Interactive Data Visualization with Plotly

Brian Yoo

December 2, 2017

Introduction

In order to supplement my learning on data visualization taught during lecture, I delved deeper into different packages and functions that allow for more presentable and flexible graphing of data. That is when I learned about plotly, which is an open source tool that allows users to create and share interactive graphs on the Web. Plotly was developed and founded in Montreal, Quebec. In R, plotly's most basic way of creating interactive graphs works by converting ggplot2 graphs into a plotly object. Utilizing the function ggplotly, the extra layer of the ggplotly function will convert the previously formed ggplot2 graph into a plotly object, making the plot more presentable, interactive, and flexible for stylistic changes. But, plotly also has its own function plot_ly, which transforms data into a plotly graph independent of ggplot. In short, by giving the user more freedom in data visualization without long chunks of code, plotly, when learned and implemented correctly, will allow users to create better graphs and plots in a shorter period of time.

Motivation

The motivation behind using plotly is, as stated above, the creation of graphs and charts that not only better present the data at hand, but also allows readers to interact with and better understand the data being presented to them. Not only is plotly easy to pick-up, as examples and guidelines to use plotly is explicitly given in the website *plot.ly*, it is easy to use, as long lines of codes are not necessary to build these codes. Furthermore, plotly gives a vast amount of options to the user, as the user can create diverse charts and graphs including, but not limited to, scatter and line plots, pie charts, error bars, 2D histograms, box plots, contour plots, heatmaps, polar charts, candlestick charts, and even 3D charts. By providing the user with more options, plotly will help you in deciding and developing the best data visualization method.

Background

Plotly was founded in 2012 by Alex Johnson, Jack Parmer, Chris Parmer, and MAtthew Sundquist in Montreal, Quebec. Plotly was initially built by Python, and offers various products, of which we will be exploring Plot.ly, which provides the graphical aspect of representing and analyzing data. The most basic function of plotly in R is to convert ggplot2 graphs into a plotly object. Therefore, the packages "ggplot2" is also a very fundamental part of plotly. Therefore, moving into the more technical parts of plotly and implementing plotly functions in examples to be given below, the packages "ggplot2" and "plotly" are necessary. Therefore, download the code with the below code chunk:

Preparation

Before we get started with examples using plotly, remember to load all packages necessary for the examples. I have created a code chunk for all necessary packages that must be loaded for this post:

```
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

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

```
## The following object is masked from 'package:graphics':
##
##     layout
```

Furthermore, the version of my RStudio is Version 1.0.153. Keep this in mind when replicating my result!

Moreover, the dataset we will be working with is *mpg*, which is contains information on the fuel economy data that can be found at http://fueleconomy.gov. You can check the structure of the dataset as shown below:

```
str(mpg)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr  "f" "f" "f" "f" ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr  "p" "p" "p" "p" ...
##  $ class       : chr  "compact" "compact" "compact" "compact" ...
```

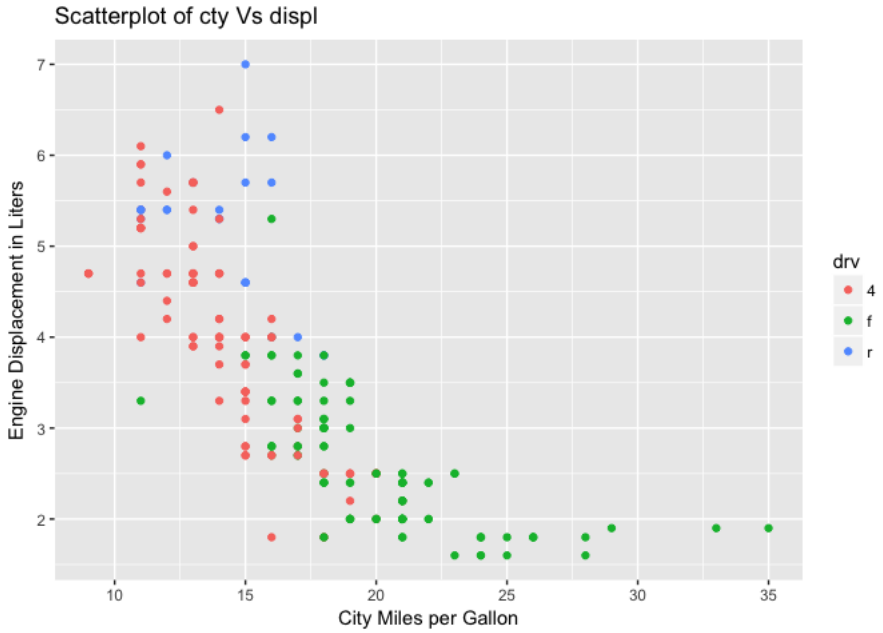
This dataset has 234 rows and 11 variables. The variables are described as shown below: Manufacturer represents the manufacturer, model represents the model name, displ represents engine displacement in liters, year represents the year of manufacture, cyl represents number of cylinders, trans represents the type of transmission, drv represents the type of drive as shown by f = front-wheel drive, r = rear wheel drive, 4 = 4wd, and cty represents city miles per gallon, hwy represents highway miles per gallon, and fl represents fuel type, and class represents the type of car. Now we are ready to work with plotly!

Examples

Comparing ggplot and plotly

First, let us compare a ggplot graph with a plotly graph that converts the ggplot graph into a plotly object. To do so, let's first create a scatterplot that compares engine displacement by the city miles per gallon.

```
ggplot(mpg, aes(x = cty, y = displ)) +
  geom_point(aes(col=drv)) + # Color the graph by the model name
  labs(y="Engine Displacement in Liters",
       x="City Miles per Gallon",
       title="Scatterplot of cty Vs displ")
```



The above graph gives good insight about how a greater city miles per gallon implies a lower engine displacement in liters.

Now let's change this ggplot into a plotly object using the ggplotly function, as shown below:

First lets take a look at ggplotly function

```
str(ggplotly)
```

```
## function (p = ggplot2::last_plot(), width = NULL, height = NULL, tooltip = "all",
##     dynamicTicks = FALSE, layerData = 1, originalData = TRUE, source = "A",
##     ...)
```

In the above output, p represents the ggplot object. Therefore, the only function command you need to make to convert the ggplot object into a plotly object is to run it through the ggplotly function as will be shown below.

```
gg <- ggplot(mpg, aes(x = cty, y = displ)) +
  geom_point(aes(col=drv)) + # Color the graph by the model name
  labs(y="Engine Displacement in Liters",
       x="City Miles per Gallon",
       title="Scatterplot of cty Vs displ")

ggplotly(gg)
```

Although the graph above may look very familiar to the graph we created before, upon some inspection, the difference will become apparent. First of all, by hovering above the points, specific information about the drv, and the number corresponding to city miles per gallon and engine displacement in liters of that specific point will be given. Furthermore, the user can also zoom specific areas by clicking the second item on the top and selecting the area in which he or she want to zoom. Furthermore, there are many more options for the user to visualize and analyze the graph, as the graph can be moved and manipulated to the liking of the user. Therefore, it becomes quite apparent that the graph created through plotly is more efficient, as it gives the reader the freedom to manipulate and interpret the graph to better suit his or her need.

Unique graphs that can be created directly through plotly

Now we are going to explore the plot_ly function that directly transforms data into a plotly visualization without having to create a ggplot. As an example, we will explore the plot I found most interesting, which is the 3D plots. Before we get started, let's explore the function plot_ly first:

```
str(plot_ly)
```

```
## function (data = data.frame(), ..., type = NULL, color, colors = NULL,
##     alpha = 1, symbol, symbols = NULL, size, sizes = c(10, 100), linetype,
##     linetypes = NULL, split, frame, width = NULL, height = NULL, source = "A")
```

While all the other arguments of the plot_ly functions deal with the aesthetics, which can you can explore by running the command "?plot_ly", the two most fundamental part of this function is data and type. The data argument requries a data frame that will be visualized and the type of visualization that the plotly function should invoke. As we are trying to construct a 3D plot, we will pass the argument for the x,y,z value and the type will be 'mesh3D'. Here are the commands to build a 3D plot.

```
ax <- list(
  title = "cty"
)

ay <- list(
  title = "hwy"
)

az <- list(
  title = "displ"
)

plot_ly(x = ~mpg$cty, y = ~mpg$hwy, z = ~mpg$displ, type = 'mesh3d') %>%
  layout(scene = list(xaxis = ax, yaxis = ay, zaxis = az), title = '3D Plot of City Miles per Gallon Vs Highway Mi
les per Gallon \n Vs Engine Displacement in Liters')
```

Here we created a 3D plot that compares engine displacement in liters based on city miles per gallon and highway miles per gallon. As for features of the graph, the user can not only rotate, zoom, and hover the graph to either get a better visualization of the graph or to get the specific numbers corresponding to displ, hwy, and cty, but also download the graph as a png file. Therefore, by plotting the the three variables as shown above, I was able to make the analysis that engine displacement grows as city miles per gallon and highway miles per gallon grows. Therefore, we can conclude that engine displacement has a positive correlation with city miles per gallon and highway miles per gallon.

All the above graphs should be reproducible in your rmarkdown file as well, so try it yourself, and try manipulating the code to your liking!

Discussion

I hope that the above examples proved my point that using the functions of plotly is much more beneficial in visualizing data than the ggplot2 package that we learned in class. I was able to make some very key inferences and analysis easier through the use of the plotly package. Furthermore, if you still feel like ggplot2 is a better visualization when compared to plotly, just remember that the plotly package allows you to change your ggplot into an interactive graph by the ggplotly function. Moreover, although I only delved into using the ggplotly function and creating a 3D plot by using the plot_ly function as I introduce the plotly package, I hope you also try to delve deeper into all the options and graphs that are made availabe to you through the plotly function. Also, if you are also knowledgeable in other programming languages and/or statistical programs such as MATLAB, Python, Pandas, etc., you could use the same knowledge you gained in building data visualization in R to these other languages and programs. Therefore, take the time to explore this visualization technique as it will become very helpful if you are to work with data later on.

Conclusions

As shown in the examples given above, the various functions within the plotly package allows you to visualize data in a much more concise and efficient manner. As an example, we specifically explored the application of the ggplotly and plot_ly functions, where each function was able to plot and make interactive the data given by the user. Therefore, I found these functions to be very useful, where knowing these functions and their proper applications could help you especially when you need to present data to others in a manner that is not only aesthetically pleasing, but useful in terms of data analysis. Furthermore, I find it very important for users to always be exploring other option that would make their data analysis and represntation skills better and more efficient. This allows the codes written by the user to be more readable and accessible to others who might be interested. Moreover, as stated before, I strongly believe that spending more time to be more efficient in your coding and visualization of data will in the long run be very beneficial for you. Therefore, going further from the scope of this material and lecture, I encourage you to be more curious and look for various methods of visualization. I hope that my introduction of plotly, a new visualization method, helped spark that interest in your endeavors in creating better data visualization.

Refereces

Blog Posts

- https://plot.ly/ggplot2/box-plots/
- https://plot.ly/ggplot2/animations/
- https://plot.ly/ggplot2/getting-started/
- https://plot.ly/r/
- https://plot.ly/r/3d-axes/
- https://www.analyticsvidhya.com/blog/2017/01/beginners-guide-to-create-beautiful-interactive-data-visualizations-using-plotly-in-r-and-python/

Videos

- https://www.youtube.com/watch?v=v_kK5c0QUnU
- https://www.youtube.com/watch?v=n4tx4-RRhdU

Data Set

- http://ggplot2.tidyverse.org/reference/mpg.html