Plotly

- Plotly provides online graphing, analytics, and statistics tools. Using their technology we can make interactive web-based graphs.
- Plotly is a high-level interface to the open source JavaScript graphing library plotly.js.
- Plotly for R runs locally in your web browser or in the R Studio viewer. You can publish your charts to the web with plotly's web service.
- There is excellent documentation at https://plot.ly/r/reference/ that will allow you to go through all of the graphing options of this package.
- We will go through some highlights here.
- To begin we will use the dataset 'txhousing' which is pre-loaded into R.
- Some functionality of plotly can be accessed fairly automatically through ggplot. Let us start with making a basic plot in ggplot (storing it) and "plotify" it.

```r
## Get started with a scatterplot
require(plotly)
require(ggplot2)
require(RColorBrewer)

g1<-ggplot(data=txhousing)+
  geom_point(aes(x=listings, y=volume))

ggplotly(g1)


# What if we would like the text to include more information
# Specifically, we would like it to also display
# the year of the sale
g2<-ggplot(data=txhousing)+
  geom_point(aes(x=listings, y=volume,
                 text=paste("Median", median, "<br>Listings:",
                            listings)))

#Text will be everything listed plus defaults
ggplotly(g2)
#Text restrained to what you specified above
ggplotly(g2, tooltip="text")
```

- To plot directly in plotly (and not through ggplot2), we need to work with new syntax. Some plots are only possible in plotly so it is worth it to try to understand some of syntax with basic examples.

```r
plot_ly(data=txhousing,
        x = ~listings,
        y = ~volume,
        type="scatter",
        mode="markers",
        marker=list(size=5, opacity=0.5)) %>%
  layout(title="Texas Housing Data by City",
         xaxis=list(title="Number of Listings"),
         yaxis=list(title="Total Value of Sales"))

# What if we want the color and the size of the marker to be a function of the median price:
plot_ly(txhousing,
        x = ~listings,
        y = ~volume,
        color = ~median,
        size = ~median,
        type="scatter",
        mode="markers")

# What if we would like the text to include more information
# Specifically, we would like it to also display
# the year of the sale

plot_ly(txhousing,
        x = ~listings,
        y = ~volume,
        color = ~median,
        colors="RdPu",
        size = ~median,
        type="scatter",
        mode="markers",
        hoverinfo="text",
        text=~city)
```

- 3-dimension scatter plots:

```
plot_ly(txhousing,
        x = ~listings,
        y = ~volume,
        z = ~median,
        type="scatter3d",
        mode="markers",
        text=~year,
        marker=list(size=5, opacity=0.5, color="red"),
        opacity=0.1)
```

- Barcharts

```
# Use diamonds data set to plot a bar chart
# You must specify the y height

require(dplyr)
diamonds_tally <- count(diamonds, cut, clarity)

plot_ly(diamonds_price,
        x = ~cut,
        y = ~price,
        color=~clarity,
        colors="Set2",
        type = "bar", opacity=1)
```

```r
#What if we want to plot the mean price
#at every combination of cut and clarity

diamonds_mean<-summarise(group_by(diamonds, cut, clarity), price=round(mean(price),2))

plot_ly(diamonds_mean,
        x = ~cut,
        y = ~price,
        color=~clarity,
        colors="Set2",
        type = "bar",
        opacity=0.7)
```

- Box plots

```r
# Boxplot for one quantitative variable
plot_ly(diamonds,
        y = ~price,
        type = "box",
        line=list(color="purple"))

# Boxplot of a quantitative variable across multiple groups
plot_ly(diamonds,
        y = ~price,
        color =~cut,
        type = "box")
```

Visualizing Matrix Data with Plotly

- Mount Eden is a volcano in the Auckland volcanic field. The volcano dataset gives topographic information for Mount Eden on a 10m by 10m grid.
- One way to look at the topographic data is via a heatmap. The heatmap's color pattern visualizes how the height of the volcano's surface fluctuates within this 10m by 10m grid.
- Alternatively, you could visualize the data by making a 3D surface plot. Namely, plotly visualizations don't actually require a data frame. This makes chart types that accept a z argument especially easy to use if you have a numeric matrix such as the volcano dataset.
- Let's create a heatmap and 3D surface plot.

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

# The 3d surface map
plot_ly(z=~volcano,
        type = "surface")
```