Post2

Making Interactive and Dynamic Visualizations

Hello! My name is Saransh Gupta, and today I will be showing you how to use ggvis for more interactive visualizations! At this point, I am sure most of you are familiar with ggplot2, which is the standard R package for visualization. Its graphs are much more visually appealing and clear than the default ones, which is why statisticians favor ggplot2 over the default R visualizations. However, these visualizations still aren't dynamic, and don't allow the user to directly interact with them. This is where ggvis comes in! Not only can you make the same visually appealing plots with ggvis as you can with ggplot, but now you can also interact with your plots and make them dynamic! Let's get started.

Understanding the Data

In this tutorial, I will be using data from https://github.com/fivethirtyeight/data. However, before we use ggvis lets install some other packages for our data explorations! Use the library command to load it. If the packages aren't already loaded, you can load them into the console with the install.package command.

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(ggvis)
```

```
##
## Attaching package: 'ggvis'
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

Now that we have the packages loaded in, we can begin our data analysis! For our data, we will use the dataset of alcohol consumption by country from the fivethirtyeight github, a famous statistics website.

```
alcohol_data = read_csv("~/Documents/GitHub/stat133-hws-fall17/post02/Data/drinks.csv")
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   beer_servings = col_integer(),
##   spirit_servings = col_integer(),
##   wine_servings = col_integer(),
##   total_litres_of_pure_alcohol = col_double()
## )
```

Great! Now that we have loaded in the data, lets first look at the structure. It is important to know our data so that we can choose appropriate information to convey to other people about the data. This is a crucial part of data preparation, and is just as important as any other part of the data lifecycle.

```
nrow(alcohol_data)
```

```
## [1] 193
```

```
head(alcohol_data)
```

```
## # A tibble: 6 x 5
##             country beer_servings spirit_servings wine_servings
##               <chr>         <int>           <int>         <int>
## 1       Afghanistan             0               0             0
## 2           Albania            89             132            54
## 3           Algeria            25               0            14
## 4           Andorra           245             138           312
## 5            Angola           217              57            45
## 6 Antigua & Barbuda           102             128            45
## # ... with 1 more variables: total_litres_of_pure_alcohol <dbl>
```

Now here we see that our data provides various metrics of alcohol comsumption, including beer, spirit, and wine servings as well as overall total alcohol intake from all of these drinks. Here we also notice that some countries have a 0 for total_litres_of_pure_alcohol consumed, so we can assume that in these cases, country data is not readily available. To prevent missing data from skewing our results, lets exclude any rows in which total_litres_of_pure_alcohol is 0. Lets see how many rows are removed from the inital 193 with this command.

```
new_alcohol_data = filter(alcohol_data, total_litres_of_pure_alcohol != 0)
nrow(new_alcohol_data)
```

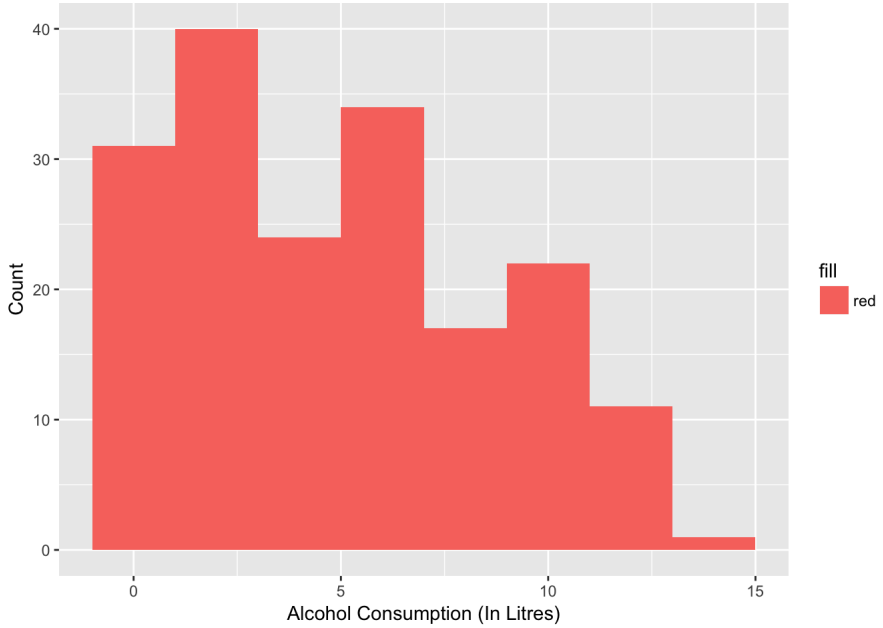
```
## [1] 180
```

Now we see that after removing all the rows in which total alcohol consumption is 0, we are left with 180 countries. This should help reduce non-information bias in our analysis. Data cleaning is a crucial part of all data analysis and exploration, and should never be forgotten!

Visualizing the Data with ggplot2 and ggvis

Now that we have sufficiently cleaned our data, let us jump into some visualizations and analysis to more clearly understand the data which we have, since 180 rows of text is not really informative. First, lets make a histogram of the total amount of alcohol consumed by country. This is very simple using ggplot.

```
ggplot(data = new_alcohol_data, aes(total_litres_of_pure_alcohol, fill = 'red'))+geom_histogram(binwidth = 2) + xl
ab("Alcohol Consumption (In Litres)") + ylab("Count")
```

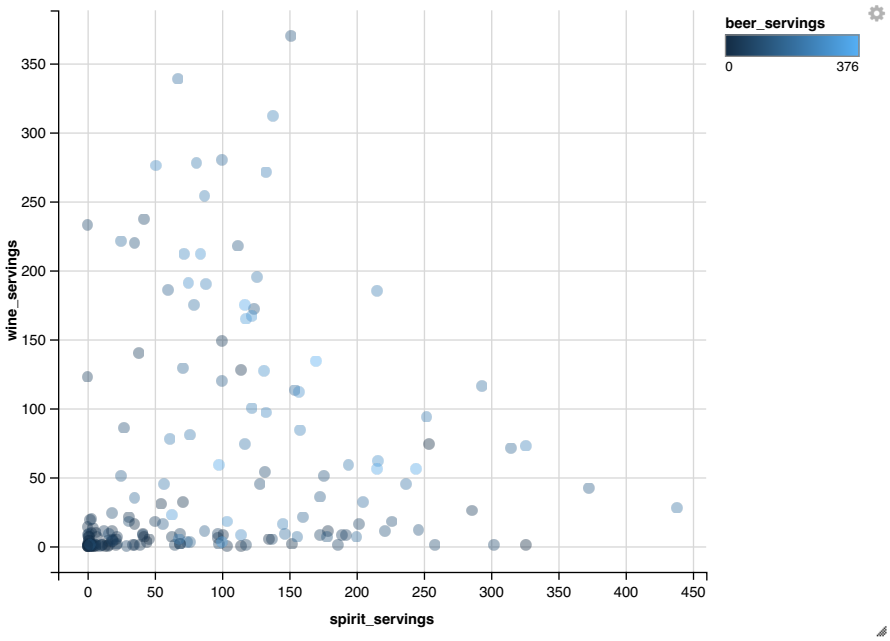


While this gives us a better picture of the overall distribution of alcohol consumption, it really doesn't help us shed insight on the particular relationships between different kinds of alcohol. In this case, a scatterplot would be much more helpful. Say for example we would like a scatterplot of spirit servings to wine servings to see whether there is a correlation between drinking more spirits to drinking more wine. Using ggplot2, we could make a visualization using the following commands.

```
ggplot(new_alcohol_data, aes(spirit_servings, wine_servings)) + geom_point() + ggtitle('Correlation between Spirit
and Wine Consumption')
```

Interesting! There actually seems to be a negative correlation between spirit consumption and wine consumption; the less spirits are consumed in a country, the more wine is consumed, and vice versa. This is good…we wouldn't want a country of alcohlics where consumption of both increased! While ggplot is good for this kind of visualization, lets say we wanted to get a better view of the data through being able to account for points which are really close to each other, namely in the bottom left. This is where ggvis comes into play! Using ggvis specific functions such as layering, we can get a better visualization of the data. We can even add functionality such as visualizing the amount of beer_servings compared to the amount of wine and spirit servings, and put it in as a third variable which can be conveyed through fill!

```
new_alcohol_data %>% ggvis(~spirit_servings, ~wine_servings, fill = ~beer_servings, opacity := 0.4) %>% layer_poin
ts()
```



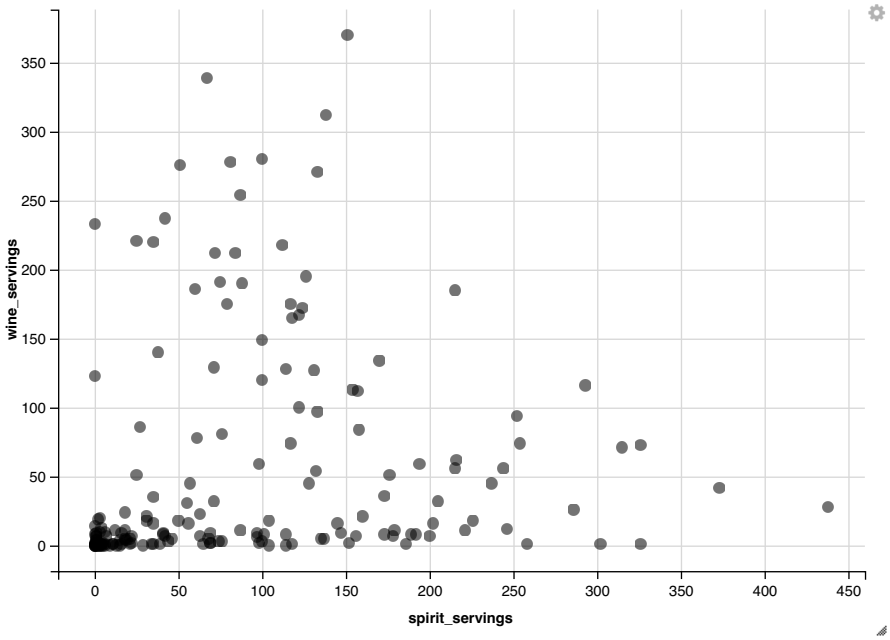
Notice how much easier it is to visualize different points, and we can clearly see where multiple points overlap, unlike in the ggplot graph. Also, another important thing to notice is the fact that ggvis unlike ggplot uses dplyr operator chaining to make its graphs. This means that ggvis can be much more flexible in making graphs on data which isn't directly provided but calculated through an intermediary function, something which is much more of a hassle in ggplot. However, ggplot still has many things you can do with it, and for a quick and easy overview you can check out this ggplot2 cheatsheet at https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf . For some extra visually aesthetic tips and tricks visit http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/.

More ggvis!

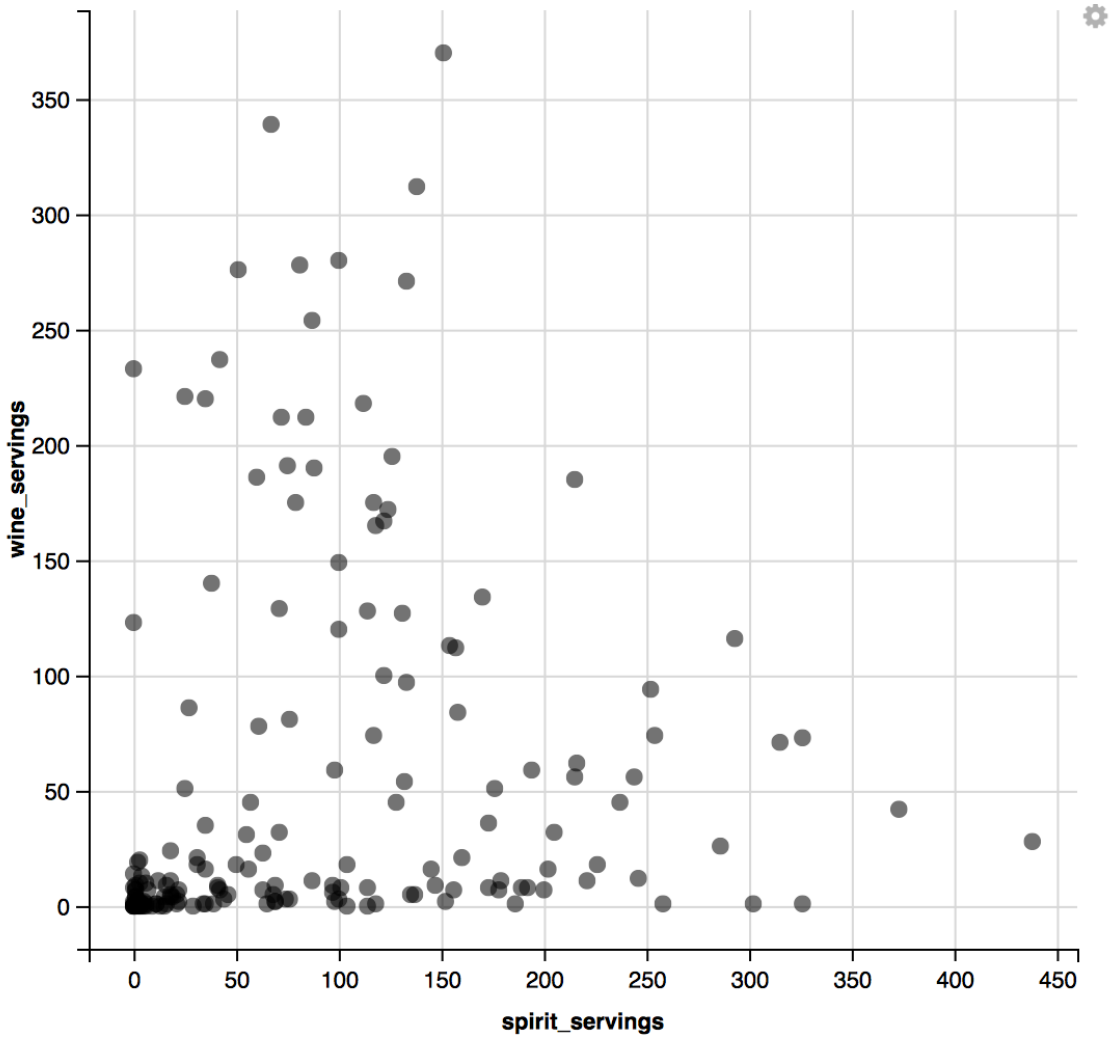
Now that we are a bit more familiar with ggvis and its format, lets see a bit more of the cool things ggvis can do! One of the coolest things which ggvis allows us to do is make interactive plots in which the users can instantly change axes values to get a better idea or visualization of the data. Here is an example of the same spirits and wine servings data but now with an interactive ggvis plot. Here, we will have a slider for both size and opacity which the user can adjust as they see fit.

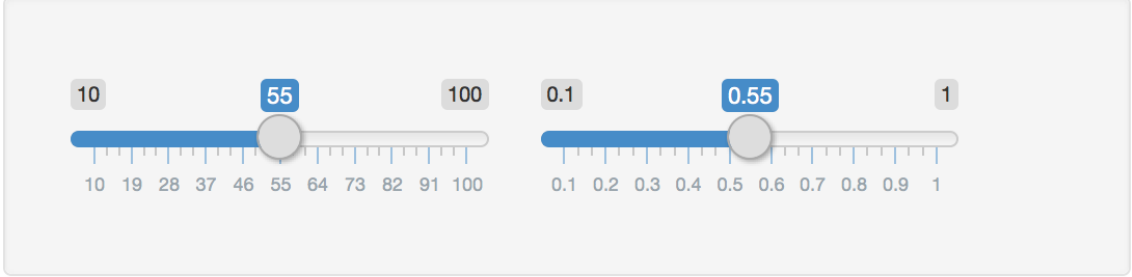
```
new_alcohol_data %>% ggvis(~spirit_servings, ~wine_servings, size := input_slider(10, 100), opacity := input_slide
r(.1, 1)) %>% layer_points()
```

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



Since this is loaded in a static HTML format, you won't be able to interact with the data, but here is what it would look like normally:

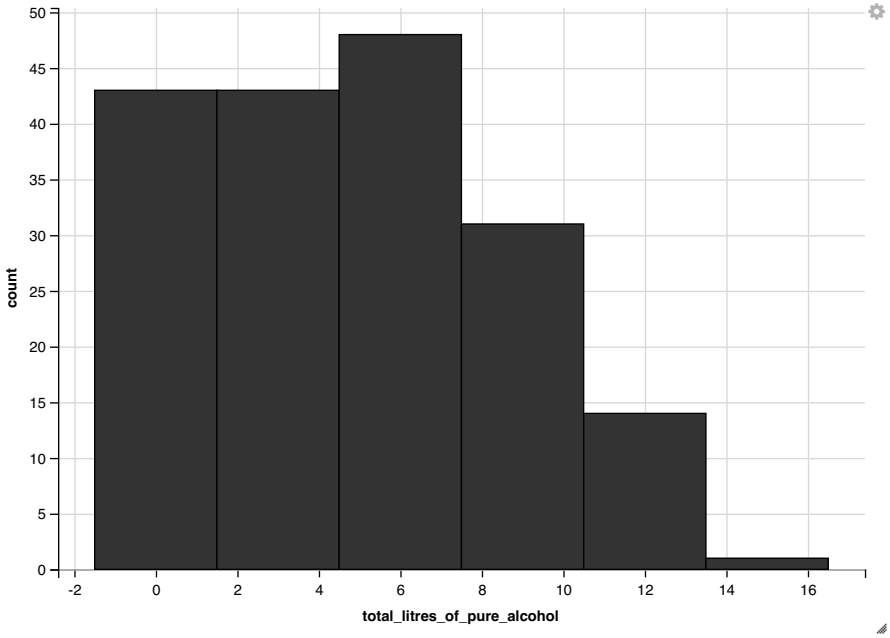




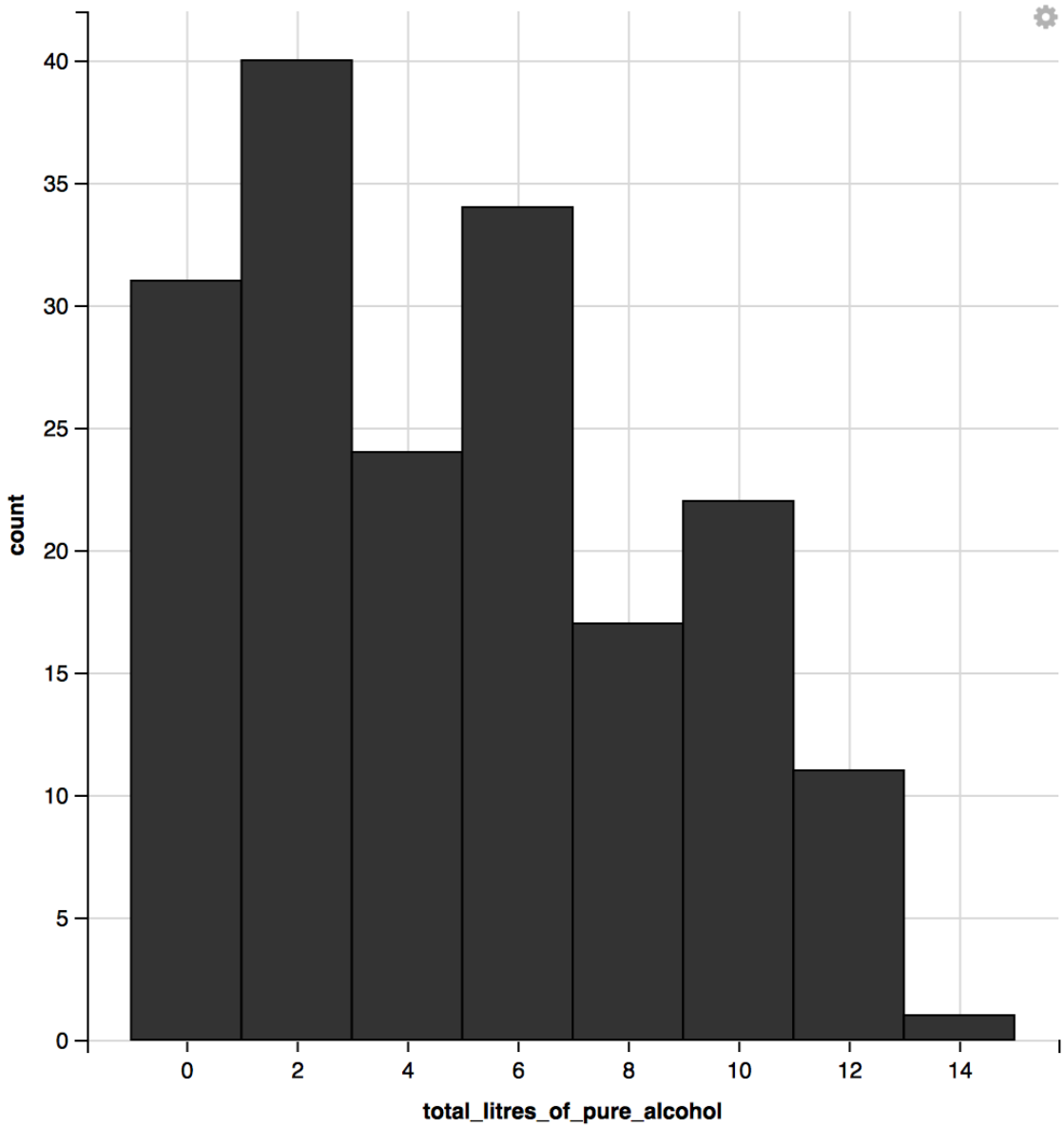
We can also do the same for histograms as well to make viewing the graph more dynamic! Here, there is a slider for the user to change the bin width, and see how this affects what the data looks like.

```
new_alcohol_data %>% ggvis(~total_litres_of_pure_alcohol) %>% layer_histograms( width = input_slider(1, 5, step =
1, label = 'bin width'))
```

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



Again, here is a screenshot of how this graph would look like if it were normally loaded in with ggvis in a dynamic format:



Both of these are examples of how ggvis can give you a more interactive and fun module beyond the static plots of ggplot2. While these may not be visible in static HTML formats because they use shiny behind the scenes, these interactive graphs will really help a user get a better understanding of the data, and are also much more fun to play around with than just your static graphs! Try it out for yourself!

Putting it All Together

All in all, ggvis is a great way to spice up your visualizations and perform more sophisticated analysis on the data than one would be able to simply with ggplot2. Data visualization is a crucial part in allowing your audience to understand the data, and making data visualization interactive takes this 1 step further, and allows the audience to be hands-on with the data, something which is much more effective at conveying stories in data than anything else. Ultimately, when computing with data, we want to be able to easily tell a story with the bytes of information we have, converting 0's and 1's to colorful and easy to understand visualizations, which is why maximizing the quality and interactivenemss of these visualizations is crucial to help an audience fully understand the data which they are working with.

References:

1. https://github.com/fivethirtyeight/data
2. https://www.statmethods.net
3. http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3
4. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
5. http://dplyr.tidyverse.org/reference
6. https://ggvis.rstudio.com/ggvis-basics.html

7. https://ggvis.rstudio.com/ggplot2.html
8. https://www.r-bloggers.com/how-to-create-interactive-data-visualizations-with-ggvis/