 
width(ht) <- 0.2
wrap(ht) <- TRUE

## Not run:
quick_html(ht)

## End(Not run)
```
Description

Subset a huxtable

Usage

```r
## S3 method for class 'huxtable'
x[i, j, drop = FALSE]
```

```r
## S3 replacement method for class 'huxtable'
x[i, j] <- value
```

```r
## S3 replacement method for class 'huxtable'
x$name <- value
```

```r
## S3 replacement method for class 'huxtable'
x[[i, j]] <- value
```

Arguments

- **x**: A huxtable.
- **i**: Rows to select.
- **j, name**: Columns to select.
- **drop**: Only included for compatibility with `[.data.frame`. Do not use.
- **value**: A matrix, data frame, huxtable or similar object.

Value

`[` returns a huxtable. `$` and `[[` return data from the underlying data frame.

Replacing existing rows and columns

For the replacement function `<-`, if value is a huxtable, then its properties will be copied into `x`. Replacement functions `$<-` and `[[<-` replace existing data without affecting any properties.

Adding new rows and columns

If new columns or rows are created, then properties will be copied from the last column or row of `x`, or from value if value is a huxtable.

These methods are stricter than their data frame equivalents in some places. You can’t add new rows or column at a numeric location without specifying all intervening rows/columns. New values must have the appropriate dimensions (vectors will be interpreted appropriately).
Examples

```
jams[1:3, ]
class(jams[1:3, ])
jams[, 1]
jams$Type
prices <- huxtable(c("Price", 1.70, 2.00, 2.20))
number_format(prices) <- 2
bold(prices) <- TRUE
jams[, 2] <- prices
jams

data(jams)
jams$price <- c("Price", 1.70, 2.00, 2.20)
jams
```
Index

* border properties
  border-colors, 16
  border-styles, 17
  borders, 18
* caption properties
  caption, 30
  caption_pos, 31
  caption_width, 32
* cell merging
  merge_across, 72
  merge_cells, 73
  merge_repeated_rows, 74
* datasets
  jams, 65
* format functions
  fmt_percent, 37
  fmt_pretty, 38
* formatting functions
  background_color, 14
  bold, 15
  font, 39
  font_size, 40
  na_string, 76
  number_format, 77
  text_color, 109
* knit_print
  knit_print.data.frame, 66
  knit_print.huxtable, 67
* mapping functions
  by_cases, 21
  by_colorspace, 22
  by_function, 23
  by_quantiles, 24
  by_ranges, 26
  by_regex, 27
  by_rows, 28
  by_values, 29
* printing functions
  print_html, 82
  print_latex, 83
  print_md, 84
  print_rtf, 85
  print_screen, 86
* table measurements
  col_width, 35
  height, 43
  row_height, 94
  width, 115
  [.huxtable, 117
  [<-.huxtable [.huxtable], 117
  [[<-.huxtable [.huxtable], 117
  $<-.huxtable [.huxtable], 117
  `bottom_border<-`(borders), 18
  `bottom_border_color<-`(border-colors), 16
  `bottom_border_style<-`(border-styles), 17
  `left_border<-`(borders), 18
  `left_border_color<-`(border-colors), 16
  `left_border_style<-`(border-styles), 17
  `right_border<-`(borders), 18
  `right_border_color<-`(border-colors), 16
  `right_border_style<-`(border-styles), 17
  `top_border<-`(borders), 18
  `top_border_color<-`(border-colors), 16
  `top_border_style<-`(border-styles), 17
  add_colnames, 5
  add_columns (add_rows), 7
  add_columns(), 33, 65
  add_footnote, 6
  add_rownames (add_colnames), 5
  add_rows, 7
  add_rows(), 65
  align, 8
align(), 47, 63
align<-> (align), 8
as.matrix(), 22, 70
as.numeric(), 13
as_hux (as_huxtable), 10
as_huxtable(), 4, 62, 63
as_Workbook, 12

background_color, 14, 15, 39, 41, 77, 78, 110
background_color<- (background_color), 14
bold, 14, 15, 39, 41, 77, 78, 110
bold<-> (bold), 15
border-colors, 16, 98
border-styles, 17, 98
borders, 17, 18, 18, 98, 99
bottom_border (borders), 18
bottom_border<- (borders), 18
bottom_border_color (border-colors), 16
bottom_border_color<- (border-colors), 16
bottom_border_style (border-styles), 17
bottom_border_style<- (border-styles), 17
bottom_padding (padding), 79
bottom_padding<- (padding), 79
brdr, 20
brdr(), 17–19, 21, 22, 99
brdr_thickness, 21
by_cases, 21, 23–26, 28, 29
by_cases(), 70
by_cols, 105
by_cols(by_rows), 28
by_cols(), 69
by_equal_groups (by_quantiles), 24
by_equal_groups(), 70
by_function, 22, 23, 23, 25, 26, 28, 29
by_function(), 70
by_quantiles, 22–24, 26, 28, 29
by_quantiles(), 70
by_ranges, 22–25, 26, 28, 29
by_ranges(), 70
by_regex, 22–26, 27, 29
by_regex(), 70
by_rows, 22–26, 28, 29, 105

by_rows(), 69
by_values, 22–26, 28, 29, 29
by_values(), 70
caption, 30, 31, 32
caption<-> (caption), 30
caption_pos, 30, 31, 32
caption_pos<- (caption_pos), 31
caption_width, 30, 31, 32
caption_width<-> (caption_width), 32
cbind.huxtable, 32
cbind.huxtable(), 7, 64
check_latex_dependencies
  (report_latex_dependencies), 89
check_latex_dependencies(), 48
col_width, 35, 43, 95, 115
col_width<-> (col_width), 35
colspan (spans), 102
colspan(), 73–75, 103
colspan<-> (spans), 102
column_to_header, 34
contents (set_contents), 100
contents<- (set_contents), 100
data frame subsetting, 93
data.frame(), 46
diag(), 49
dplyr-verbs (mutate.huxtable), 75
dplyr::arrange(), 75
dplyr::case_when(), 21
dplyr::grouped_df(), 11
dplyr::mutate(), 75
dplyr::pull(), 75
dplyr::rename(), 75
dplyr::select(), 75
dplyr::slice(), 75
dplyr::summarise(), 75
dplyr::transmute(), 75
escape_contents, 36
escape_contents<-> (escape_contents), 36
evens (stripe), 104
evens(), 94
every (stripe), 104
everywhere (stripe), 104
final, 37
final(), 94
flextable::flextable(), 9
fmt_percent, 37, 38
fmt_percent(), 78
fmt_percent(), 38, 38
fmt_pretty(), 78
fmt_pretty(), 38, 38
font, 14, 15, 39, 41, 77, 78, 110
font<-(font), 39
font_size, 14, 15, 39, 40, 77, 78, 110
font_size<-(font_size), 40
format.huxtable(print.huxtable), 81
formatC(), 37
generics::glance(), 44, 45, 113
generics::tidy(), 44, 45, 113
get_default_properties
  (set_default_properties), 100
glance.tidy_override(tidy_override), 112
glue::glue(), 45
grepl(), 27
guess_knitr_output_format, 41
guess_knitr_output_format(), 62
header.cols, 42
header.cols<- (header.cols), 42
header.rows (header.cols), 42
header.rows<- (header.cols), 42
height, 35, 43, 95, 115
height(), 13
height<- (height), 43
hx<huxtable>, 46
hux.logo, 63
hxreg, 43
hxtable, 46
hxtable(), 4, 11, 62, 63, 101
hxtable=FAQ, 4, 48
hxtable=news, 49
hxtable=options, 39, 47, 62, 66, 67, 78, 81, 101
hxtable-package, 4, 47
insert.column, 64
insert.column(), 7
insert.row(insert.column), 64
insert.row(), 7
install_latex_dependencies
  (report_latex_dependencies), 89
install_latex_dependencies(), 48
is_hux(as_huxtable), 10
is_huxtable(as_huxtable), 10
italic (bold), 15
italic<-(bold), 15
jams, 65
knit_print.data.frame, 66, 67
knit_print.huxtable, 66, 67
knitr::knit_print(), 67
label, 67
label(), 49
label<- (label), 67
latex_float, 68
latex_float(), 108
latex_float<- (latex_float), 68
left_border (borders), 18
left_border<- (borders), 18
left_border_color (border-colors), 16
left_border_color<- (border-colors), 16
left_border_style (border-styles), 17
left_border_style<- (border-styles), 17
left.padding (padding), 79
left.padding<- (padding), 79
library(), 62
lmtest::coeftest(), 45
map_align (align), 8
map_all_borders (set-multiple), 97
map_all_border_colors (set-multiple), 97
map_all_border_styles (set-multiple), 97
map_all_borders (padding), 79
map_all_padding (padding), 79
map_background_color, 105
map_background_color (background_color), 14
map_bold (bold), 15
map_bottom_border (borders), 18
map_bottom_border_color (border-colors), 16
map_bottom_border_style (border-styles), 17
map_bottom_padding (padding), 79
map_colspan (spans), 102
map_contents (set_contents), 100
map_escape_contents (escape_contents), 36
map_font (font), 39
map_font_size (font_size), 40
map_italic (bold), 15
map_left_border (borders), 18
map_left_border_color (border-colors), 16
map_left_border_style (border-styles), 17
map_left_padding (padding), 79
map_lr_border_colors (set-multiple), 97
map_lr_border_styles (set-multiple), 97
map_lr_borders (set-multiple), 97
map_lr_padding (set-multiple), 97
map_markdown (markdown), 71
map_na_string (na_string), 76
map_number_format (number_format), 77
map_right_border (borders), 18
map_right_border_color (border-colors), 16
map_right_border_style (border-styles), 17
map_right_padding (padding), 79
map_rotation (rotation), 92
map_rowspan (spans), 102
map_tb_border_colors (set-multiple), 97
map_tb_border_styles (set-multiple), 97
map_tb_borders (set-multiple), 97
map_tb_padding (set-multiple), 97
map_text_color (text_color), 109
map_top_border (borders), 18
map_top_border_color (border-colors), 16
map_top_border_style (border-styles), 17
map_top_padding (padding), 79
map_valign (valign), 114
map_wrap (wrap), 116
mapping-functions, 8, 14–16, 18, 19, 22–26, 28, 29, 36, 39, 40, 49, 69, 71, 76, 77, 80, 92, 98, 100, 102, 109, 114, 116
mapping_functions (mapping-functions), 69
markdown, 71
markdown(), 36, 102
markdown<- (markdown), 71
merge_across, 72, 74, 75
merge_across(), 103
merge_cells, 73, 73, 75
merge_cells(), 73–75, 103
merge_down (merge_across), 72
merge_down(), 103
mergeRepeated_rows, 73, 74, 75
mutate (mutate.huxtable), 75
mutate.huxtable, 75
na_string, 14, 15, 39, 41, 76, 78, 110
na_string<- (na_string), 76
nobs.tidy_override (tidy_override), 112
number_format, 14, 15, 39, 41, 77, 77, 110
number_format(), 8, 37, 38, 44, 47, 48
number_format<- (number_format), 77
odds (stripe), 104
odds(), 94
openxlsx::openxlsx(), 12
openxlsx::saveWorkbook(), 13
padding, 79, 98
position, 80
position(), 31, 108
position<- (position), 80
prettyNum(), 38
print.huxtable, 81
print.huxtable(), 62
print.html, 82, 83, 84, 86, 87
print_latex, 82, 83, 84, 86, 87
print_md, 82, 83, 84, 86, 87
print_md(), 72, 102
print_notebook (print_html), 82
print_rtf, 82–84, 85, 87
print_rtf(), 95
print_screen, 82–84, 86, 86
print_screen(), 81
quick-output, 87
quick_docx (quick-output), 87
quick_html (quick-output), 87
quick_latex (quick-output), 87
quick_pdf (quick-output), 87
quick_pdf(), 63
quick_pptx (quick-output), 87
quick_rtf (quick-output), 87
quick_xlsx (quick-output), 87
rbind.huxtable (cbind.huxtable), 32
rbind.huxtable(), 7
regular expression, 27
report_latex_dependencies, 89
report_latex_dependencies(), 48
restack_across_down, 90, 104
restack_down(restack_across-down), 90
restack_across-down, 90
restacking, 42
right_border(borders), 18
right_border<- (borders), 18
right_border_color(border-colors), 16
right_border_color<-(border-colors), 16
right_border_style(border-styles), 17
right_border_style<-(border-styles), 17
right_padding (padding), 79
right_padding<-(padding), 79
rotation, 92
rotation() , 85
rotation<- (rotation), 92
row_height, 35, 43, 94, 115
row_height<-(row_height), 94
rowspan (spans), 102
rowspan() , 73-75, 103
rowspan<-(spans), 102
rowspecs, 8, 14-16, 18, 19, 35-37, 39, 40, 42, 69, 71, 73, 74, 76, 77, 79, 92, 93, 94, 98-103, 105, 106, 109, 114, 116
rtf_fc_tables, 95
rtf_fc_tables(), 85
sanitize, 96
sanitize(), 36
scipen, 48
set-multiple, 17, 18, 20, 80, 97
set_outer, 80, 99
set_align (align), 8
set_all_border_colors(set-multiple), 97
set_all_border_styles(set-multiple), 97
set_all_borders(set-multiple), 97
set_all_padding(set-multiple), 97
set_background_color
  (background_color), 14
set_bold (bold), 15
set_bottom_border (borders), 18
set_bottom_border_color
  (border-colors), 16
set_bottom_border_style
  (border-styles), 17
set_bottom_padding (padding), 79
set_caption (caption), 30
set_caption_pos (caption_pos), 31
set_caption_width (caption_width), 32
set_cell_properties(style-functions), 106
set_cell_properties(), 6
set_col_width(col_width), 35
set_collapse (spans), 102
set_contents, 100
set_contents(), 101
set_default_properties, 100
set_escape_contents (escape_contents), 36
set_font(font), 39
set_font_size(font_size), 40
set_header_cols (header_cols), 42
set_header_rows (header_cols), 42
set_height (height), 43
set_italic (bold), 15
set_label (label), 67
set_latex_float (latex_float), 68
set_left_border (borders), 18
set_left_border_color (border-colors), 16
set_left_padding (padding), 79
set_left_padding_color (set-multiple), 97
set_left_padding_color (set-multiple), 97
set_left_borders(set-multiple), 97
set_left_padding(set-multiple), 97
set_markdown (markdown), 71
set_markdown(), 101
set_markdown_contents, 101
set_markdown_contents(), 72
set_na_string (na_string), 76
set_number_format (number_format), 77
set_outer_border_colors (set-outer), 99
set_outer_border_styles (set-outer), 99
set_outer_borders(set-outer), 99
set_outer_padding(set-outer), 99
set_position (position), 80
set_right_border (borders), 18
set_right_border_color (border-colors), 16
set_right_border_style (border-styles), 17
set_right_padding (padding), 79
set_rotation (rotation), 92
set_row_height (row_height), 94
set_rowspan (spans), 102
set_table_environment
  (table_environment), 108
set_tabular_environment
  (tabular_environment), 108
set_tb_border_colors (set-multiple), 97
set_tb_border_styles (set-multiple), 97
set_tb_borders (set-multiple), 97
set_tb_padding (set-multiple), 97
set_text_color (text_color), 109
set_top_border (borders), 18
spans, 102
split-across-down, 91, 103
split_across (split-across-down), 103
split_down (split-across-down), 103
split_across (split-across-down), 103
split_across_down (split_across_down), 103
split_down (split_across_down), 103
sprint(), 77
stripe, 104
stripe(), 94
stripe_columns (stripes), 105
stripe_rows (stripes), 105
stripes, 105
style-functions, 106
style_cells (style-functions), 106
style_header_cols (style-functions), 106
style_header_rows (style-functions), 106
style_headers (style-functions), 106
style_headers(), 42
t.huxtable, 107
table_environment, 108
table_environment<-
  (table_environment), 108
tabular_environment, 108
tabular_environment<-
  (tabular_environment), 108
text_color, 14, 15, 39, 41, 77, 78, 109
text_color<-(text_color), 109
theme_article (themes), 110
theme_basic (themes), 110
theme_blue (themes), 110
theme_bright (themes), 110
theme_compact (themes), 110
theme_green (themes), 110
theme_grey (themes), 110
theme_monterrey (themes), 110
theme_orange (themes), 110
theme_plain (themes), 110
theme_plain(), 63, 66
theme_striped (themes), 110
themes, 110
tibble::tribble(), 46
tidy.tidy_override (tidy_override), 112
tidy_override, 112
tidy_override(), 45
tidy_replace (tidy_override), 112
tidyselect, 103
tidyselect::language, 93
tidyselect::with_vars(), 94
tinytex::tlmgr_install(), 89
to_html (print_html), 82
to_html(), 81
to_latex (print_latex), 83
to_latex(), 81
to_md (print_md), 84
to_rtf (print_rtf), 85
to_rtf(), 95
to_screen (print_screen), 86
top_border (borders), 18
top_border<-(borders), 18
top_border_color (border-colors), 16
top_border_color<-(border-colors), 16
top_border_style (border-styles), 17
top_border_style<-(border-styles), 17
top_padding (padding), 79
top_padding<-(padding), 79
ttribble_hux (huxtable), 46
valign, 114
valign<-(valign), 114
width, 35, 43, 95, 115
width(), 13, 48, 72, 85, 116
width<-(width), 115
wrap, 116
wrap(), 48, 72, 85
wrap<-(wrap), 116