As someone very interested in storytelling, ggplot2 is easily my data visualization tool of choice. It is like the Swiss army knife for data visualization. One of my favorite features is the ability to pack a graph chock-full of dimensions. This ability is incredibly handy during the data exploration phases. However, sometimes I find myself wanting to look at trends without all the noise. Specifically, I often want to look at very dense scatterplots for outliers. Ggplot2 is great at this, but when we’ve isolated the points we want to understand, we can’t easily examine all possible dimensions right in the static charts.

**Enter plotly.** The plotly package and ggploty function do an excellent job at taking our high quality ggplot2 graphs and making them interactive.

During this tutorial, we are going to explore the median reported wages of creative occupations within the city of Austin for 2016 and 2017. [This data set](https://data.austintexas.gov/Recreation-and-Culture/Median-Earnings-of-Creative-Sector-Occupations-CLL/2qxc-8cme) was sourced from the [Austin open data portal](https://data.austintexas.gov/).

**Install and Load Packages**

Before we get rolling, we need to install and load the necessary packages.

##### Install and load packages

install.packages("plotly")

install.packages("tidyverse")

install.packages("htmlwidgets")

library(plotly)

library(tidyverse)

library(htmlwidgets)

**Load The Data Set**

##### Import data

creatives <- read.csv(file="https://raw.githubusercontent.com/lgellis/MiscTutorial/master/ggplotly/Median\_Earnings\_of\_Creative\_Sector\_Occupations\_\_CLL.B.1.csv",

header=TRUE, sep=",", stringsAsFactors = FALSE)

dim(creatives)

summary(creatives)

str(creatives)

head(creatives)

**Transform the data**

We are performing some simple transformations to prep the data for an easy scatterplot. First, I renamed the columns and then I created a new column to calculate the 2016 to 2017 year over year percent changes in median wage per occupation.

##### Data processing

#Rename the columns and create a new column for year over year change

creatives <- creatives %>%

rename(Median\_2016 = X2016.Median.hourly.earnings, Median\_2017 = X2017.Median.hourly.earnings) %>%

rowwise() %>%

mutate(Percent\_Improvement = round((Median\_2016-Median\_2017)/Median\_2016\*100,2))

**Create the scatterplot**

I created a simple ggplot2 scatterplot of the occupations showing the 2017 median wage vs year over year percent improvement. I added a simple horizontal line to mark zero on the plot. This allows us to more easily digest the year over year changes in median wage. We then take one final step and input our ggplot2 scatterplot into the ggplotly function. This creates an interactive graph!

##### Create a scatterPlot

scatterPlot <- creatives %>%

ggplot(aes(x = Median\_2017, y = Percent\_Improvement)) +

geom\_point(alpha=0.7, colour = "#51A0D5") +

labs(x = "Median Occupation Hourly Wage in 2017",

y = "% Improvement Year over Year (2016 to 2017)",

title = "Austin Creative Occupations Median Hourly Wage") +

geom\_hline(yintercept=0, linetype="dashed", color = "#2C528C", size=0.5) +

theme\_classic()

ggplotly(scatterPlot)

**Add the Labels**

The above graph is great because we’ve successfully used plotly to make the ggplot2 scatterplot interactive. However, the mouseover data doesn’t have everything we want and it’s not very nicely formatted. Let’s fix this!

We simply add the mouseover details through the ggplot aesthetic text property. We then assign it to the tooltip in the ggplotly function.

##### Adding text

scatterPlot <- creatives %>%

ggplot(aes(x = Median\_2017, y = Percent\_Improvement,

text = paste(

"Occupation: ", Occupation, "\n",

"2017: ", Median\_2017, "\n",

"2016: ", Median\_2016, "\n",

"% Improvement Year over Year: ", Percent\_Improvement, "\n",

sep = ""

))) +

geom\_point(alpha=0.7, colour = "#51A0D5") +

labs(x = "Median Occupation Hourly Wage in 2017",

y = "% Improvement Year over Year (2016 to 2017)",

title = "Austin Creative Occupations Median Hourly Wage") +

geom\_hline(yintercept=0, linetype="dashed", color = "#2C528C", size=0.5) +

theme\_classic()

ggplotly(scatterPlot, tooltip = "text")

**Save the Data Locally**

Now we have the interactive scatterplot and it’s time to share it around! Start by saving locally.

##### Save it locally

htmlwidgets::saveWidget(as\_widget(p), "OccupationWages.html")

**Save the Data on plot.ly**

You may also want to share your interactive graph by sending folks a link. A super easy way of doing this is just to upload it to your plotly account.

##### Save it on ploty

#Create a plotly account: https://plot.ly/

#Create API Keys: https://plot.ly/settings/api

Sys.setenv("plotly\_username"="USERNAME")

Sys.setenv("plotly\_api\_key"="PASSWORD")

#2) Save to online file

api\_create(scatterPlot, filename = "Occupation Wages")

After running the code, the URL for your hosted plotly chart will pop up and voila, you have a hosted plotly graph! You can send it off to friends and colleagues or you can embed it into a website like I did!

**THANK YOU**

Thank you for exploring how we can use the plotly package to make our ggplots interactive.  Please comment below if you enjoyed this blog, have questions, or would like to see something different in the future.