LaTeX Output, Table Borders, and `hhline`

Benjamin Nutter

2016-04-08

Contents

When rendering to LaTeXoutput, `pixiedust` offers two styles of table borders. Borders may be drawn using either the facilities of the `xcolor` and `arydshlin` packages, or using the `hhline` packages (Note: these are LaTeXpackages). By default, `pixiedust` uses `xcolor` and `arydshln`.

Comparison of Styles

Feature	`xcolor`	`hhline`
Dashed borders	Yes	No
Colored borders	Yes	No
Double borders	No	Yes
Works with background colors	No	Yes

Before getting started, please note that the YAML front matter for this documents is as follows:

```
---
title: "LaTeX Output, Table Borders, and `hhline`"
author: "Benjamin Nutter"
date: "2016-04-08"
output: pdf_document
header-includes:
- \usepackage{amssymb}
- \usepackage{arydshln}
- \usepackage{graphicx}
- \usepackage{hhline}
- \usepackage{longtable}
- \usepackage{multirow}
- \usepackage[dvipsnames,table]{xcolor}
- \makeatletter
- \newcommand*\vdashline{\rotatebox[origin=c]{90}{\$\dabar@\dabar@\dabar@\$}}
- \makeatother
---
```

For these illustrations, we will use our usual example linear model. We will also apply `medley_model` to the table before applying any other customizations.

```
library(pixiedust)
options(pixiedust_print_method = "latex")

# source("https://gist.githubusercontent.com/nutterb/8a5e39544df395f6f4198e520580a80f/raw/f4f416ecc9cee

medley_all_borders <- function(x, rows=NULL, cols=NULL,
                               horizontal = TRUE, vertical = TRUE,
                               part = "body")
{
  part <- part <-
    match.arg(part,
              c("table", "head", "body", "interfoot", "foot"),
              several.ok = TRUE)
  if ("table" %in% part)
  {
    part <- c("head", "body", "interfoot", "foot")
  }

  for (p in part)
  {
    if (!is.null(x[[p]]))
    {
      part_rows <- if (is.null(rows)) max(x[[p]][["row"]]) else rows
      part_cols <- if (is.null(cols)) max(x[[p]][["col"]]) else cols

      x <- sprinkle(x,
                    rows = 1:part_rows,
                    cols = 1:part_cols,
                    border = c(if (vertical) "left" else NULL,
                               if (horizontal) "bottom" else NULL),
                    part = p)
      if (horizontal)
      {
        x <- sprinkle(x,
                      rows = 1,
                      border = "top",
                      part = p)
      }
      if (vertical)
      {
        x <- sprinkle(x,
                      cols = part_cols,
                      border = "right",
                      part = p)
      }
    }
  }
  x
}

fit <- lm(mpg ~ wt + qsec + factor(am),
```

Table 2: Table borders using xcolor

term	estimate	std.error	statistic	p.value
(Intercept)	9.62	6.96	1.38	0.18
wt	-3.92	0.71	-5.51	< 0.001
qsec	1.23	0.29	4.25	< 0.001
factor(am)1	2.94	1.41	2.08	0.047

Table 3: Table borders using hhline

term	estimate	std.error	statistic	p.value
(Intercept)	9.62	6.96	1.38	0.18
wt	-3.92	0.71	-5.51	< 0.001
qsec	1.23	0.29	4.25	< 0.001
factor(am)1	2.94	1.41	2.08	0.047

```
          data = mtcars)
```

A Common Shortcoming

In both styles, vertical borders can become thicker than expected when adjoining cells have the adjoining borders defined. For example, if column 2 has a right border and column 3 has a left border, the adjoining borders appear as one thick border.

```
dust(fit,
     caption = "Table borders using xcolor",
     hhline = FALSE) %>%
  medley_model() %>%
  sprinkle(border = "all")
```

```
dust(fit,
     caption = "Table borders using hhline",
     hhline = TRUE) %>%
  medley_model() %>%
  sprinkle(border = "all")
```

This can be avoided by only assigning one vertical border.

```
dust(fit,
     caption = "Table borders using xcolor",
     hhline = FALSE) %>%
  medley_model() %>%
  #* Left border on all columns
  sprinkle(cols = 1:5,
           border = c("left", "top", "bottom")) %>%
  #* Right border on last column
  sprinkle(cols = 5,
           border = "right")
```

`pixiedust` provides a medley (`medley_all_borders`) that performs this task for you.

Table 4: Table borders using xcolor

term	estimate	std.error	statistic	p.value
(Intercept)	9.62	6.96	1.38	0.18
wt	-3.92	0.71	-5.51	< 0.001
qsec	1.23	0.29	4.25	< 0.001
factor(am)1	2.94	1.41	2.08	0.047

Table 5: Table borders using the all borders medley

term	estimate	std.error	statistic	p.value
(Intercept)	9.62	6.96	1.38	0.18
wt	-3.92	0.71	-5.51	< 0.001
qsec	1.23	0.29	4.25	< 0.001
factor(am)1	2.94	1.41	2.08	0.047

```
dust(fit,
     caption = "Table borders using the all borders medley",
     hhline = FALSE) %>%
  medley_model() %>%
  medley_all_borders()
```

Dashed Borders

Colored Borders

Double Borders

Background Colors