Post02: Data Visualisation with Package Plotly in R

Chloe Lin

2017/12/2

Introduction

R is a powerful programming language in terms of data science because of its extensive open-sources available to public. It is common that there are more than one packages serving the similar porposes. So far, we have been dealing with `ggplot2` and taking advantange from its efficient functions to achieve *data visualisation*, but it still has limits. One thing that `ggplot2` cannot do is to make graphs **interactive**. The package `plotly` we are going to explore today, is the answer for this requirement. In addition, `plotly` is also compatible with `ggplot2`, which means that it includes functions to transfrom still image made from `ggplot2` to interactive ones. In this post, we will see how `plotly` can extend our abilities in designing fine data visualisations.

Preperation

In order to manipulate all the data and funtions freely through our process, we need to load all packages and set up all data we need at the beginning.

Library

The required packages are what we mentioned:

- `ggplot2`
- `plotly`

We can use the in-built function `library()` to load them all.

```
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
```

Data

For **computationally reproducible** purpose, I will use datasets provided by R itself, so they are available for everyone. I will also include information about my Rstudio settup and versions of different packages:

- RStudio version 1.0.153
- package ggplot2 version 2.2.1
- package plotly version 4.7.1
- dataset USArrests
- dataset diamonds
- dataset midwest
- dataset valcano

We can again simply use the in-built function `data()` to load all the data:

```
data("USArrests")
data("diamonds")
data("midwest")
data("volcano")
```

With these two steps completed, we are ready to start our interactive `plotly` experience!

Basic Visualization

Components

Recalling our methods of creating plots with `ggplot2`, we need two main elements to completely deliver our graphs–the layout, and the plot type such as histogram, scatterplot, etc. Similarly, there is no extra effort we need to pay for using `plotly`.

1. `plot_ly(data, x, y, type, mode, color ,size )`

- For the plot part, `plot_ly()` is the function that we need to call. In this function, we need specify the name of the data frame, the variables for x- and y-axes and the type of the plot, etc.

2. `layout(plot ,title , xaxis = list(title ,titlefont ), yaxis = list(title ,titlefont ))`

- For the layout part, `layout()` is the function that we can use to modify the look. In this function, we need to apply the plot name first, so that it knows which plot we are editing, and then we can adjust all kinds of related facts, such as title, legends and labels.

Examples

Now, knowing how to use the `plotly` functions and arguments, we can see some examples of such basic interactive plots. Not surprisingly, we can click on or drag various parts of the graphs and see how they can be changed and modified through our operations.

Histogram

```
#plotting a histogram with area variable and storing it in hist
hist<-plot_ly(midwest, x=midwest$area,type='histogram')

#defining labels and title using layout()
layout(hist,title = "Midwest Dataset vs Area",
xaxis = list(title = "Area"),
yaxis = list(title = "Count"))
```

Boxplot

```
#plotting a Boxplot with caret variable and storing it in box
box<-plot_ly(diamonds, y=diamonds$carat,type='box',color=diamonds$cut)

#defining labels and title using layout()
layout(box,title = "Diamonds Dataset - Caret-Cut Boxplot",
yaxis = list(title = "Caret"))
```

Scatterplot

```
#plotting a Scatter Plot with Murder and Assault variables and storing it in scatter
scatter<-plot_ly(USArrests, x=USArrests$Murder,y=USArrests$Assault,type='scatter',mode='markers', color = USArrest
s$UrbanPop, size = USArrests$Rape)

#defining labels and titile using layout()
layout(scatter,title = "USArrests Dataset - Murder vs Assault",
xaxis = list(title = "Murder"),
yaxis = list(title = "Assault"))
```

Advanced Visualization

It is true that basic plots seem not very different with interactive attribute, but in more advanced plots, such as heat map, or plots with more dimensions, we can have much better understanding about those plots if we can tweak or move those graphs. And now, we are going to see how those plots will be displayed.

Examples

Heatmap

```
plot_ly(z=~volcano,type="heatmap")
```

3D Scatterplot

```
#Plotting the diamonds dataset in 3D
plot_ly(diamonds, x=diamonds$depth,y=diamonds$table,z=diamonds$price,type="scatter3d",mode='markers',size=diamonds
$carat,color = diamonds$cut)
```

3D Surface

```
#Plotting the volcano 3D surface
plot_ly(z=~volcano,type="surface")
```

Using plotly with ggplot2

As we mentioned, one of the most outstanding advantage of `plotly` is that it is perfectly compatible with `ggplot2`. In this way, we do not have to give away any of our `ggplot2` knowledge, but can add interactivity to the plots with trivial steps.
Generally, we just build a ggplot graph as usuall and save it in a varaible, and then we call the function `ggplotly()` with the new plot variable to realize the conversion.

```
p <- ggplot(diamonds, aes(cut, fill = clarity)) +
  geom_bar(position = "fill")
p <- ggplotly(p)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

```
p
```

Since `ggplotly()` function turns the plot into a plotly item, we are now able to design its layout completely under `plotly` environment. For example, we could simply apply the `layout()` function, or we could add a rangeslider with relative function as well.

Conclusion and Message

Until now, we have covered the basic and some advanced visualisations of the package `plotyly`, and we also see how convenient it is to transform the ggplot results into interactive images. However, these are just part of the content of the whole `plotly` capacity. In addition, this tool can not only be used in R, but also in other programming languages, such as Python, Perl, MATLAB, etc. Therefore, when other data visualisation tools like `ggplot2` are not sufficient, we could take advantage of this package and create interactive plots that are becoming more and more important in data science field.

References

1. https://plotly-book.cpsievert.me/index.html
2. https://plot.ly/r/getting-started/
3. https://www.analyticsvidhya.com/blog/2017/01/beginners-guide-to-create-beautiful-interactive-data-visualizations-using-plotly-in-r-and-python/
4. https://beta.rstudioconnect.com/jjallaire/htmlwidgets-plotly/htmlwidgets-plotly.html
5. https://www.rdocumentation.org/packages/plotly/versions/4.7.1/topics/plot_ly
6. http://neondataskills.org/R/Plotly
7. http://www.htmlwidgets.org/showcase_plotly.html