AIRLINE TRAVEL VISUALIZATIONS & STATISTICS

Abstract

The package I will be exploring for my STAT 484 project, will be R's plotly library. I will be using the R graphing package to make interactive graphs, displaying information pertaining to airliner travel data.

By: Eric Brown

Table of Contents in Reference to RPubs Page of Airline Data Project

Introduction & Example 1

For this first example, we are going to explore the creation and usage of bar charts in R's plotly graphing library. However, it is imperative that one goes to the reference section of this paper and clicks on the link below *2020passengerData1* dataset to download a copy of the airline input data used throughout this example code. Once, you have completed this action you can begin, to use the code below to import the excel worksheet into R.

To start one might be wondering, what exactly does the data in the excel file called 2020passengerData1 represent. So in response, the data in the input statement above represents the passenger attendance information for all commercial service airports in the United States. We are fortunate that the data in this excel dataset is somewhat sorted already. However, if we had to sort the data, we could easily sort it using R's order() function and insert a dataframe's column name inside of the function to sort by either descending(default) or ascending order. Just make sure you have a "," written before the order function to make sure R sorts the data by columns instead of rows. I made this mistake once.

However, in return to this example, lets say we wanted to find out which airports had the most passenger enplaning flights during the year of 2020. To display this finding, we could create a bar chart that shows the top 20 busiest airports in the year of 2020. And to create this bar chart in plotly, first we would have to select which part of the dataframe to select as our x and y axis. Since we want to show the top 20 busiest airports for the year 2020, we would need the column named CY20_Enplanements from the passengerData dataframe, that we created above.

As noted, the code shown above does this exact task and stores the airport names and passenger enplanements in a new dataframe called topAirports. This code is important because we will use this variable to get both our x and y axis arguments. Which are needed arguments for our plotly function to create our bar chart. After the creation of the variable, topAirports, we need to index both the airport names column(*topAirports$Airport_Name*) for our x-axis and the passenger count column(*topAirports$CY20_Enplanements*) for our y-axis. Now we are ready to create our first bar plot in plotly.

Using the bar chart above, we can make an observation that the airport who has the most number of passengers is Georgia's Hartsfield-Jackson Atlanta International Airport. And if you hover over the Hartsfield-Jackson Atlanta Intl. airport bar on the chart above, you will see that it has about 21 million people travelling through the airport in the year of 2020. This is a pretty high number, considering the US was dealing with the coronavirus pandemic at the time.

While the bar chart above is a useful plot that plotly can provide, lets see if we can make it little more presentable with the following code.

Now on the chart above, we have created a title header for the chart that establishes what the data represents, and we created axis labels for both the x and y axis. All of these additional graphics were created through plotly's layout function inside of the initial plotly function. The layout function in plotly is also able to establish a numerous range of plot properties that can alter how your plot is displayed in its output. For example the layout function can determine a plot's height, margins, colorway, annotations, legends, and etc…

Feel free to take a look at all the features the layout function provides in plotly with the following link (https://plotly.com/r/reference/layout/).

However, one graphic you might have not noticed that we added to the plot, was hovering labels. If you hover over any of the bars in the plot above, you will see that the size of airport is displayed in a text box below the passenger count number. This feature was created through the text argument that plotly functions provides. All one has to do to use this feature is create a character type variable and assign it to the text argument in plotly's function to create this hovering display. Another feature you may not have noticed is that the Hartsfield-Atlanta Intl Airport bar is now colored red. This feature makes the highest bar value stand out more from the rest of the bars in the bar plot. One could easily pick out Atlanta's Intl airport has the busiest airport out of the 20 airports displayed in the plot. The max color feature was created through the marker argument of the plotly function. All one as to has to do to use this feature is create a list of color values to pass to the marker argument to color each bar a certain color in the final output.

While these features all enhance the display of the previous bar plot of top 20 busiest airports in the US. There is one concept we have not touched on that plotly can do for bar plots. This feature involves grouping data to show grouped bar charts. Grouped bar charts can be handy when you want to see change in numeric values between intervals of a given variable. For example, say we wanted to see which airport had the most change in passenger enplanements between the years of 2019 and 2020. In order to accomplish this task, we need to use the following code:

Look at that, we were able to create a grouped bar plot using plotly. Now we are able to see that Hartsfield-Jackson Atlanta Intl airport had the most change in passenger attendance between the years of 2019 and 2020. I guess when you have the most number of passengers travelling through your airport, it can be adversely affected when external factors play a role in global travel. Such as the covid travel restrictions the world faced in 2019 to 2020.

In order to create this plot, most of code used from the previous bar plot stayed the same. However, we did have to add an add_trace function. Since this method accepts a plot_ly figure trace and adds it to the figure. Which allows you to start with an empty figure again, to add traces to the figure sequentially. Which is why we were able to add the same code structure of the plotly function inside of the add_trace function. With the only addition of the line assignment statement to make a defined distinction between the two grouped bars, even though the color of the grouped bars were different.

Example 2

So now that we explored the realm of barplots in R's plotly, lets take a look at how scatterplots are created in plotly, as well. However, to start one should use the link below the *newark_passengerData1* dataset in the reference section, in order to download the dataset used throughout this example of creating scatterplots. Once, you have completed this action you can begin to use the code below to import the excel worksheet into R.

To begin our exploration of scatterplots in plotly, we first most understand what the data outputted from the code above represents. Thus, to answer this question, the data represents the number of domestic and international flights going through Newark Liberty International airport from the years of 2002 to 2021. While the data is segmented by years it is also segmented by months, as well. The month of January is represented by a numeric value of 1 and the month of February is represented by a numeric value of 2. This number sequence continues until the numeric value of 12 is reached to represent the month of December.

As a side note, the main reason why I picked this airport is because I live in the New York metropolitan area, closer to Newark, New Jersey. So, what better way to conduct our usage with scatterplots that to answer the question whether there is a positive correlation between years and the number of passengers entering Newark Liberty Intl airport. I assumed there would be a positive one since the global population normally increases every year. But with the creation of a scatterplot, we could either confirm or reject this hypothesis.

To start we need to find and index both the number of passengers entering Newark airport and the year number that is tied to this numeric value. To accomplish this task, we need to use the following code below.

Now that you can see the dataframe that was created, we are able to access the passenger count through the with function. While using the order function to order the passenger count by year then by month in ascending order. Do note that a comma is needed after the index argument of the original dataframe, to order the data by rows instead of columns. As a result, we are ready to create our first scatterplot in plotly.

And there it is, our first scatterplot created through R's plotly library. Consequently, we now can see there is a positive correlation between the variable year and the total number of passengers travelling through Newark Liberty Intl Airport. So, this plot supports my initial hypothesis that each year there is an increase in the number of passengers that travel through Newark Airport. However, the plot also shows that there was a significant drop off in passenger attendance from the year 2019 to 2020. Which was probably because of the coronavirus pandemic, as we addressed earlier. There was also a slight decrease in passenger attendance from the year 2008 to 2010, but this decrease was marginal. So further explanation probably would not have been warranted.

With regards, to the code used for creating a scatterplot in R, one can see all you need to provide is a x and y argument, in conjunction to a data argument. However, lets try to get more detailed and add axis labels and custom color scales to the data points based on months. This way we could see if the number of passengers increases the further we are in a given year.

Now the data points seem so vibrant and clear. We can even see for the most part, as the months progress in every year, the number of passengers increases, as well. However, there is some deviation from this observation, though. Since in every year the highest number of travelers travelling through Newark International Airport, seem to happen in the months of August-8 and September-9. And not November-10 through December-12, which is probably because of summer break for most people. More data would have to analyzed to conform to this hypothesis, though.

With regards to the code, all we had to do was add a color argument to the plotly function to let the compiler know how to color the data points. In our case, we made the color scheme related to the months column of the dataframe. Then we added a mode argument to the plotly function to specify that markers would be displayed instead of lines. Lastly, we added a forward pipe operator to pass the function into the layout function, which had all of plot's properties, such as labels and title. We could even make this plot more statistically significant by segmenting the datapoints by travel type instead of months. This means we would have a scatterplot for Domestic & International travelers, instead of the total passenger count for both types of travelers. However, I will leave that scatterplot creation up to the reader to create.

Example 3

With this example we should take some time to explore how plotly is able to display the distribution of a dataset using boxplots. Just in case you were wondering, boxplots are able to visually show the distribution of numerical data and skewness through the five-number summary, which includes the sample minimum, lower quartile, median, upper quartile, and sample maximum.

Now that we got a basic understanding of what boxplots are used for, lets begin with our example. Say for instance, we wanted to get a better understanding on how many flights were delayed because of weather at Newark Liberty International Airport. We could create a boxplot for this example using the following code:

As one can see, the code above has various columns that show the number of delays that happen because of various reasons at Newark Intl Airport. I chose weather delays to make our analysis of the data simpler to dissect for our boxplot exploration. But one could select another column for another reason why a flight was delayed.

Again, it is imperative to download this dataset using the link below the *newarkAirportDelay_2019_2020* dataset in the reference section of this paper. In order, to have access to the data used throughout this example. One must also set their working directory to a file path similar to one displayed in the code above, to use the file import statement, as well. Once you have downloaded the dataset from the website provided and imported it into R, you can begin using the plotly function to create a boxplot. To find out how many flights are affected by weather delays at Newark Intl. Airport.

The plot above is a basic boxplot that plotly can create using the Newark Intl Airport weather delay data, which we gathered from the dataset above. The only arguments you need to provide to create a boxplot of this simplicity is a dataset name and a y-axis variable. The dataset name we already discussed was created from the Newark Airport weather delay comma delimited file. And the y-axis variable was created from the variable inside of the Newark Airport weather delay dataframe, which had the number of flights delayed by weather issues at Newark Airport. Lastly, the type argument instructed plotly to create a boxplot for display instead of other plot types.

While this plot would be sufficient to get a basic understanding of the distribution of weather delays at EWR airport, lets try to use our boxplot to incorporate the data we gathered in the previous examples, to conduct another analysis. Lets try to discover the distribution of weather delays at the top 20 busiest airports, that we found in example one. Consequently, to conduct this analysis, we are going to have to download and import data from the link provided below the *allAirportDelay_2019_2020* dataset in the reference section of this paper.

Now that we have the dataset containing the top 20 busiest airports weather delays, imported into R, we can begin to create our boxplot. In order to accomplish this task we will use the following code:

Look at that we were able to create a boxplot with the top 20 busiest airports weather delays, displayed on the plot. Now we can take a look at the distribution of weather delays between airports. And right away we can see that Dallas Forth-Worth Intl airport had the largest distribution of weather delays and had the highest upper quartile (q3) value of 1108 delays. While Salt Lake City Intl airport had the lowest quartile (q3) value of 288.5 delays. More observations can be made from this plot, as well. Which further backs its significance in plotly arsenal of statistical plotting capabilities.

The code for creating this boxplot is pretty concise, as well. Since, all you have to do to add more boxplots to a given plot is to add a color argument. Which states how you want plotly to group the y values listed in the function. In our case we wanted to show each airport that was displayed in example 1, in the plot with the airport's own boxplot. So, all we had to do was add an argument that had the names of all the airports, listed in the dataframe. The column/variable airport_name contained this information. With this argument plotly additionally assigned each boxplot with its own individual color. Which made each boxplot stand out even more from each other.

Example 4

For our last example we are going to explore the realm of heatmaps. In case you were wondering generally a heatmap is a graphical representation tool that uses color-coding schemes to represent different value types. While heatmaps are commonly used to understand user behavior on online websites, we are going to use heatmaps in this example to show which flights provided by United Airlines, generated the most passenger attendance.

To make this example less cumbersome, we are only going to take the top 100 flights that United Airlines provided in the month of August in the year 2021. Upon completion of our heatmap, we will be able to see which flights United Airlines probably should keep in the month of August. The data we will be using will be provided from the link below the 2021_MARKET_ALL_CARRIER dataset in the reference section of this paper. So please download the dataset onto your computer and then precede to follow the code below.

One might wonder what the code above entails, so lets start with the *unitedDes* dataframe. This dataframe subsets the initial imported dataset by airline named United Air Lines. We need to create this dataframe first because we do not want to conduct our analysis on airlines that are not named United Airlines. It would lead to false observations if we did otherwise. In conjunction to slicing our initial imported dataset by airline name, we also needed to get flights only scheduled in the month of August. Therefore we added another conditional argument of the month variable equaling the number 8; 8 represents August in numerical month order of the calendar. Then we included the next assignment statement of the *unitedDes* dataframe to order the flights with the most passengers on top. The "-" sign is key, to sort a variable/column in descending order, instead of ascending order. Finally, the last statement in the code above uses the head function to get the top 100 flights that United Airlines provides in the month of August. Now we are ready to create our heatmap.

Now that the heatmap has been created, it is first a good idea not to get overwhelmed by the different rectangles being displayed in the plot. Since the legend shows us the colors that are closest to yellow are flights that have the most passenger attendance. Thus, making it a popular flight for United Airlines to keep on their schedule for the month of August.

So, when we look at the heatmap, we can see that flights that start out at Denver Intl Airport and stop at Orlando Intl Airport (ORD) are the most popular in the month of August. Just around 53,000 passengers used this flight path offered by United Airlines in the month of August. The next best flight offered by United Airlines was flights going from Orlando Intl Airport to Denver Intl Airport, with a total of around 51,000 passengers. Which makes sense because passengers going from Denver to Orlando eventually need to come back to their home state. And while some of those passengers might live in other states, most of them probably live in the same state, they boarded their initial flight.

To create this heatmap in plotly, all we needed to provide was four arguments to the plotly function. The first argument was the x-axis value, in our case it made sense to make this argument equal to the origin of the flight. The second argument was the y-axis value, and for this value the destination airport location, needed to be provided for the origin airport. Then the third argument which we needed to provide included what our unit of measurement was going to equal. Now keep in mind plotly conducts heatmaps in terms of a matrix structure. So, the z value would be the only value that could have any numeric representation. In our case, we needed to get the passenger count of each flight, so the passenger variable was needed from our user created dataframe, *topUnitedDes*. Lastly, the fourth and final argument we needed to provide was the type of plot needed for display, which was a heatmap.

References & Datasets Used:

(2020passengerData1) – Dataset found using link below:

https://www.faa.gov/airports/planning_capacity/passenger_allcargo_stats/passenger/

(newark_passengerData1) – Dataset found using link below:

https://www.transtats.bts.gov/Data_Elements.aspx?Data=1

(newarkAirportDelay_2019_2020 & allAirportDelay_2019_2020) –
Dataset found using link below:

https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E

(2021_MARKET_ALL_CARRIER) – Dataset found using link below:

https://www.transtats.bts.gov/DL_SelectFields.asp?gnoyr_VQ=GED&QO_fu146_anzr=Nv4%20Pn44vr45