Introduction to Huxtable

David Hugh-Jones

2017-03-06

```
## Warning: package 'knitr' was built under R version 3.3.2
```

Huxtable is a package for writing LaTeX and HTML tables. It is powerful, but easy to use. It is meant to be a replacement for packages like xtable, which is useful but not always very user-friendly.

To create a table with huxtable, use the function `huxtable`, or `hux` for short.

```
library(huxtable)
ht <- hux(Employee = c('John Smith', 'Jane Doe', 'David Hugh-Jones'), Salary = c(50000, 50000, 40000),
      add_colnames = TRUE)
```

Or, if you already have your data in a data frame, use `as_hux`.

```
data(mtcars)
car_ht <- as_hux(mtcars)
```

Huxtables are simply data frames, along with some extra information on how to display them. If you look at them in R, they'll appear just like ordinary data frames. Notice that we've added the column names to the data frame itself. We're going to print them out, so it makes sense that they need to be part of the actual table.

```
print(ht)
```

```
##           Employee Salary
## 1         Employee Salary
## 2       John Smith  50000
## 3         Jane Doe  50000
## 4 David Hugh-Jones  40000
```

To print them out using LaTeX or HTML, just call `print_latex` or `print_html`. In knitr documents, like this one, you can simply evaluate the hux. It will know what format to print itself in.

```
ht
```

Employee	Salary
John Smith	50000.00
Jane Doe	50000.00
David Hugh-Jones	40000.00

The default output is a very plain table. To customize it, you can set various properties. Let's make our table headings bold, draw a line under the header row, and align the second row to the right:

```
bold(ht)[1,]           <- TRUE
bottom_border(ht)[1,]  <- TRUE
align(ht)[,2]        <- 'right'
right_padding(ht)      <- 10
left_padding(ht)       <- 10

ht
```

You set properties by assigning to the property name, just as you assign `names(x) <- new_names` in base R.

Employee	Salary
John Smith	50000.00
Jane Doe	50000.00
David Hugh-Jones	40000.00

Some properties, like `bold` and `bottom_border`, are cell-level. You can set them for individual cells in your data. For example, the line `bold(ht)[1,] <- TRUE` in the code above sets the `bold` property for the first row of the huxtable. And `align(ht)[,2] <- 'right'` sets the alignment for the second column.

In fact, `right_padding` and `left_padding` are also cell-level properties. But we set them for all cells at once. You can do that for any property - just do `property(ht) <- value`.

By contrast, `caption` is a table-level property. It only takes one value, which sets a table caption.

```
caption(ht) <- 'Employee table'

ht
```

Table 1: Employee table

Employee	Salary
John Smith	50000.00
Jane Doe	50000.00
David Hugh-Jones	40000.00

See the help files for a list of all properties you can set. Most properties work the same for LaTeX and HTML, though there are some exceptions.

Pipe style syntax

If you prefer to use the `magrittr` pipe operator (`%>%`), then you can do that too using `set_property` functions:

```
# after loading magrittr or dplyr:
ht %>%
      set_bold(1, 1:2, TRUE)           %>%
      set_bottom_border(1, 1:2, 1)     %>%
      set_align(-1, 2, 'right')        %>%
      set_right_padding(1:4, 1:2, 10) %>%
      set_left_padding(1:4, 1:2, 10)
```

Table 2: Employee table

Employee	Salary
John Smith	50000.00
Jane Doe	50000.00
David Hugh-Jones	40000.00

To see the current properties of a huxtable, just use the properties function without the left arrow:

```
italic(ht)
```

```
##   Employee Salary
## 1    FALSE  FALSE
## 2    FALSE  FALSE
## 3    FALSE  FALSE
## 4    FALSE  FALSE
```

```
position(ht)
```

```
## [1] "center"
```

```
bottom_border(ht)[1:2,] # first two rows
```

```
##   Employee Salary
## 1        1      1
## 2        0      0
```

Subsetting a huxtable

You can subset, sort and generally data-wrangle a huxtable just like a normal data frame.

```
cars_mpg <- car_ht[, c('mpg', 'cyl', 'am')]
cars_mpg <- cars_mpg[order(cars_mpg$cyl),]

cars_mpg <- cars_mpg %>%
      add_rownames(colname = 'Car Name') %>%
      add_colnames()

cars_mpg[1:5,]
```

Car Name	mpg	cyl	am
Datsun 710	22.80	4.00	1.00
Merc 240D	24.40	4.00	0.00
Merc 230	22.80	4.00	0.00
Fiat 128	32.40	4.00	1.00

However, in general it is a good idea to prepare your data first, before styling it. For example, it was easier to sort the `cars_mpg` data by cylinder before adding column names to the data frame itself.

Column and row spans

As well as changing styling, you can let cells span multiple rows or columns.

```
cars_mpg <- cbind(car_type = rep("", nrow(cars_mpg)), cars_mpg)
cars_mpg$car_type[1] <- 'Four cylinders'
cars_mpg$car_type[13] <- 'Six cylinders'
cars_mpg$car_type[20] <- 'Eight cylinders'
rowspan(cars_mpg)[1, 1] <- 12
rowspan(cars_mpg)[13, 1] <- 7
rowspan(cars_mpg)[20, 1] <- 14

cars_mpg <- rbind(c('', 'List of cars', '', '', ''), cars_mpg)
colspan(cars_mpg)[1, 2] <- 4
align(cars_mpg)[1, 2] <- 'center'

top_border(cars_mpg)[c(2, 14, 21),] <- 1
right_border(cars_mpg)[,1] <- 1

cars_mpg
```

	List of cars			
	Car Name	mpg	cyl	am
Four cylinders	Datsun 710	22.80	4.00	1.00
	Merc 240D	24.40	4.00	0.00
	Merc 230	22.80	4.00	0.00
	Fiat 128	32.40	4.00	1.00
	Honda Civic	30.40	4.00	1.00
	Toyota Corolla	33.90	4.00	1.00
	Toyota Corona	21.50	4.00	0.00
	Fiat X1-9	27.30	4.00	1.00
	Porsche 914-2	26.00	4.00	1.00
	Lotus Europa	30.40	4.00	1.00
	Volvo 142E	21.40	4.00	1.00
Six cylinders	Mazda RX4	21.00	6.00	1.00
	Mazda RX4 Wag	21.00	6.00	1.00
	Hornet 4 Drive	21.40	6.00	0.00
	Valiant	18.10	6.00	0.00
	Merc 280	19.20	6.00	0.00
	Merc 280C	17.80	6.00	0.00
	Ferrari Dino	19.70	6.00	1.00
Eight cylinders	Hornet Sportabout	18.70	8.00	0.00
	Duster 360	14.30	8.00	0.00
	Merc 450SE	16.40	8.00	0.00
	Merc 450SL	17.30	8.00	0.00
	Merc 450SLC	15.20	8.00	0.00
	Cadillac Fleetwood	10.40	8.00	0.00
	Lincoln Continental	10.40	8.00	0.00
	Chrysler Imperial	14.70	8.00	0.00
	Dodge Challenger	15.50	8.00	0.00
	AMC Javelin	15.20	8.00	0.00
	Camaro Z28	13.30	8.00	0.00
	Pontiac Firebird	19.20	8.00	0.00
	Ford Pantera L	15.80	8.00	1.00
	Maserati Bora	15.00	8.00	1.00

Number formatting

You can change how huxtable formats numbers using `number_format`. Huxtable guesses whether your cell is a number based on its contents, not on the column type. Set `number_format` to a number of decimal places (for more advanced options, see the help files).

```
number_format(cars_mpg) <- 0
cars_mpg[1:7,]
```

Quick themes

Huxtable comes with predefined themes that change various parts of formatting:

	List of cars			
Four cylinders	Car Name	mpg	cyl	am
	Datsun 710	23	4	1
	Merc 240D	24	4	0
	Merc 230	23	4	0
	Fiat 128	32	4	1
	Honda Civic	30	4	1
Six cylinders	Mazda RX4	21	6	1
	Mazda RX4 Wag	21	6	1
	Hornet 4 Drive	21	6	0
	Valiant	18	6	0
	Merc 280	19	6	0
	Merc 280C	18	6	0
	Ferrari Dino	20	6	1

```
theme_striped(cars_mpg[14:20,], stripe = 'bisque1', header_col = FALSE, header_row = FALSE)
```

Printing on screen

Lastly, you can print a huxtable on screen using `print_screen`. Borders, column and row spans and cell alignment are shown:

```
print_screen(ht)
```

```
## Employee table
##
##    Employee              Salary
##  ------------------------------
##    John Smith          50000.00
##
##    Jane Doe            50000.00
##
##    David Hugh-Jones    40000.00
##
```