Post02: Advanced Visualization

Xiaoya Li

November 26, 2017

Introduction

Data visulization is one of the most powerful usage of R. So far, we've seen many type of data visualization, for example, Histogram, Barchart, Boxplot, Scatter plot, etc. These are common and efficient, but other than that, there are still many choices of interesting data visualization. In today's post, I'm going to introduce three of them: Heat Map, Mosaic Plot and 3D Graph.

Advanced Visualization

Before we jump into the topic, let's load all packages we are going to use in this post.

```r
#datasets contains most of the database we are going to use
library(datasets)
library(boot)

#graphic
library(ggplot2)
library(ggmosaic)
library(plotly)
```

I. Heat Map

Heat map represents relationship between two or more variable in a two demensional graph, using intensity of color in a tabular format.

Example 1

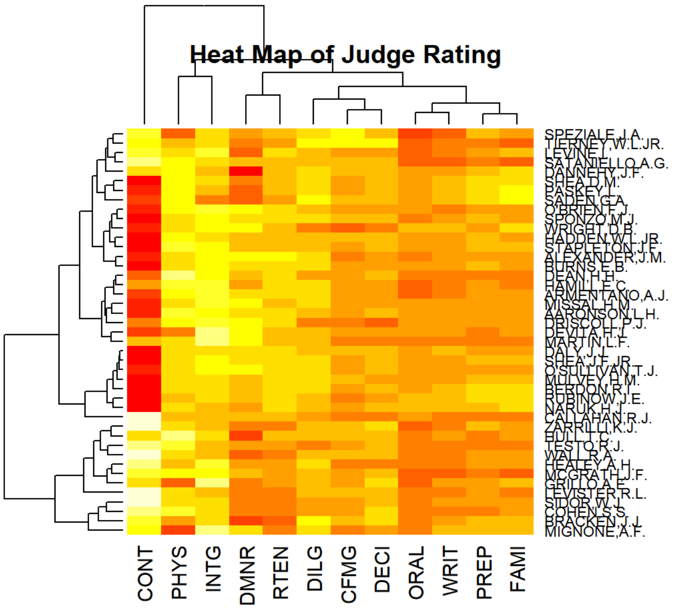
For the first example, let's see how to use the `heatmap()` function in R to generate a heat map.
The `USJudgeRatings` dataset contains numeric rating of selected judges from different persepective. Let's take a look.

```r
head(USJudgeRatings)
```

```
##                CONT INTG DMNR DILG CFMG DECI PREP FAMI ORAL WRIT PHYS RTEN
## AARONSON,L.H.   5.7  7.9  7.7  7.3  7.1  7.4  7.1  7.1  7.1  7.0  8.3  7.8
## ALEXANDER,J.M.  6.8  8.9  8.8  8.5  7.8  8.1  8.0  8.0  7.8  7.9  8.5  8.7
## ARMENTANO,A.J.  7.2  8.1  7.8  7.8  7.5  7.6  7.5  7.5  7.3  7.4  7.9  7.8
## BERDON,R.I.     6.8  8.8  8.5  8.8  8.3  8.5  8.7  8.7  8.4  8.5  8.8  8.7
## BRACKEN,J.J.    7.3  6.4  4.3  6.5  6.0  6.2  5.7  5.7  5.1  5.3  5.5  4.8
## BURNS,E.B.      6.2  8.8  8.7  8.5  7.9  8.0  8.1  8.0  8.0  8.0  8.6  8.6
```

`heatmap()` function requires our input to be a matrix.

```r
#convert it to a matrix
heatmap(as.matrix(USJudgeRatings))
#add a title
title(main = "Heat Map of Judge Rating")
```



Description: This graph represents the rating of each judge. For each column, the dark portion indicates the best judge in this attribute; for each row, the dark portion indicates the best rating of each judge got.

===

Example 2

Alternatively, we can also use `ggplot` to plot the heat map. `airquality` dataset contains air quality measurement in New York City.

```
head(airquality)
```

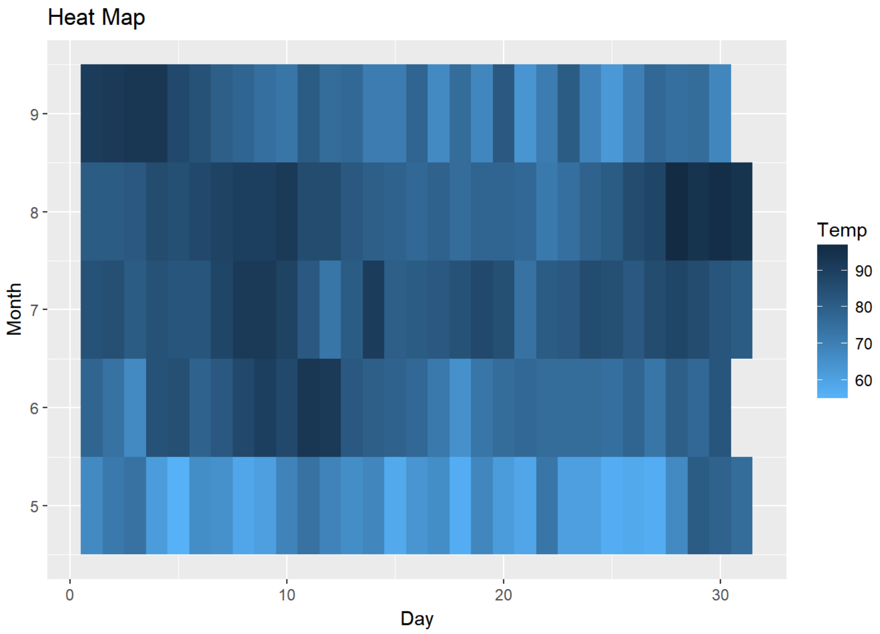
```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

This time, other than plotting all the attribute as the previous graph, we want to compare the temperature on each day in each month. Here's the simple R code using `ggplot()` and `geom_tile()`.

```
ggplot(airquality, aes(Day, Month)) +
  #build the heat map
  geom_tile(aes(fill = Temp)) +

  #reverse legend color such that dark color related to higher temperature
  scale_fill_gradient(high = "#132B43", low = "#56B1F7") +

  #add a title
  ggtitle(label = "Heat Map")
```



Description: This graph represents the temperature of each day in each month. Horizontally, the dark portion indicates the hottest day in this month; vertically, the dark portion indicates the hottest month.

===

Mosaic Plot

Mosaic plot is useful to represents categorical data.

Example 3

`mosaicplot()` function in R provides a simple way to generate a mosaic plot.
`HairEyeColor` dataset contains the Hair and Eye Color of Statistics Students. Let's take a look at the dataset.

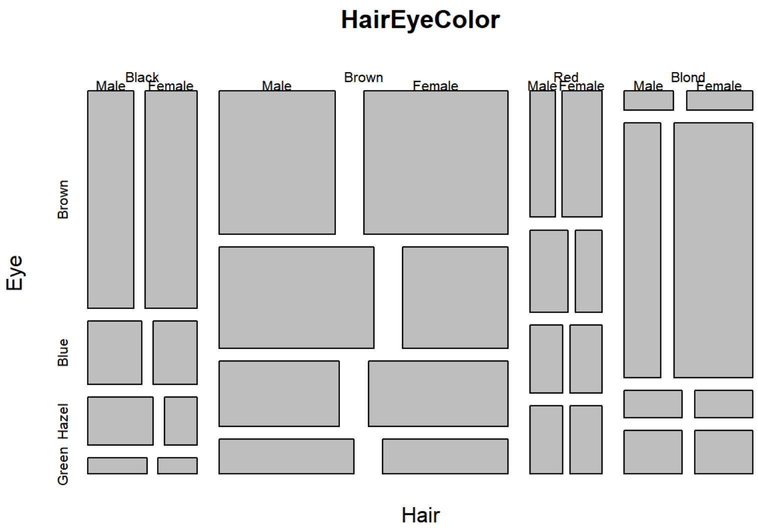
```
HairEyeColor
```

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

`HairEyeColor` is in the class of `table`, which is different from what we used to have in data.frame.

Here's the one line simple R code.

```
mosaicplot(HairEyeColor)
```



Description: This graph represents the proportion of students of different hair and eye colors in the size of the area.

===

Example 4

Still, we can use `ggplot()` to graph a detailed mosaic plot. Inorder to graph in ggplot, we need to add another package `ggmosaic`.

`Titanic` is a bit more complicated dataset, it contains information of Survival of passengers on the Titanic. Also, the `Titanic` in `dastasets` is in the class of table, but since `ggplot()` works with dataframe instead of table, we need the `Titanic` data in its original form.

```
#download .csv file online
download.file(url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/csv/datasets/Titanic.cs
v", destfile = "../Data/titanic_raw.csv")

#read in data
titanic_raw <- read.csv(file = "../Data/titanic_raw.csv")
head(titanic_raw)
```

```
##   X                                          Name PClass   Age    Sex
## 1 1                  Allen, Miss Elisabeth Walton    1st 29.00 female
## 2 2                   Allison, Miss Helen Loraine    1st  2.00 female
## 3 3           Allison, Mr Hudson Joshua Creighton    1st 30.00   male
## 4 4 Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st 25.00 female
## 5 5                 Allison, Master Hudson Trevor    1st  0.92   male
## 6 6                            Anderson, Mr Harry    1st 47.00   male
##   Survived SexCode
## 1        1       1
## 2        0       1
## 3        0       0
## 4        0       1
## 5        1       0
## 6        1       0
```

We need some data preparation before we draw the graph.
We add a column `AgeDecade` that categorize each passengers' age into decade.

```r
#Initialize a empty character vector
AgeDecade <- rep(" ", nrow(titanic_raw))

#Use a for loop to categorize each passengers' age
for(i in 1:nrow(titanic_raw)){
  x = titanic_raw$Age[i]
  if(is.na(x)){AgeDecade[i] = "unknown"}
  else if(x >= 0 & x < 10){AgeDecade[i] = "0 - 9"}
  else if(x >= 10 & x < 20){AgeDecade[i] = "10 - 19"}
  else if(x >= 20 & x < 30){AgeDecade[i] = "20 - 29"}
  else if(x >= 30 & x < 40){AgeDecade[i] = "30 - 39"}
  else if(x >= 40 & x < 50){AgeDecade[i] = "40 - 49"}
  else if(x >= 50 & x < 60){AgeDecade[i] = "50 - 59"}
  else if(x >= 60 & x < 70){AgeDecade[i] = "60 - 69"}
  else{AgeDecade[i] = "70 +"}
}

#Add that vector into a column of the dataframe
titanic_raw$AgeDecade <- as.factor(AgeDecade)
head(titanic_raw)
```

```
##   X                                          Name PClass   Age    Sex
## 1 1                  Allen, Miss Elisabeth Walton    1st 29.00 female
## 2 2                   Allison, Miss Helen Loraine    1st  2.00 female
## 3 3           Allison, Mr Hudson Joshua Creighton    1st 30.00   male
## 4 4 Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st 25.00 female
## 5 5                 Allison, Master Hudson Trevor    1st  0.92   male
## 6 6                            Anderson, Mr Harry    1st 47.00   male
##   Survived SexCode AgeDecade
## 1        1       1   20 - 29
## 2        0       1     0 - 9
## 3        0       0   30 - 39
## 4        0       1   20 - 29
## 5        1       0     0 - 9
## 6        1       0   40 - 49
```

Now, we are ready to graph. I want to visualize the proportion of survivals within sex and age interval.

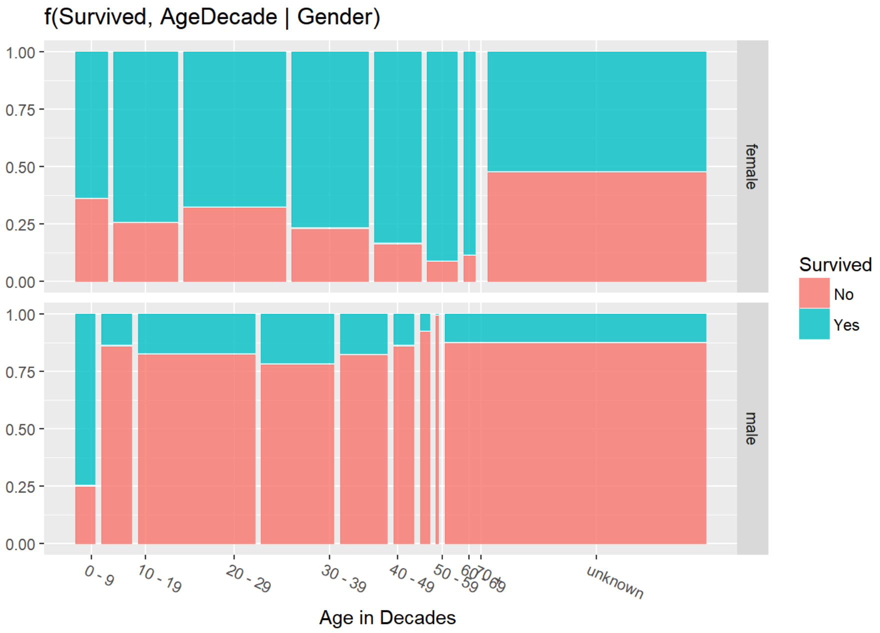
```r
ggplot(titanic_raw) +
  #pick the attribute we are interested in
  geom_mosaic(aes(x = product(Survived,AgeDecade), fill = factor(Survived))) +

  #adjust the x-axis to make it more readable
  theme(axis.text.x = element_text(angle = -25, hjust = .1)) +

  #facet the data into female and male
  facet_grid(Sex ~ .) +

  #add title of the graph
  labs(x="Age in Decades ", title='f(Survived, AgeDecade | Gender)') +

  #add label to legend
  scale_fill_discrete(name="Survived", labels = c("No", "Yes"))
```



Description: This graph visually shows the proportion of survivals within different age intervals and sex. Generally, more female survived than male. And people in age 20-29 survived most in both female and male.

```
=======================================================================================
```

III. 3D Graph

Obviously, the 3D graph is one of the impressive graphs of all, it can represent relation between up tp three variables visually straightforward.

To draw a 3D graph, package `plotly` is a nice tool combined with `ggplot`.

Example 5

For this example, I’m showing you how to graph the most basic 3d scatterplot.

The dataset we used is `cane`, which contains the Sugar-cane Disease Data. Since the column name are not meaningful enough, here’s the documentation from R about this dataset.

> n: The total number of shoots in each plot.
> r: The number of diseased shoots.
> x: The number of pieces of the stems, out of 50, planted in each plot.
> var: A factor indicating the variety of sugar-cane in each plot.
> block: A factor for the blocks.

Let’s take a look of the data.

```
head(cane)
```

```
##       n  r  x var block
## 1  87 76 19   1     A
## 2 119  8 14   2     A
## 3  94 74  9   3     A
## 4  95 11 12   4     A
## 5 134  0 12   5     A
## 6  92  0  3   6     A
```

We’re going to visualize the relation of total number of shoots, number of diseased shoots and number of pieces of the stems. Here’s the R code:

```
plot_ly(cane, x = ~n, y = ~r, z = ~x, color = ~block,
        colors = c('#4AC6B7', '#1972A4', '#965F8A', '#FF7070')) %>%

  #add label to axis
  add_markers() %>%

  #change the label of the axis
  layout(scene = list(
    xaxis = list(title = '# shoots in each plot'),
    yaxis = list(title = '# diseased shoots'),
    zaxis = list(title = '# pieces of the stems')),

    #change the label of the legend
         annotations = list(
           x = 1.13, y = 1.05,
           text = 'Block',
           xref = 'paper',
           yref = 'paper',
           showarrow = FALSE))
```

Note: You can interact with the graph by clicking the legend.
Description: This 3d scatter plot shows relation of total number of shoots, number of diseased shoots and number of pieces of the stems, with different color representing different blocks.

===

Example 6

In this example, I'm showing to how to use 3D surface graph to visualize the probability density estimation.

`cars` in packages `dataset` contains infomation of Speed and Stopping Distances of Cars.

```
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

What if we want to estimate the stopping distance given the speed of the car? **Kernel density estimation** is a way to estimate the probability density function of a random variable. To learn more about **Kernel density estimation**, click here.

There's a function in package `MASS` called `kde2d()`, "Two-Dimensional Kernel Density Estimation", that allows us to carry out the estimation.

You would end up with a list with x being your independent variable and y being the dependent variable that you're trying to estimate based on x.

```
#prepare the data using kernel density estimation

#MASS:: allows us to call functions without actually loading MASS package

#n is the number of grid points in each direction,
#the larger of n, the smoother your graph would looks like
kd <- with(cars, MASS::kde2d(x = speed, y = dist, n = 50))

str(kd)
```

```
## List of 3
##  $ x: num [1:50] 4 4.43 4.86 5.29 5.71 ...
##  $ y: num [1:50] 2 4.41 6.82 9.22 11.63 ...
##  $ z: num [1:50, 1:50] 0.000303 0.000325 0.000345 0.000362 0.000376 ...
```

```
plot_ly(x = kd$x, y = kd$y, z = kd$z) %>%

  #plot the 3d surface
  add_surface() %>%

   #change the label of the axis
  layout(scene = list(
    xaxis = list(title = 'Speed (mph)'),
    yaxis = list(title = 'Stopping Distance (ft)')))
```

Description: This 3D surface estimate the Stopping distance of a car giving its speed. Rotate the object to see more detail.

==

Conclusion

I hope this post can give you more idea about data visulization.

- To visualize data from many perspectives at the same time, consider **Heat Map**.
- To visualize categorical data, try **Mosaic Plot**.
- To visualize multi-variables, try using different kind of **3D Graph**.

Reference

1. http://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/
2. https://www.r-bloggers.com/7-visualizations-you-should-learn-in-r/
3. https://www.analyticsvidhya.com/blog/2015/07/guide-data-visualization-r/
4. https://www.statmethods.net/advgraphs/mosaic.html
5. https://cran.r-project.org/web/packages/ggmosaic/vignettes/ggmosaic.html
6. https://plot.ly/r/3d-scatter-plots/#basic-3d-scatter-plot
7. https://plot.ly/r/3d-surface-plots/