# ggvis vs ggplot2 as a visulaization tool

## ggvis vs ggplot2 as a visulaization tool

### Introduction

Data visualization is one of the most important parts of any data driven project. If done successfully, it lets people who are new to the project understand the basics of it at a quick glance. When given simply raw numbers, tables, and facts, it is very difficult to be able to digest the information efficienty, this is where data visualization comes into play. In this class we have looked at many different packages to visualize data: the base package, ggplot2, and now ggvis.

### Motivation

Amongst the trials and tribulations of completing homework 4, I found the most difficult part of it all to be developing the Shiny App. Since it is my first time working with R, I had never used this method to create data visualization application before, or rather, I have never made anything like it in general. Even though it was the most difficult part of the homework for me, I was very fascinated with Shiny app and ggvis when we were first introduced to it in lecture and lab. Similar to my interest in ggplot2, it was very interesting to see how with ggvis as well we were able to display our results so cleanly and uniquely to anyone that interacted with it. I am dedicating this post to learning more about ggvis and comparing it to a package we have already worked with, ggplot2 to see how when it is best to use each package.

### History

Ggvis is a package that can be downloaded to R. It takes the best parts of ggplot2 and combines it with the interactive framework of shiny app and the drawing web graphics using vega. Hadley Wickham created ggplot2 back in 2005 as a data visualization package for R. His implementation of it was a result of Leland Wilkinson's Grammer of Graphics a general scheme for visualization that breaks up graphs into semantic components such as scales and layers.

### Now let's see how it works!

#### Necessities

First we have to make sure to load all the packages and datasets that we will need for the following examples

```
# if you don't already have ggvis, make sure youffff run install.packages('ggvis') first
library(ggvis)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:ggvis':
##
##     resolution
```

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

mtcars is about motor trends and car facts for various brands of cars faithful gives you information on the eruptions and waiting times for Old Faithful Gyser

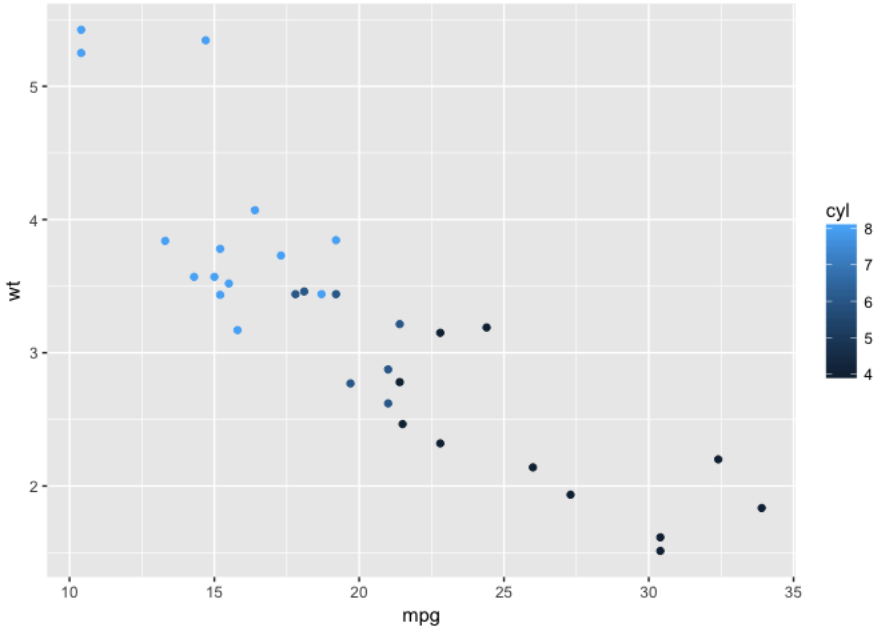
#### Syntax Differences:

If it seems like ggvis is very similar to ggplot2, that is becasue it is! Before we jump into some of its more profound differences in functionality, let's look at some of the differences in syntax between the two packages: Here is a table I made for some of the ways the syntax differs:

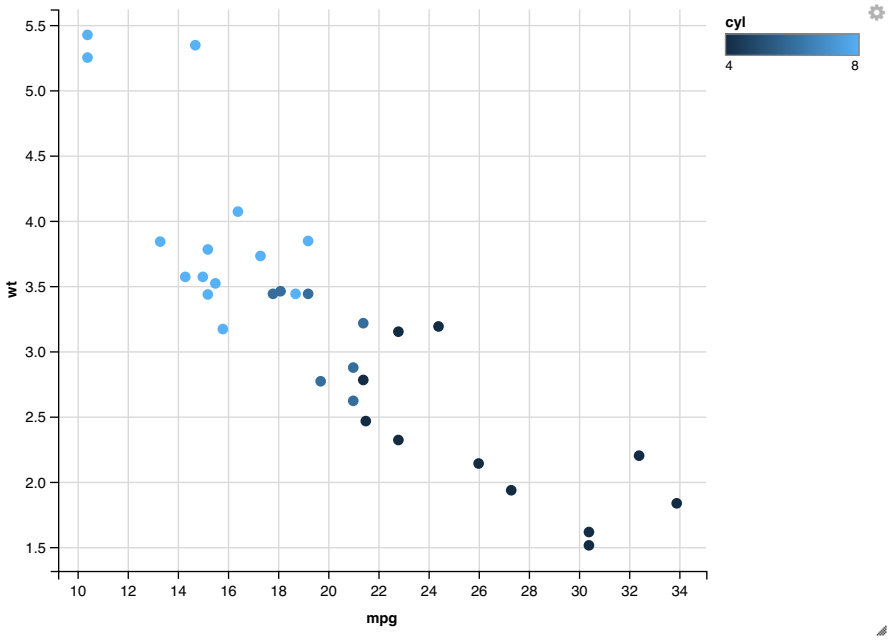
```
##                ggplot2    ggvis
## variable choice aes()      props()
## layers          Layer,geom layer function
## syntax          +          %>%
## variable syntax " "        ~
```

The table above shows some of the main syntax differences that ggplot2 differs from ggvis, in the example below, I make a scatterplot of the same data but one graph is using ggplot and the other using ggvis. Take a look at the code to see the differences in syntax but also notice how the end chart is nearly identical in style:

```
# scatterplot using ggplot2
cars <- ggplot(data= mtcars, aes(x=mpg, y=wt, colour = cyl)) + geom_point()
cars
```



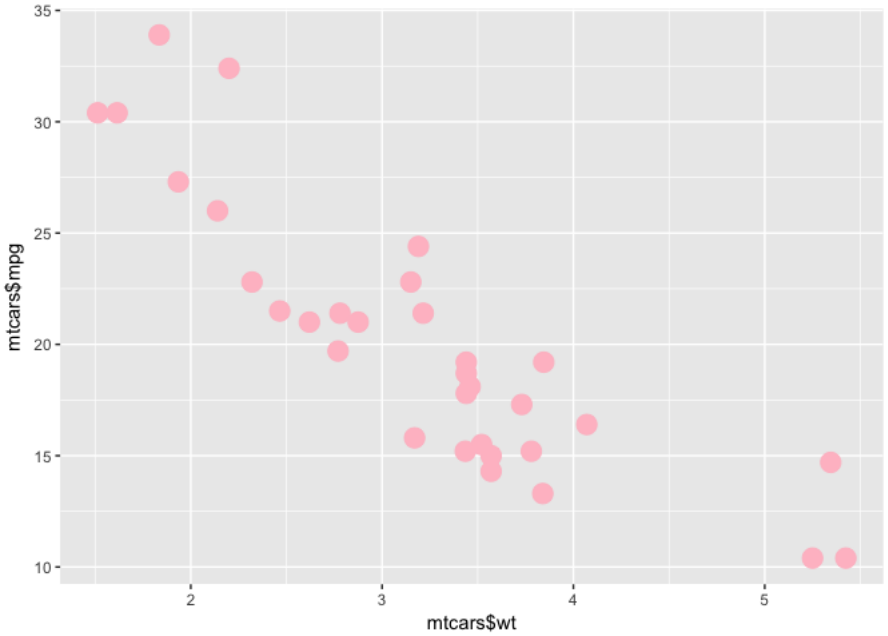

```
cars2 <- ggvis(mtcars, x= ~mpg, y= ~wt, fill= ~cyl) %>% layer_points()
cars2
```



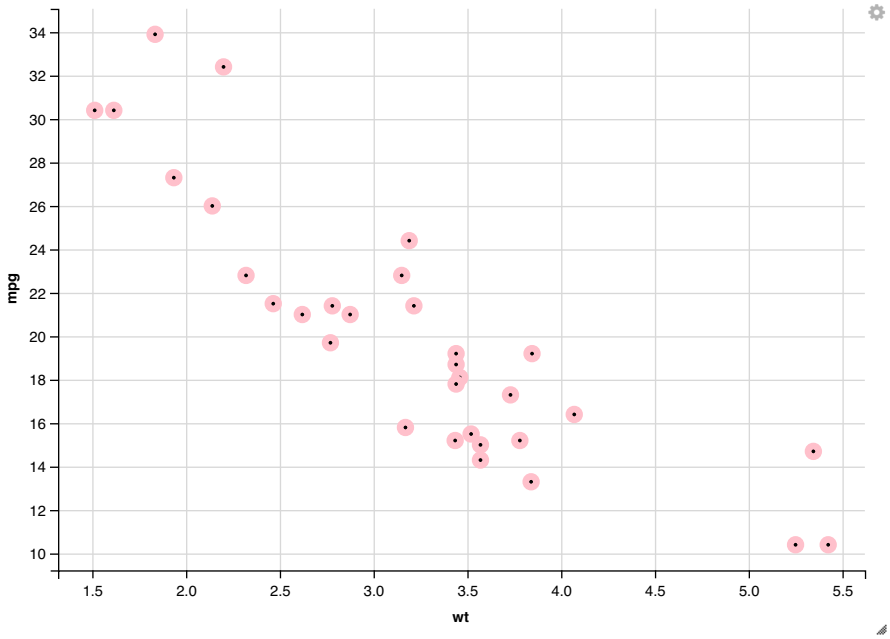

## Fundamental differences:

Now that we see the differnces in syntax, let's explore more about the processes that each package goes through when displaying data- this is where the main differences between the two packages lie. ##### levels Ggplot2 has two levels: there are data and aes specifications in each layer. Ggvis has an unlimited hierarchy. This means that you can have as many levels as you need and the data is only computed once.
The reason for these differences in levels is because of how ggplot2 and ggvis look at properties and scales. As we saw above ggvis uses the props() wrapper, this means that each mark in ggvis is associated with a set of properties that governs how it is displayed. These properties can either be constant (ex. 5, 'red', 'circle') or can be mapped to variables in the data set. Ggplot2 uses aes() function which makes a distinction between mapping variables and setting constants.

```
# the differences in 'levels' for ggplot2 and ggvis
# ggplot2:
ggplot() + geom_point(aes(x= mtcars$wt, y= mtcars$mpg), color = 'pink', size = 5)
```




```
# ggvis
mtcars %>% ggvis() %>%
  layer_points(x = ~wt, y = ~mpg, stroke := "pink", strokeWidth := 5)
```




The reason that ~ is used in the props function for ggvis is because it is the operater that says that the expression should be evalued in data because the data can keep changing in ggvis; it should be used so that the program knows to evaluate it at each unique environment rather than a constant one as ggplot2 would do.

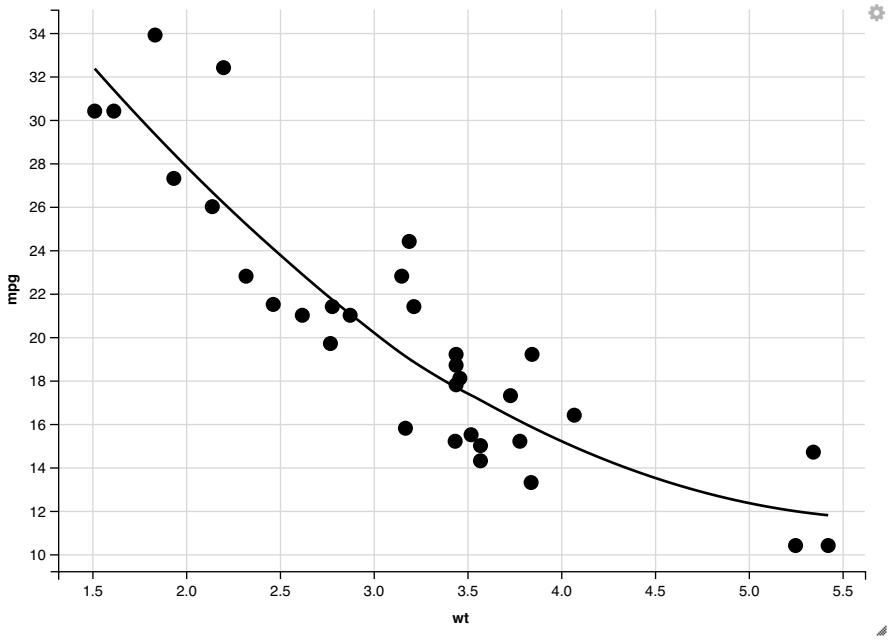
### interactivity

And finally for the most exciting and unique part of ggvis: it is interactive! We saw in HW4 how we could combine the displaying abilities of ggvis with the interactive structure of shiny app to create reactive plots. Let's first see some of the various inputs we can choose to make the data interactive: - input_checkbox(): a check-box - input_checkboxgroup(): a group of check boxes - input_numeric(): a spin box - input_radiobuttons(): pick one from a set options - input_select(): create a drop-down text box - input_text(): arbitrary text input

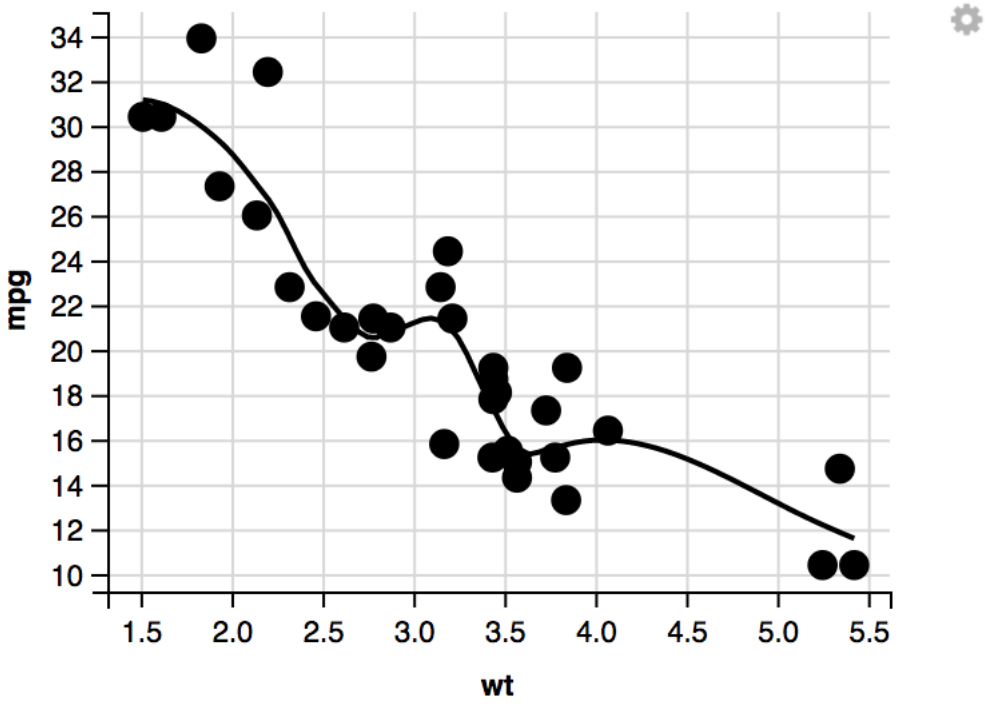
You can put whichever values you want in each slider as well as choose the default one:

```
mtcars %>%
  ggvis(~wt, ~mpg) %>%
  layer_smooths(span = input_slider(0.5, 1, value = 1)) %>%
  layer_points(size := input_slider(100, 1000, value = 100))
```

```
## Warning: Can't output dynamic/interactive ggvis plots in a knitr document.
## Generating a static (non-dynamic, non-interactive) version of the plot.
```




Since we can't see the interactivity of it unless we look at it in R, I will put photos of it here





In looking at the scatterplot between the weight and mpg of the cars in mtcars, we put in a slider to adjust how large we wanted the points to be and one for the span of the line which we can adjust as well.

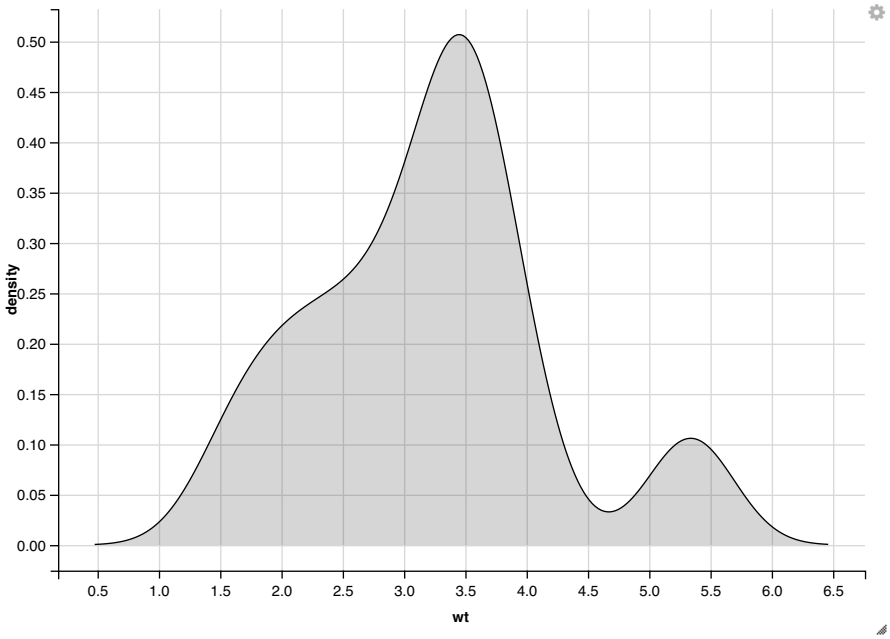
You can also put multiple interactive features on each graph:

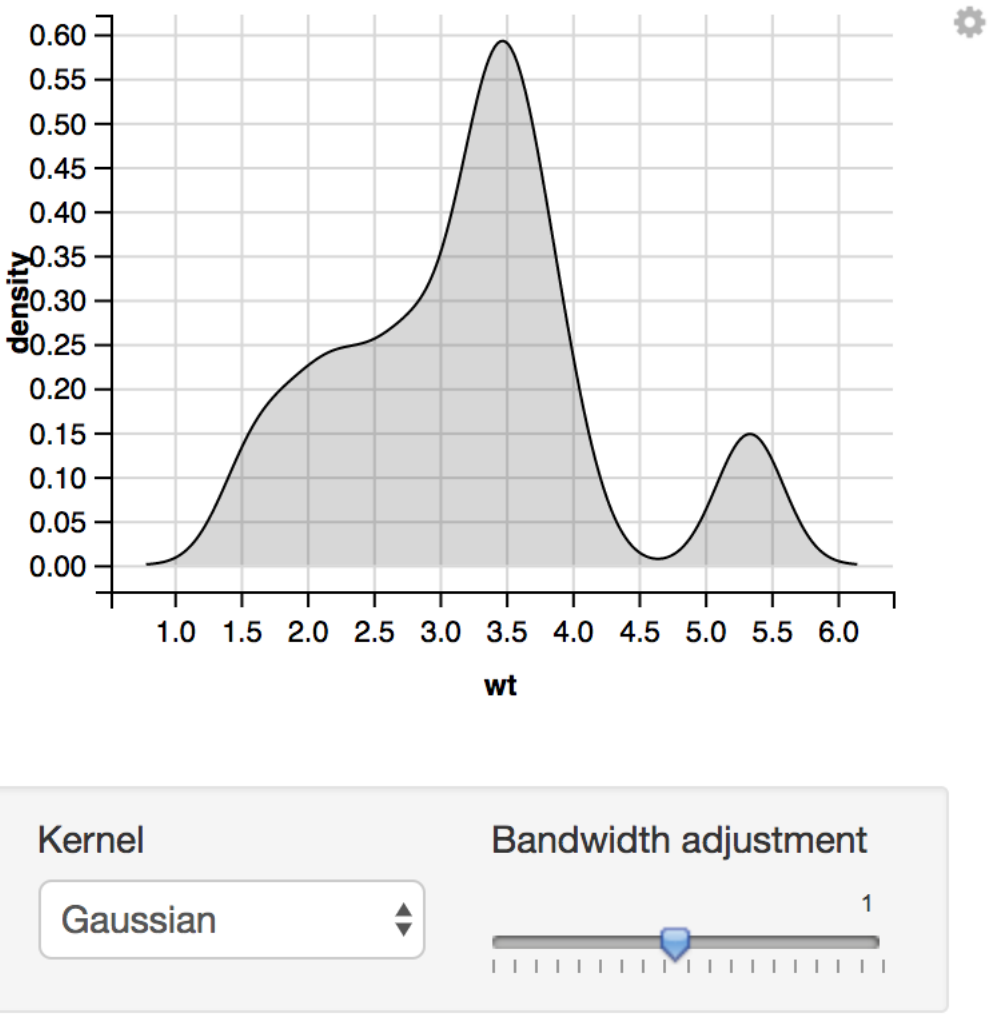
```
# slider and drop down menu
# lets choose from a sample of the mtcars data
# use set seed for reproducibility
set.seed(12); sample(mtcars$wt, 100, replace = TRUE)
```

```
##   [1] 2.320 2.140 3.570 3.150 3.460 2.875 3.460 2.465 2.620 2.620 3.730
##  [12] 2.140 3.730 3.730 3.150 5.250 5.250 2.200 3.520 3.215 3.570 1.935
##  [23] 3.215 3.435 3.570 3.150 5.345 3.570 5.250 3.520 3.190 3.170 3.170
##  [34] 2.140 2.465 3.570 3.435 2.140 3.730 3.730 1.615 5.345 2.780 3.570
##  [45] 1.513 3.215 3.730 2.875 3.440 3.845 3.730 4.070 3.780 3.440 2.200
##  [56] 2.780 2.200 1.615 2.465 2.140 3.840 3.780 2.780 2.200 3.440 3.460
##  [67] 3.440 3.845 4.070 3.845 3.440 3.435 1.835 3.730 3.440 3.570 1.935
##  [78] 3.435 3.460 2.140 2.200 3.170 3.215 3.190 3.435 3.150 2.875 3.170
##  [89] 3.730 3.215 3.520 3.440 3.460 4.070 1.615 5.424 3.845 4.070 3.440
## [100] 3.440
```

```
mtcars %>% ggvis(x = ~wt) %>%
    layer_densities(
      adjust = input_slider(.1, 2, value = 1, step = .1, label = "Bandwidth adjustment"),
      kernel = input_select(
        c("Gaussian" = "gaussian",
          "Epanechnikov" = "epanechnikov",
          "Rectangular" = "rectangular",
          "Triangular" = "triangular",
          "Biweight" = "biweight",
          "Cosine" = "cosine",
          "Optcosine" = "optcosine"),
        label = "Kernel")
    )
```

```
## Warning: Can't output dynamic/interactive ggvis plots in a knitr document.
## Generating a static (non-dynamic, non-interactive) version of the plot.
```






In this density plot of weights for mtcars, we have a slider to adjust the bandwidth and also a drop down menu for choosing various graphing methods where the options are already loaded into the input_slider function.

Here we see that where we would usually put static values for ggplot2 or for a regular plot, to make it interactive we instead put 'input_slider' or 'input_select'. These create interactive ways to change the values on the graph or to choose a different method. However there are some limitations: currently interactive inputs can only be used as arguments to transformations (ex. layer_smooth) or as properties (ex. props(size= input_slider())). This means that inputs can only modify the data not the underlying plot specification– there is no way to add or remove layers or switch between different data sets.

### Take home message

There are specific times in which each of these packages should be used. One, obviously, when trying to show the differences between different variables and being able to add and remove things, ggvis should be used because it is the only one of the two that is interactive. Ggvis is also much faster than ggplot2 especially when changing data because ggvis is linked to the data so it changes itself whenever the data changes. Ggplot2 however, is simpler to learn for beginners, has a cleaner look, and has plot types that ggvis cannot support (ex. polar coordinates). Additionally, ggplot2 also has the faceting feature which is incredibly useful in separating data visually while ggvis does not have this.

### Conclusion

Since ggvis is a data visualization tool, it application possibilities are endless. From science projects and school projects to business proposals and data exploration, ggvis can make the graphics for all these topics visually appealing, easy to understand, and relatively easy to make. It's ability to scale and layer multiple data sets as well as it's functionality with shiny and vega is what makes it unique and powerful. I look forward to learning more about it and applying it to the new datasets we encounter.

## References


- history of ggvis
- overview of ggvis
- ggplot2 vs ggvis
- more ggvis vs ggplot2
- what is vega?
- interactivity of ggvis
- data hierarchy
- layers