Creating heatmap visualizations of data in R

Emahn Novid

11/28/2017

In this post we cover rendering heatmap visualizations in R. Heatmap plots allow us to visualize numerical data using colors. There are several types of heatmaps, and they all replace numbers with color scales - the shade of the color typically represents how high the number is. For example, meteorologists often use heatmaps to show weather patterns - bright colors like red are used to indicate the more severe areas of weather, for instance, dangerous areas near the center of a hurricane.

Heatmaps are especially useful because they provide a quicker way to see patterns im data. Using colors in place of numbers makes it easy to spot high and low values. To demonstrate this point imagine looking at a grid of numbers and being asked to find the highest value - you would have to use your eyes to linearly search each line of the grid reading every number in the grid until you found ome that's higher than all the others. Compare this slow linear search to the act of spotting colors - if instead the highest number was represented by the brightest shade pf red then your eyes would immediately be drawn to its location. Our minds perceptual system is quicker at spotting differences in color than it is at reading numbers.

We're going to walk through three different ways to make heatmaps in R. First, using the built in heatmap function, then with two different packages - gplot annd d3heatmap.

Using the builtin heatmap function in R

`heatmap()` is a base R fucntion, part of the stats package.

The `heatmap()` function produces simple looking heatmaps and using it is straightforward.

For this example, we use the builtin dataset **mtcars**.

`heatmap()` Example

1. Preparing the data

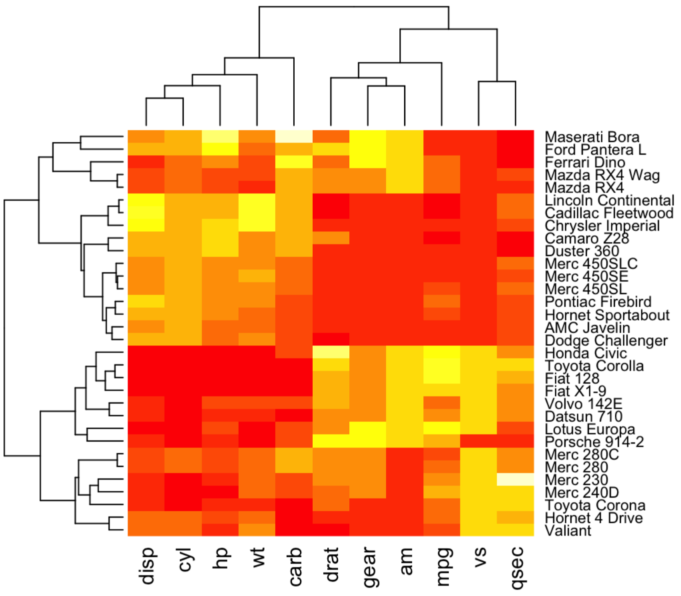
First, we rescale the variables in the **mtcars** dataset so that we can meaningfully compare them.

```
dat <- scale(mtcars)
```

2. Drawing the heatmap

We specify two parameters: 1. **dat** - the data frame holding our data. 2. **scale** - Either "row" (default), "column", or "none" specifying whether the values should be centered and scaled in a certain direction - by row, column or neither.

```
heatmap(dat, scale = "none")
```



In the resulting visualization, we see that the intersection of car model **rows** and variable **columns** is a tile colored with shades of yellow to red corresponding to lower and higher values.

Yellow = Low value Red = High Value

Using this plot we can quickly get a general idea of the differences between the various car models. For example, we can easily see the cars with the best fuel efficiency as the ones with red tiles in the **mpg** column.

Using heatmap.2() to create heatmaps in R

gplot is a package that provides us with a more advanced heatmap function: `heatmap.2()`.

An adavantage of `heatmap.2()` is it allows us to color the tiles with a colored gradient which gives us a more fine grained look at the data, along with a key telling the reader how to interpret the colors.

`heatmap.2()` Example

1. Install the `gplot` package

```
# install.packages("gplots")
```

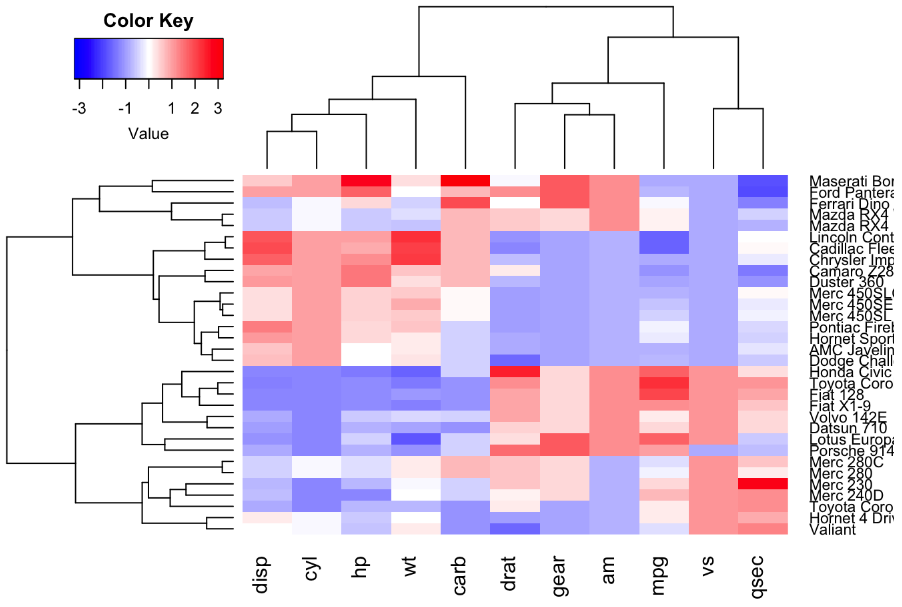
2. Generate the heatmap

```
library("gplots")
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
heatmap.2(dat, scale = "none", col = bluered(100),
          trace = "none", density.info = "none")
```



Here we use a gradient to represent the difference between values using the parameter `col` with a value of `bluered(100)`.

`bluered(n)` is one of several functions used to generate the gradients. The other options are `redgreen(n)`, `greenred(n)`, `bluered(n)` and `redblue(n)`.

The last two parameters `trace` and `density.info` are set to “none” to exclude some automatically generated overlays that are beyond the scope of this post. However, those are examples of how `heatmap.2()` has more capabilities than `heatmap()`.

Using d3heatmap() to create heatmaps in R

`d3heatmap()` is part of the package `d3heatmap`. the function produces an interactive heatmap that respons to user input. We can click and zoom in on the heatmap in order to inspect the data more closely.

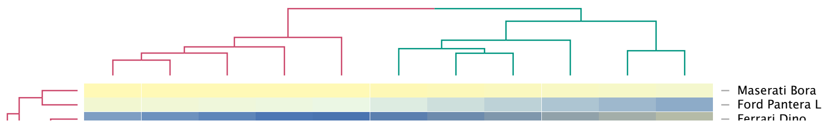
`d3heatmap()` Example

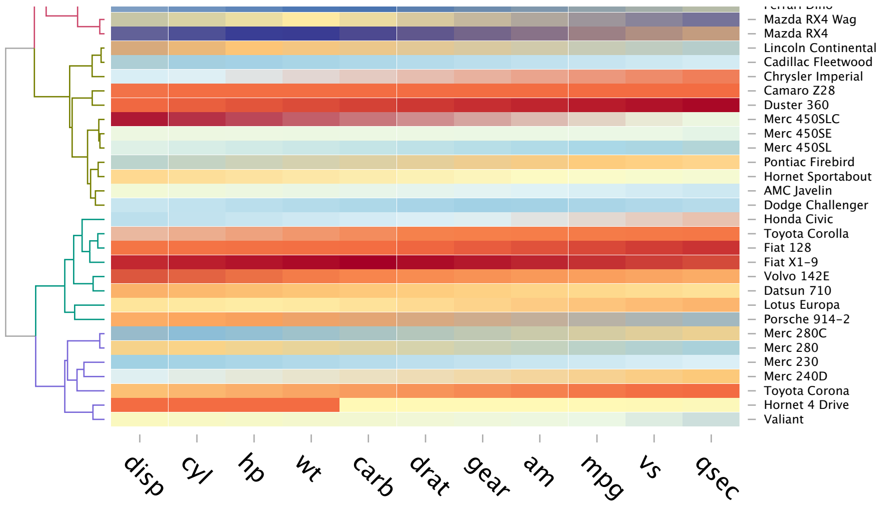
1. Install the `d3heatmap` package

```
# install.packages("d3heatmap")
```

2. Generate the heatmap

```
library("d3heatmap")
d3heatmap(dat,
          colors = "RdYlBu",
          k_row = 4,
          k_col = 2)
```





There are a few new parameters here: `colors`, `k_row`, and, `k_col`.

`colors` = Colors to use for cells

`k_row` = The number of groups in rows

`k_col` = The number of groups in columns.

You can interact with this heatmap by hovering over cells to see the row name, column name, and value.

You can also click and drag over an area of the heatmap to zoom in. Clicking again on the heatmap zooms you out.

This type of interactivity is useful when viewing a large set of data because otherwise it can be tiring to trace the intersection of rows and columnsusing only your eyes. It also allows you to see the real value corresponding to the color without sacrificing the utility of the color representations.

Conclusion

We covered three functions of increasing complexity to generate heatmaps in R. The advanctage of using packages over the base heatmap function comes down to better visualizations, additional configurations parameters, and interactivity. Heatmaps allow us to efficiently surveying our data, so learning how to make them look and behave the way we want enables us to get the most out of our visualizations.

References

1. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/heatmap.html
2. http://sebastianraschka.com/Articles/heatmaps_in_r.html
3. http://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/
4. http://www.sthda.com/english/articles/28-hierarchical-clustering-essentials/93-heatmap-static-and-interactive-absolute-guide/
5. https://cran.r-project.org/web/packages/d3heatmap/d3heatmap.pdf
6. http://www.datavis.ca/papers/HeatmapHistory-tas.2009.pdf
7. http://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/