Package 'huxtable'

October 3, 2019

Type Package

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

Version 4.7.0

Author David Hugh-Jones [aut, cre]

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

Description Like 'xtable', creates styled tables. Export to HTML, LaTeX, 'Word', 'Excel', 'PowerPoint' and RTF. 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, generics, glue, memoise, rlang, stats, stringr (>= 1.2.0), tibble, tidyselect, utils

Suggests bookdown, broom (>= 0.5.1), broom.mixed, covr, crayon, devtools, dplyr, flextable, ggplot2, httr, knitr, lazyeval, lme4, lmtest, nlme, nnet, officer, openxlsx, psych, rmarkdown, sandwich, scales, testthat, tinytex

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-10-03 05:00:10 UTC

R topics documented:

`huxtable-package` *Quick introduction to huxtable*

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()`:

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

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

```
# 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:

```
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.

Author(s)

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

See Also

Useful links:

- https://hughjonesd.github.io/huxtable
- Report bugs at https://github.com/hughjonesd/huxtable/issues

add_colnames *Add column or row names*

Description

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

Usage

```
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

```
ht <- huxtable(
        First  = rnorm(5),
        Second = rnorm(5)
      )
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

```
add_footnote(ht, text, border = 0.8, ...)
```

Arguments

`ht`	A huxtable.
`text`	Text for the footnote.
`border`	Width of the footnote's top border. Set to 0 for no border.
`...`	Other properties, passed to `set_cell_properties()` for the footnote cell.

Value

The modified huxtable

Examples

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

add_rows *Insert one matrix into another.*

Description

These functions combine two matrix-like objects and return the result.

Usage

```
add_rows(x, y, after = nrow(x), ...)

add_columns(x, y, after = ncol(x), ...)
```

Arguments

x	A matrix-like object, e.g. a huxtable
y	Matrix or vector to be inserted into x
after	Row or column after which y is inserted. Can be 0. Can be a row or column name. By default, inserts y after the end of x.
...	Arguments passed to rbind() or cbind()

Details

For huxtable objects, arguments in ... can include copy_cell_props.

Value

For add_rows, the result of rbind(x[1:after,],y,x[-(1:after),]. For add_columns the same but with columns. after = 0 and after = nrow(x) or ncol(x) are handled correctly.

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` *Alignment*

Description

Functions to get or set the *alignment* property of huxtable cells.

Usage

```
align(ht)
align(ht) <- value
set_align(ht, row, col, value, byrow = FALSE)
map_align(ht, row, col, fn)
```

Arguments

`ht`	A huxtable.
`value`	A character vector or matrix which may be "left", "center", "right" , `NA` or a single character. Set to `NA` to reset to the default, which is `"left"`.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Details

This sets the horizontal alignment of the cell. If `value` is a single character (e.g. a decimal point), then the cell is aligned on this character.

Value

For `align`, the `align` property. For `set_align` and `map_align`, the modified huxtable.

Examples

```
align(jams) <-  "right"
align(jams)


set_align(jams, "right")
set_align(jams,
      2:3, 1, "right")
map_align(jams,
      by_rows("right", "left"))
```

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

```
as_FlexTable(x, ...)

as_flextable(x, ...)

## 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.)

`as_FlexTable` is deprecated and calls `as_flextable` with a warning.

Properties are supported, with the following exceptions:

- Rotation of 0, 90 or 270 is supported.
- Non-numeric column widths and row heights are not supported.
- Table height, wrap, captions and table position are not supported.
- Border style "double" is not supported and becomes "solid".

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, tables, ftables, matrices and (most) vectors. is_hux[table] tests if an object is a huxtable.

Usage

```
as_huxtable(x, ...)

as_hux(x, ...)

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

is_huxtable(x)

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.

Value

An object of class "huxtable".

Examples

```
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)
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")
```

---

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, ...)
```

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.

Details

Use `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.

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

Contents are only stored as numbers if a whole column is numeric as defined by `is_a_number()`; otherwise they are stored as text.

Value

An object of class `Workbook`.

Examples

```
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")
```

---

background_color *Background color*

Description

Functions to get or set the *background color* property of huxtable cells.

Usage

```
background_color(ht)
background_color(ht) <- value
set_background_color(ht, row, col, value, byrow = FALSE)
map_background_color(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A character vector or matrix of valid R color names. Set to NA to reset to the default, which is NA.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Value

For background_color, the background_color property. For set_background_color and map_background_color, the modified huxtable.

See Also

Other formatting functions: bold, font_size, font, na_string, number_format, text_color

Examples

```
background_color(jams) <-  grey(.95)
background_color(jams)


set_background_color(jams, grey(.95))
set_background_color(jams,
      2:3, 1, grey(.95))
map_background_color(jams,
      by_rows(grey(.95), "yellow"))
```

---

bold *Cell text style*

Description

Functions to get or set the *cell text style* property of huxtable cells.

Usage

```
bold(ht)
bold(ht) <- value
set_bold(ht, row, col, value, byrow = FALSE)
map_bold(ht, row, col, fn)

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

Arguments

ht	A huxtable.
value	A logical vector or matrix. TRUE for bold/italic. Set to NA to reset to the default, which is FALSE.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Value

For bold, the bold property. For set_bold and map_bold, the modified huxtable.

Similarly for italic and friends.

See Also

Other formatting functions: background_color, font_size, font, na_string, number_format, text_color

Examples

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


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

by_cases *Map cell contents to properties using* case_when

Description

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

Usage

```
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. Otherwise, NA normally resets to the default.

Details

Within the formulas, the variable . will refer to the content of ht[rows,cols] (converted 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.

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

```
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

Map numeric cell contents smoothly to colors

Usage

```
by_colorspace(..., range = NULL, na_color = NA, ignore_na = TRUE)
```

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. Otherwise, NA normally resets to the default.

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"))
```

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_xxx. Useful functions include scales and palettes from the scales package.

Usage

```
by_function(inner_fn, ignore_na = TRUE)
```

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. 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,
  ignore_na = TRUE)

by_equal_groups(n, values, ignore_na = TRUE)
```

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. Otherwise, `NA` normally resets to the default.
`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_colorspace`, `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", "white", "green")
      ))

map_background_color(ht,
      by_equal_groups(
        3,
        c("red", "yellow", "green")
      ))
```

---

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. 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

```
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

Map cells matching a string or regex to cell properties

Usage

```
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. 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_colorspace, by_function, by_quantiles, by_ranges, by_rows, by_values

Examples

```
ht <- hux(c("The cat sat", "on the", "mat"))

map_bold(ht, by_regex("at" = TRUE))
map_bold(ht, by_regex("a.*a" = TRUE))

map_bold(ht, by_regex(
        "the" = TRUE,
        .grepl_args = list(
          ignore.case = TRUE
        )
      ))
```

---

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. 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_colorspace, 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

Map specific cell values to cell properties

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. 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_colorspace, by_function, by_quantiles, by_ranges, by_regex, by_rows

Examples

```
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 *Caption*

Description

Functions to get or set the table-level *caption* property of a huxtable.

Usage

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

Arguments

ht	A huxtable.
value	A length-one character vector. Set to NA to reset to the default, which is NA.

Details

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

Value

For caption, the caption property. For set_caption, the modified huxtable.

See Also

caption_pos()

Examples

```
caption(jams) <-  "An example table"
caption(jams)
jams

# escape caption characters:
caption(jams) <- sanitize(
      "Make $$$ with jam",
      type = "latex")
```

`caption_pos` *Caption position*

Description

Functions to get or set the table-level *caption position* property of a huxtable.

Usage

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

Arguments

`ht`	A huxtable.
`value`	A length-one character vector, one of "top", "bottom", "topleft", "topcenter", "topright", "bottomleft", "bottomcenter", "bottomright". Set to `NA` to reset to the default, which is `"top"`.

Details

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

Value

For `caption_pos`, the `caption_pos` property. For `set_caption_pos`, the modified huxtable.

See Also

`caption()`

Examples

```
caption(jams) <- "Price list"
jams
caption_pos(jams) <- "top"
jams
```

`cbind.huxtable` *Combine rows or columns*

Description

Combine rows or columns

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 a character vector of cell properties, then the named cell properties will be copied to non-huxtables. Objects on the left or above get priority over those on the right or below. These properties may also include `"row_height"` (for rbind) or `"col_width"` (for cbind). Numeric row heights and column widths will be rescaled to 1.

If `copy_cell_props` is `TRUE`, the default set of cell properties (everything but `colspan` and `rowspan`, including row heights/column widths) will be copied.

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

NB: 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

```
ht1 <- hux(a = 1:3, b = 4:6)
ht2 <- hux(
        d = letters[1:3],
        e = letters[4:6]
      )
bold(ht1)[1, ] <- TRUE
bold(ht2) <- TRUE
vec <- LETTERS[1:3]

cbind(ht1, vec, ht2)
cbind(ht1, vec, ht2,
      copy_cell_props = FALSE)
```

---

col_width *Column widths*

Description

Functions to get or set the *column widths* property of huxtable cols.

Usage

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

Arguments

ht	A huxtable.
value	A vector. If numeric, they are treated as proportions of the table width. If character, they must be valid CSS or LaTeX lengths.
col	A col specifier. See rowspecs for details.

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

For `col_width`, the `col_width` property. For `set_col_width`, the modified huxtable.

See Also

Other row/column heights: `row_height`

Examples

```
col_width(jams) <-  c(.2, .8)
col_width(jams)
```

`escape_contents` *Escape cell contents*

Description

Functions to get or set the *escape cell contents* property of huxtable cells.

Usage

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

Arguments

`ht`	A huxtable.
`value`	A logical vector or matrix. If `TRUE`, cell contents will be HTML or LaTeX escaped. Set to `NA` to reset to the default, which is `TRUE`.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Value

For `escape_contents`, the `escape_contents` property. For `set_escape_contents` and `map_escape_contents`, the modified huxtable.

Examples

```
ht <- huxtable(
        Exponent = 2:4,
        Example  = paste0("$x^", 2:4, "$"),
        add_colnames = TRUE
      )
escape_contents(ht)[,2] <- FALSE
## Not run:
  quick_pdf(ht)
```

```
## End(Not run)


jams2 <- set_escape_contents(jams,
      TRUE)
escape_contents(jams2)

jams3 <- set_escape_contents(jams,
      2:3, 1, TRUE)
escape_contents(jams3)

jams4 <- map_escape_contents(jams,
      by_rows(
        TRUE,
        FALSE)
      )
escape_contents(jams4)
```

`every` *Return every n row or column numbers*

Description

This is a convenience function to use in row or column specifications. In this context, `every(n,from)` will return `from,from + n,...,` up to the number of rows or columns of the huxtable. `evens` and `odds` return even and odd numbers, i.e. they are equivalent to `every(2,2)` and `every(2,1)` respectively. `everywhere` returns all rows or columns, equivalently to `every(1)`.

Usage

```
every(n = 1, from = n)

everywhere(ht, dimension)

evens(ht, dimension)

odds(ht, dimension)
```

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

Examples

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

`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)
```

font *Font*

Description

Functions to get or set the *font* property of huxtable cells.

Usage

```
font(ht)
font(ht) <- value
set_font(ht, row, col, value, byrow = FALSE)
map_font(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A character vector of font names. Set to NA to reset to the default, which is NA.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

LaTeX and HTML use different font names. If you want to use the same font names across document formats, set options("huxtable.latex_use_fontspec") to TRUE. See huxtable-options.

Value

For font, the font property. For set_font and map_font, the modified huxtable.

See Also

Other formatting functions: background_color, bold, font_size, na_string, number_format, text_color

Examples

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


jams2 <- set_font(jams,
      "times")
```

```
font(jams2)

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

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

`font_size` *Font size*

Description

Functions to get or set the *font size* property of huxtable cells.

Usage

```
font_size(ht)
font_size(ht) <- value
set_font_size(ht, row, col, value, byrow = FALSE)
map_font_size(ht, row, col, fn)
```

Arguments

`ht`	A huxtable.
`value`	A numeric vector. This sets the font size in points. Set to `NA` to reset to the default, which is `NA`.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Value

For `font_size`, the `font_size` property. For `set_font_size` and `map_font_size`, the modified huxtable.

See Also

Other formatting functions: `background_color`, `bold`, `font`, `na_string`, `number_format`, `text_color`

Examples

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


jams2 <- set_font_size(jams,
      14)
font_size(jams2)

jams3 <- set_font_size(jams,
      2:3, 1, 14)
font_size(jams3)

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

---

guess_knitr_output_format

Guess knitr output format

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

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

## End(Not run)
```

height *Table height*

Description

Functions to get or set the table-level *table height* property of a huxtable.

Usage

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

Arguments

ht	A huxtable.
value	A length-one vector. If numeric, it is treated as a proportion of the containing block height for HTML, or of text height (`\textheight`) for LaTeX. If character, it must be a valid CSS or LaTeX width. Set to `NA` to reset to the default, which is `NA`.

Value

For `height`, the `height` property. For `set_height`, the modified huxtable.

See Also

Other table measurements: `width`

Examples

```
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_style = c("stderr",
  "ci", "statistic", "pvalue"), error_pos = c("below", "same", "right"),
  number_format = "%.3f", align = ".", pad_decimal = ".",
  ci_level = NULL, tidy_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_style`	Deprecated. One or more of "stderr", "ci" (confidence interval), "statistic" or "pvalue".
`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.
`pad_decimal`	Deprecated in favour of `align`.
`ci_level`	Confidence level for intervals. Set to `NULL` to not calculate confidence intervals.
`tidy_args`	List of arguments to pass to `broom::tidy()`. You can also pass a list of lists; if so, the nth element will be used for the nth column.
`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

```
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)
```

---

huxtable *Create a huxtable*

Description

huxtable, or hux, creates a huxtable object.

Usage

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

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

tribble_hux(..., add_colnames = getOption("huxtable.add_colnames",
  FALSE), add_rownames = FALSE,
  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` currently defaults to `FALSE`, but this will change in future. 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.

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

```
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

- LaTeX output isn't working.

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()`.

- 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:

```
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.

- My tables aren't centered correctly (LaTeX).

Try adjusting `width(ht)`.

- 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 something like:

```
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 have another problem.

If you have a bug - i.e. a problem with the software - or have a feature request, please report it to https://github.com/hughjonesd/huxtable/issues. Otherwise, ask a question on StackOverflow or 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-options` *Package options*

Description

- `options('huxtable.add_colnames')` sets the default value for `add_colnames` in `huxtable()` and `as_huxtable()`. If it is unset, `add_colnames` defaults to `FALSE`; in a future release, the default will become `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.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` (the default), data frames in knitr will be pretty-printed using huxtable.
- `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 the xetex or xelatex engine, which can be set using an .rmd header option. Note that `quick_pdf()` may use pdflatex. It defaults to `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(...,Date = "center",...)`.

---

hux_hex *Deprecated functions*

Description

These functions are deprecated and will be removed in future versions of huxtable.

Usage

```
hux_hex()

hex_hux()
```

Details

To replace `pad_decimal` use `align()`, e.g. `align(ht) <-"."`.

To replace `is_a_number` use e.g. `! is.na(as.numeric(x))`

To replace the 3 argument form of `set_xxx` functions, use `map_xxx`.

hux_logo *Huxtable 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())
```

insert_column *Insert a row or column*

Description

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

Usage

```
insert_column(ht, ..., after = 0, fill = NULL, rowspan = 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. this. 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

```
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
)
```

`jams` *Prices of 3 jams*

Description

A huxtable of jams.

Usage

```
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

```
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

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

options(huxtable.knit_print_df_theme
      = mytheme)
# groovy!
data.frame(
        a = 1:5,
        b = 1:5
      )

## End(Not run)
```

---

knit_print.huxtable *Print a huxtable within knitr*

Description

Print a huxtable within knitr

Usage

```
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 *Table label*

Description

Functions to get or set the table-level *table label* property of a huxtable.

Usage

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

Arguments

ht	A huxtable.
value	A length-one character vector to be used as a table label in LaTeX, or as an ID for the table in HTML. Set to NA to reset to the default, which is NA.

Details

LaTeX table labels typically start with "tab:".

If you use bookdown, and set a label on your table, the caption will automatically be prefixed with (#label). You can then refer to the table using @ref(label). label needs to start with "tab:"; if it doesn't, the "tab" prefix will be added automatically.

Value

For label, the label property. For set_label, the modified huxtable.

Examples

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

latex_float *Float position for LaTeX*

Description

Functions to get or set the table-level *float position for latex* property of a huxtable.

Usage

```
latex_float(ht)
latex_float(ht) <- value
set_latex_float(ht, value)
```

Arguments

ht	A huxtable.
value	A length-one character vector, used by LaTeX for positioning the float. Set to NA to reset to the default, which is "h".

Details

Quick reference: "h" here, "h!" definitely here, "t" top of page, "b" bottom of page, "p" page of floats. See LaTeX documentation for more details. If you use "H" (definitely here), you must require the TeX `float` package.

Value

For `latex_float`, the `latex_float` property. For `set_latex_float`, the modified huxtable.

Examples

```
latex_float(jams) <-  "h"
latex_float(jams)
```

left_border *Borders*

Description

Functions to get or set the *borders* property of huxtable cells.

Usage

```
left_border(ht)
left_border(ht) <- value
set_left_border(ht, row, col, value, byrow = FALSE)
map_left_border(ht, row, col, fn)

right_border(ht)
right_border(ht) <- value
set_right_border(ht, row, col, value, byrow = FALSE)
map_right_border(ht, row, col, fn)

top_border(ht)
top_border(ht) <- value
set_top_border(ht, row, col, value, byrow = FALSE)
map_top_border(ht, row, col, fn)

bottom_border(ht)
bottom_border(ht) <- value
set_bottom_border(ht, row, col, value, byrow = FALSE)
map_bottom_border(ht, row, col, fn)
```

Arguments

`ht`	A huxtable.
`value`	A numeric vector or matrix giving border widths in points. Set to 0 for no border. Set to `NA` to reset to the default, which is `0`.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Details

Currently in LaTeX, all non-zero border widths on a given line must be the same, and vertical border widths can only be present (if `value > 0`) or absent.

Value

For `left_border`, the `left_border` property. For `set_left_border` and `map_left_border`, the modified huxtable.

Similarly for the other functions.

Note

huxtable currently sets borders on specific cells. This can lead to surprising behaviour when cells span multiple rows or columns: see the example. This behaviour may be improved in a future release.

See Also

set_all_borders()

Examples

```
left_border(jams) <-  1
left_border(jams)
jams

set_left_border(jams, 1)
set_left_border(jams,
      2:3, 1, 1)
map_left_border(jams,
      by_rows(1, 0))
# When cells span multiple rows:
ht <- tribble_hux(
  ~Col1,                   ~Col2,
  "Cell 1,1 spans 2 rows", "Cell 1,2",
  "Cell 2,1 is invisible", "Cell 2,2"
)

rowspan(ht)[1, 1] <- 2
ht

bottom_border(ht)[2, ] <- 1
bottom_border_color(ht)[2, ] <- 'red'

# Cell 1, 1 does not have a border set:
ht

# Fixed:
bottom_border(ht)[1, 1] <- 1
bottom_border_color(ht)[1, 1] <- 'red'
ht
```

---

left_border_color *Border colors*

Description

Functions to get or set the *border colors* property of huxtable cells.

Usage

```
left_border_color(ht)
left_border_color(ht) <- value
set_left_border_color(ht, row, col, value, byrow = FALSE)
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, byrow = FALSE)
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, byrow = FALSE)
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, byrow = FALSE)
map_bottom_border_color(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A vector or matrix of colors. Set to NA to reset to the default, which is NA.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

Huxtable collapses borders and border colors. Right borders take priority over left borders, and top borders take priority over bottom borders.

Value

For left_border_color, the left_border_color property. For set_left_border_color and map_left_border_color, the modified huxtable.

Similarly for the other functions.

Note

huxtable currently sets borders on specific cells. This can lead to surprising behaviour when cells span multiple rows or columns: see the example. This behaviour may be improved in a future release.

See Also

set_all_border_colors()

Examples

```
ht <- huxtable(a = 1:3, b = 3:1)
ht <- set_all_borders(ht, 1)
set_left_border_color(ht, "red")
set_left_border_color(ht,
      1:2, 1, "red")

# When cells span multiple rows:
ht <- tribble_hux(
  ~Col1,                    ~Col2,
  "Cell 1,1 spans 2 rows", "Cell 1,2",
  "Cell 2,1 is invisible", "Cell 2,2"
)

rowspan(ht)[1, 1] <- 2
ht

bottom_border(ht)[2, ] <- 1
bottom_border_color(ht)[2, ] <- 'red'

# Cell 1, 1 does not have a border set:
ht

# Fixed:
bottom_border(ht)[1, 1] <- 1
bottom_border_color(ht)[1, 1] <- 'red'
ht
```

---

left_border_style *Border styles*

Description

Functions to get or set the *border styles* property of huxtable cells.

Usage

```
left_border_style(ht)
left_border_style(ht) <- value
set_left_border_style(ht, row, col, value, byrow = FALSE)
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, byrow = FALSE)
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, byrow = FALSE)
map_top_border_style(ht, row, col, fn)

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

Arguments

ht	A huxtable.
value	A character vector or matrix of styles, which may be "solid", "double", "dashed" or "dotted". Set to NA to reset to the default, which is "solid".
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

Huxtable collapses borders and border colors. Right borders take priority over left borders, and top borders take priority over bottom borders.

Border styles only apply if the border width is greater than 0.

Value

For left_border_style, the left_border_style property. For set_left_border_style and map_left_border_style, the modified huxtable.

Similarly for the other functions.

Quirks

- 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.

Note

huxtable currently sets borders on specific cells. This can lead to surprising behaviour when cells span multiple rows or columns: see the example. This behaviour may be improved in a future release.

Examples

```
ht <- huxtable(a = 1:3, b = 3:1)
ht <- set_all_borders(ht, 1)
set_left_border_style(ht, "double")
set_left_border_style(ht, 1:2, 1,
      "double")

# When cells span multiple rows:
ht <- tribble_hux(
  ~Col1,                    ~Col2,
  "Cell 1,1 spans 2 rows", "Cell 1,2",
  "Cell 2,1 is invisible", "Cell 2,2"
)

rowspan(ht)[1, 1] <- 2
ht

bottom_border(ht)[2, ] <- 1
bottom_border_color(ht)[2, ] <- 'red'

# Cell 1, 1 does not have a border set:
ht

# Fixed:
bottom_border(ht)[1, 1] <- 1
bottom_border_color(ht)[1, 1] <- 'red'
ht
```

---

left_padding *Cell padding*

Description

Functions to get or set the *cell padding* property of huxtable cells.

Usage

```
left_padding(ht)
left_padding(ht) <- value
set_left_padding(ht, row, col, value, byrow = FALSE)
map_left_padding(ht, row, col, fn)

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

top_padding(ht)
top_padding(ht) <- value
```

```
set_top_padding(ht, row, col, value, byrow = FALSE)
map_top_padding(ht, row, col, fn)

bottom_padding(ht)
bottom_padding(ht) <- value
set_bottom_padding(ht, row, col, value, byrow = FALSE)
map_bottom_padding(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A vector or matrix. Characters must be valid CSS or LaTeX lengths. Numbers will be interpreted as lengths in points. Set to NA to reset to the default, which is 4.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Value

For left_padding, the left_padding property. For set_left_padding and map_left_padding, the modified huxtable.

Similarly for the other functions.

Examples

```
left_padding(jams) <-  20
left_padding(jams)


jams2 <- set_left_padding(jams,
      20)
left_padding(jams2)

jams3 <- set_left_padding(jams,
      2:3, 1, 20)
left_padding(jams3)

jams4 <- map_left_padding(jams,
      by_rows(
        20,
        10)
      )
left_padding(jams4)
```

mapping-functions *How to set cell properties variably by cell contents*

Description

This help page explains how to set properties differently for cells, depending on their contents.

Details

For example, in a table of p-values, you could bold cells where p < 0.05:

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

Or you can use red text for a particular value:

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

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

```
map_xxx(ht, row, col, fn)
```

where `xxx` is the property name.

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

```
map_xxx(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 `by_values()`.
- To set property values for cells within a numeric range, use `by_ranges()`.
- To set property values for cells by quantiles, use `by_quantiles()` or `by_equal_groups()`.
- To set property values for cells that match a string or regular expression, use `by_regex()`.
- To map numeric values to a colorspace, use `by_colorspace()`.
- For a more general solution, use `by_function()` or `by_cases()`.

Value

The modified huxtable.

Caveat

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

Technical details

`fn` must be a function taking four arguments: the (entire) original huxtable `ht`, a numeric vector of `rows`, a numeric vector of `cols`, and the `current` property values for `ht[rows,cols]`, as a matrix. It should return the new property values for `ht[rows,cols]`, as a matrix.

Here's an example. Suppose we want to highlight cells of a correlation matrix with low p values:

```
data(attitudes)
att_corr <- psych::corr.test(attitudes)
# att_corr has components r (correlations) and p (p values)

corr_hux <- as_hux(att_corr$r)
by_p_value <- function (ht, rows, cols, current) {
   result <- current
   pvals <- att_corr$p[rows, cols]
   result[pvals < 0.01] <- "red"
   result[pvals < 0.05] <- "orange"
   result
}
```

Examples

```
ht <- hux(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 <- hux(rnorm(5), rnorm(5), rnorm(5))
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",
      by_ranges(0.05, c(TRUE, FALSE)))

# Problems with as.matrix:

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

`merge_cells` *Merge a range of cells*

Description

Merge a range of cells

Usage

```
merge_cells(ht, row, col)
```

Arguments

`ht`	A huxtable.
`row`	A row specifier. See `rowspecs` for details. Only the minimum and maximum rows and columns are used.
`col`	A column specifier.

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.

See Also

merge_repeated_rows

Examples

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

`merge_repeated_rows` *Merge repeated rows into multirow cells*

Description

Merge repeated rows into multirow cells

Usage

```
merge_repeated_rows(ht, row, col)
```

Arguments

`ht`	A huxtable.
`row`	A row specification.
`col`	A column specification.

Details

Repeated rows in each column are merged into cells with `rowspan > 1`.

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

Value

The modified huxtable.

See Also

merge_cells

Examples

```
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")
```

mutate.huxtable *Dplyr verbs for huxtable*

Description

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

Usage

```
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

```
ht <- hux(a = 1:5, b = 1:5, c = 1:5, d = 1:5)
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 *NA string*

Description

Functions to get or set the *na string* property of huxtable cells.

Usage

```
na_string(ht)
na_string(ht) <- value
set_na_string(ht, row, col, value, byrow = FALSE)
map_na_string(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A character string. This will be used to replace NA values in the display. Set to NA to reset to the default, which is "".
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Value

For na_string, the na_string property. For set_na_string and map_na_string, the modified huxtable.

See Also

Other formatting functions: background_color, bold, font_size, font, number_format, text_color

Examples

```
na_string(jams) <-  "--"
na_string(jams)
jams[2,2] <- NA
jams

jams2 <- set_na_string(jams,
      "--")
na_string(jams2)

jams3 <- set_na_string(jams,
      2:3, 1, "--")
```

```
na_string(jams3)

jams4 <- map_na_string(jams,
      by_rows(
        "--",
        "")
      )
na_string(jams4)
```

`number_format` *Number format*

Description

Functions to get or set the *number format* property of huxtable cells.

Usage

```
number_format(ht)
number_format(ht) <- value
set_number_format(ht, row, col, value, byrow = FALSE)
map_number_format(ht, row, col, fn)
```

Arguments

`ht`	A huxtable.
`value`	A character or integer vector, or a list containing a function, or `NA`. Note that setting to `NA` does not reset to the default.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Details

Number formatting is applied to any parts of cells that look like numbers (defined as an optional minus sign, followed by numerals, followed by an optional decimal point and further numerals). The exception is exponents in scientific notation; huxtable attempts to detect and ignore these.

If `value` is

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

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.

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.

To set number_format to a function, enclose the function in `list`. The function should take one argument and return a string.

Versions of huxtable before 2.0.0 applied `number_format` only to cells that looked like numbers in their entirety. The default value was "%5.2f".

Value

For `number_format`, the `number_format` property. For `set_number_format` and `map_number_format`, the modified huxtable.

See Also

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),
        add_colnames = TRUE
      )

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

number_format(ht)[6, -1] <- list(
        function(x)
          prettyNum(x, big.mark = ",",
                scientific = FALSE)
      )

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)
# alternatively:
huxtable("10000 Maniacs", autoformat = TRUE)


set_number_format(jams, 2)
set_number_format(jams,
      2:3, 1, 2)
map_number_format(jams,
      by_rows(2, 3))
```

`position` *Table position*

Description

Functions to get or set the table-level *table position* property of a huxtable.

Usage

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

Arguments

`ht`	A huxtable.
`value`	A length-one character vector which may be "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.

If your tables are too far to the right under LaTeX, try setting their `width()` explicitly.

Value

For `position`, the `position` property. For `set_position`, the modified huxtable.

Examples

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

`print.huxtable` *Default print method for huxtables*

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

```
## 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

Examples

```
## Not run:
  # to print LaTeX output:
  options(huxtable.print = print_latex)

## End(Not run)

format(jams, output = "screen")
format(jams, output = "md")
```

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)
```

print_md *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 rmarkdown 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: print_html, print_latex, print_md, print_screen

Examples

```
print_rtf(jams)
```

---

print_screen *Print a huxtable on screen*

Description

Print a huxtable on screen

Usage

```
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), ...)
```

Arguments

ht	A huxtable.
...	Passed on to to_screen.
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.
compact	Logical. To save space, don't print lines for empty horizontal borders.
colnames	Logical. Whether or not to print colum names.
color	Logical. Whether to print the huxtable in color (requires the crayon package).

Details

colspan, rowspan, align and caption properties are shown. So are borders (but not border styles). If the crayon package is installed, output will be colorized (and contents bolded or italicized) by default; this will work in recent daily builds of RStudio as of October 2017.

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

```
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)
```

`quick-output` *Quickly print objects to a PDF, TeX, HTML, Microsoft Office or RTF document.*

Description

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

Usage

```
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.
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

```
## 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

Tools for LaTeX dependencies

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 the LaTeX packages that huxtable requires. It calls tinytex::tlmgr_install() if possible, or tlmgr install directly.

Usage

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

check_latex_dependencies(quiet = FALSE)

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

```
report_latex_dependencies()

## Not run:
  check_latex_dependencies()

## End(Not run)

## Not run:
  install_latex_dependencies()

## End(Not run)
```

---

rotation *Text rotation*

Description

Functions to get or set the *text rotation* property of huxtable cells.

Usage

```
rotation(ht)
rotation(ht) <- value
set_rotation(ht, row, col, value, byrow = FALSE)
map_rotation(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A numeric vector. Anti-clockwise from the x axis, so 0 is left to right, 90 is going up, etc. Set to NA to reset to the default, which is 0.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

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

Value

For rotation, the rotation property. For set_rotation and map_rotation, the modified huxtable.

Examples

```
rotation(jams) <-  90
rotation(jams)


jams2 <- set_rotation(jams,
      90)
rotation(jams2)

jams3 <- set_rotation(jams,
      2:3, 1, 90)
rotation(jams3)

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

rowspan *Row and column span*

Description

Functions to get or set the *row and column span* property of huxtable cells.

Usage

```
rowspan(ht)
rowspan(ht) <- value
set_rowspan(ht, row, col, value, byrow = FALSE)
map_rowspan(ht, row, col, fn)

colspan(ht)
colspan(ht) <- value
set_colspan(ht, row, col, value, byrow = FALSE)
map_colspan(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	An integer vector or matrix. Set to NA to reset to the default, which is 1.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

The rowspan and colspan of a cell determine its height and width, in rows and columns. A cell with rowspan of 2 covers the cell directly below it. A cell with rowspan of 2 and colspan of 2 covers a 2 x 2 square, hiding three other cells.

Value

For rowspan, the rowspan property. For set_rowspan and map_rowspan, the modified huxtable.

Examples

```
rowspan(jams)[1, 1] <-  2
rowspan(jams)
jams <- set_all_borders(jams, 1)
jams
```

rowspecs *Different ways to select rows and columns*

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,byrow = FALSE).

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 every(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 select_helpers for a full list.
- Set byrow = TRUE to set properties by row rather than by column.

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 (excluding byrow) 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(), every() and final() return functions for this purpose.

Examples

```
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 *Row heights*

Description

Functions to get or set the *row heights* property of huxtable rows.

Usage

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

Arguments

ht	A huxtable.
value	A vector.
row	A row specifier. See rowspecs for details.

Details

If character, value must contain valid CSS or LaTeX lengths. If numeric, in HTML, values are scaled to 1 and treated as proportions of the table height. In LaTeX, they are treated as proportions of the text height (\textheight).

Value

For row_height, the row_height property. For set_row_height, the modified huxtable.

See Also

Other row/column heights: col_width

Examples

```
row_height(jams) <-  c(.2, .1, .1, .1)
row_height(jams)
```

`rtf_fc_tables` *Create RTF font and color tables*

Description

Create RTF font and color tables

Usage

```
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.

Examples

```
# 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)
```

`sanitize` *Escape text for various formats*

Description

This escapes a string for LaTeX, HTML or RTF.

Usage

```
sanitize(str, type = c("latex", "html", "rtf"))
```

Arguments

`str`	A character object.
`type`	`"latex"`, `"html"` or `"rtf"`.

Details

HTML and LaTeX code was copied over from `xtable::sanitize()`.

Value

The sanitized character object.

Examples

```
txt <- "Make $$$ with us"
sanitize(txt, type = "latex")
```

`set-multiple` *Set left, right, top and bottom properties*

Description

These are convenience functions which set left, right, top and bottom properties simultaneously for the specified cells.

Usage

```
set_all_borders(ht, row, col, value, byrow = FALSE)

map_all_borders(ht, row, col, fn)

set_all_border_colors(ht, row, col, value, byrow = FALSE)

map_all_border_colors(ht, row, col, fn)

set_all_border_styles(ht, row, col, value, byrow = FALSE)

map_all_border_styles(ht, row, col, fn)

set_all_padding(ht, row, col, value, byrow = FALSE)

map_all_padding(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.
byrow	Deprecated. Use by_cols() instead.
fn	A mapping function. See mapping-functions for details.

Value

The modified huxtable.

Examples

```
ht <- huxtable(a = 1:3, b = 1:3)
set_all_borders(ht, 1:3, 1:2, 1)
ht <- set_all_border_colors(ht, "red")
ht <- set_all_border_styles(ht, "double")
ht <- set_all_padding(ht, 1:3, 1:2, "20px")
```

---

set_cell_properties *Set multiple cell properties*

Description

Set multiple cell properties

Usage

```
set_cell_properties(ht, row, col, ...)
```

Arguments

ht	A huxtable.
row	A row specification.
col	A column specification.
...	Named list of cell properties.

Value

The modified huxtable object.

Examples

```
ht <- hux(a = 1:3, b = 1:3)
ht <- set_cell_properties(ht, 1, 1,
      italic = TRUE, text_color = "red")
text_color(ht)
ht
```

---

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, byrow = FALSE)
map_contents(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	Cell contents.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use `by_cols()` instead.

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.

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

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

`set_outer_borders` *Set borders around a rectangle of cells*

Description

Set borders around a rectangle of cells

Usage

```
set_outer_borders(ht, row, col, value)

set_outer_border_colors(ht, row, col, value)

set_outer_border_styles(ht, row, col, value)
```

Arguments

`ht`	A huxtable.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`value`	Border width in points, border color, or border style (see `left_border_style()`).

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.

Examples

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

# Problems with colspan:
rowspan(ht2)[2, 1] <- 2
set_outer_borders(ht2, 1:2, 1:2, 1)
```

t.huxtable *Transpose a huxtable*

Description

Transpose a huxtable

Usage

```
## 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 object.

Examples

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

tabular_environment *Tabular environment*

Description

Functions to get or set the table-level *tabular environment* property of a huxtable.

Usage

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

Arguments

ht	A huxtable.
value	A length-one character vector. Set to NA to reset to the default, which is "tabularx".

Details

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

Value

For tabular_environment, the tabular_environment property. For set_tabular_environment, the modified huxtable.

Examples

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

---

text_color *Text color*

Description

Functions to get or set the *text color* property of huxtable cells.

Usage

```
text_color(ht)
text_color(ht) <- value
set_text_color(ht, row, col, value, byrow = FALSE)
map_text_color(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A character vector or matrix of valid R colors. Set to NA to reset to the default, which is NA.
row	A row specifier. See rowspecs for details.
col	An optional column specifier.

fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

Colors can be in any format understood by R, e.g. "red", "#FF0000" or rgb(1,0,0).

Value

For text_color, the text_color property. For set_text_color and map_text_color, 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, "blue")
set_text_color(jams,
      2:3, 1, "blue")
map_text_color(jams,
      by_rows("blue", "red"))
```

---

themes *Theme a huxtable*

Description

These functions quickly set default styles for a huxtable.

Usage

```
theme_plain(ht, position = "left")

theme_basic(ht, header_row = TRUE, header_col = TRUE)

theme_striped(ht, stripe = grDevices::grey(0.9), header_row = TRUE,
  header_col = TRUE)

theme_grey(ht, header_row = TRUE, header_col = TRUE)

theme_blue(ht, header_row = TRUE, header_col = TRUE)
```

```
theme_orange(ht, header_row = TRUE, header_col = TRUE)

theme_green(ht, header_row = TRUE, header_col = TRUE)

theme_article(ht, header_row = TRUE, header_col = TRUE)

theme_mondrian(ht, prop_colored = 0.1, font = "Arial")
```

Arguments

ht	A huxtable object.
position	"left", "center" or "right"
header_row	Logical: style first row differently?
header_col	Logical: style first column differently?
stripe	Background colour for alternate 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 just adds a border for header rows and/or columns.

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_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 black borders and randomized colors.

Value

The huxtable object, appropriately styled.

Examples

```
theme_striped(jams)
theme_plain(jams)
theme_basic(jams)
theme_article(jams)
theme_mondrian(jams)
theme_grey(jams)
theme_blue(jams)
theme_orange(jams)
theme_green(jams)
```

```
## Not run:
  quick_pdf(
          theme_striped(jams),
          theme_plain(jams),
          theme_basic(jams),
          theme_article(jams),
          theme_mondrian(jams),
          theme_grey(jams),
          theme_blue(jams),
          theme_orange(jams),
          theme_green(jams)
        )

## End(Not run)
```

`tidy_override` *Override a model's* `tidy` *output*

Description

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

Usage

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

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

## S3 method for class '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 statistics?
`object`	A `tidy_override` object.

Value

An object of class "tidy_override". When `tidy` and `glance` are called on this, it will return results from the underlying model, replacing some columns with your own data.

Examples

```
if (! requireNamespace("broom")) {
  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))

broom::tidy(fixed_lm1)

cbind(huxreg(fixed_lm1), huxreg(lm1))
```

---

valign *Vertical alignment*

Description

Functions to get or set the *vertical alignment* property of huxtable cells.

Usage

```
valign(ht)
valign(ht) <- value
set_valign(ht, row, col, value, byrow = FALSE)
map_valign(ht, row, col, fn)
```

Arguments

ht	A huxtable.
value	A character vector or matrix which may be "top", "middle", "bottom" or NA. Set to NA to reset to the default, which is "top".
row	A row specifier. See rowspecs for details.
col	An optional column specifier.
fn	A mapping function. See mapping-functions for details.
byrow	Deprecated. Use by_cols() instead.

Details

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

Value

For valign, the valign property. For set_valign and map_valign, the modified huxtable.

Examples

```
valign(jams) <-  "bottom"
valign(jams)


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

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

jams4 <- map_valign(jams,
      by_rows(
        "bottom",
        "bottom")
      )
valign(jams4)
```

`width` *Table width*

Description

Functions to get or set the table-level *table width* property of a huxtable.

Usage

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

Arguments

`ht`	A huxtable.
`value`	A length-one vector. If numeric, `value` is treated as a proportion of the surrounding block width (HTML) or text width (LaTeX). If character, it must be a valid CSS or LaTeX width. Set to `NA` to reset to the default, which is `0.5`.

Value

For `width`, the `width` property. For `set_width`, the modified huxtable.

See Also

Other table measurements: `height`

Examples

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

`wrap` *Text wrapping*

Description

Functions to get or set the *text wrapping* property of huxtable cells.

Usage

```
wrap(ht)
wrap(ht) <- value
set_wrap(ht, row, col, value, byrow = FALSE)
map_wrap(ht, row, col, fn)
```

Arguments

`ht`	A huxtable.
`value`	A logical vector or matrix. If `TRUE`, long cell contents will be wrapped into multiple lines. Set to `NA` to reset to the default, which is `FALSE`.
`row`	A row specifier. See rowspecs for details.
`col`	An optional column specifier.
`fn`	A mapping function. See mapping-functions for details.
`byrow`	Deprecated. Use `by_cols()` instead.

Value

For `wrap`, the `wrap` property. For `set_wrap` and `map_wrap`, the modified huxtable.

Examples

```
ht <- huxtable(paste(
      rep("Some long text.", 20),
      collapse = " "))
width(ht) <- 0.2
wrap(ht) <- TRUE
## Not run:
  quick_html(ht)

## End(Not run)
```

```
jams2 <- set_wrap(jams,
      TRUE)
wrap(jams2)

jams3 <- set_wrap(jams,
      2:3, 1, TRUE)
wrap(jams3)

jams4 <- map_wrap(jams,
      by_rows(
        TRUE,
        FALSE)
      )
wrap(jams4)
```

`[.huxtable` *Subset a huxtable*

Description

Subset a huxtable

Usage

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

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

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

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

Arguments

`x`	A huxtable.
`i`	Rows to select.
`j, name`	Columns to select.
`drop`	Not used.
`value`	A matrix, data frame, huxtable or similar object.

Details

`[` always returns a huxtable, while `$` and `[[` return the underlying data.

For the replacement function `[<-`, if `value` is a huxtable, then its cell properties will be copied into `x`. In addition, if `value` fills up an entire column, then column properties will be copied into the replaced columns of `x`, and if it fills up an entire row, then row properties will be copied into the replaced rows of `x`.

Replacement functions `$<-` and `[[<-` replace existing data without affecting any properties.

Value

A huxtable.

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