Getting familiar with ggvis - interactive features

Dongseong Seo, 24293621

November 21, 2017

Introduction - motivation

Approaching to the end of the semester, it was very glad to meet the package `ggvis`. The term *interactive graphics* was the thing which instantly grabbed my attention.

However, with limited time available, we could not have enough time to learn about it. Although some materials related to `ggvis` were asked in the **HW4**, I think it was not enough for us to get familiar with `ggvis`. In short, the time allocation was not fair in comparison with its value.

This is why I have chosen `ggvis`, especially its **interactive features**, as my topic so that we can have additional time to understand `ggvis`.

Overall content

1. This post would not mainly focus on very basic structures of `ggvis` such as %>% and layers, `layer_points`.
2. Rather, this post is going to focus on **interactive features** of `ggvis`.
3. This post includes some practical tips, for beginners like me, to maintain errors in `ggvis`.

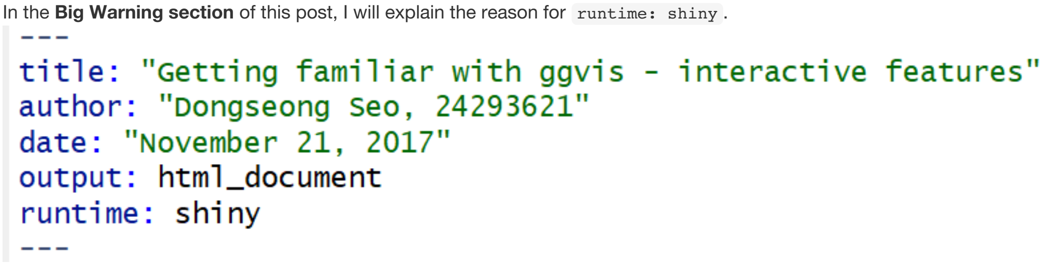
Version of tools

Before going in to `ggvis`, here are details about version of tools that you would need to replicate this post.
1. R version 3.4.2 (2017-09-28) – "Short Summer"
2. RStudio version 1.0.153
3. `ggvis` version 0.4.3

How to replicate

In addition to copying and pasting codes, there are **two** additional things which will facilitate your replicating process.

1. It would be very appreciated, if you put `runtime: shiny` command in your Rmd header. By this, it will be able to see some interactive features in this post.
   - In the **Big Warning section** of this post, I will explain the reason for `runtime: shiny`.

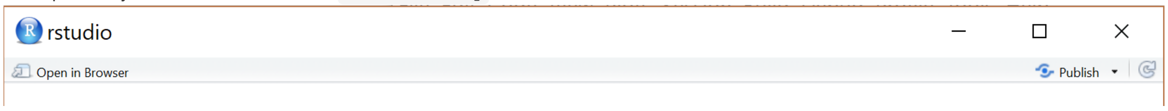


```
---
title: "Getting familiar with ggvis - interactive features"
author: "Dongseong Seo, 24293621"
date: "November 21, 2017"
output: html_document
runtime: shiny
---
```

2. With instability of `ggvis`, when we knit our html with the `runtime: shiny` command, there seems to be no outputs like this image.

   This error frequently occurs, especially, when we copy and paste codes to replicate plots. However, fortunately, we can still see the bottom right interactive part which allows us to resize our plots. This one.

Therefore, when you try to resize the blank outputs with this interactive arrow, the plots suddenly come out. However, it would be frustrating if we have to do this for every plot. Therefore, it would also be **appreciated**, if you click this "Open in Browser" icon, located at the top left of your knitted html with the `runtime: shiny` command.



By this, every plot will be there for you!

Preparation

Let's load the required package, `ggvis`.

```
# If you have not installed ggvis
# install.packages("ggvis")
library(ggvis)    # version 0.4.3
```

We are going to use the data set, `USArrests`, already come in R.

```
# one of the data sets already come in R
head(USArrests, n = 3)
```

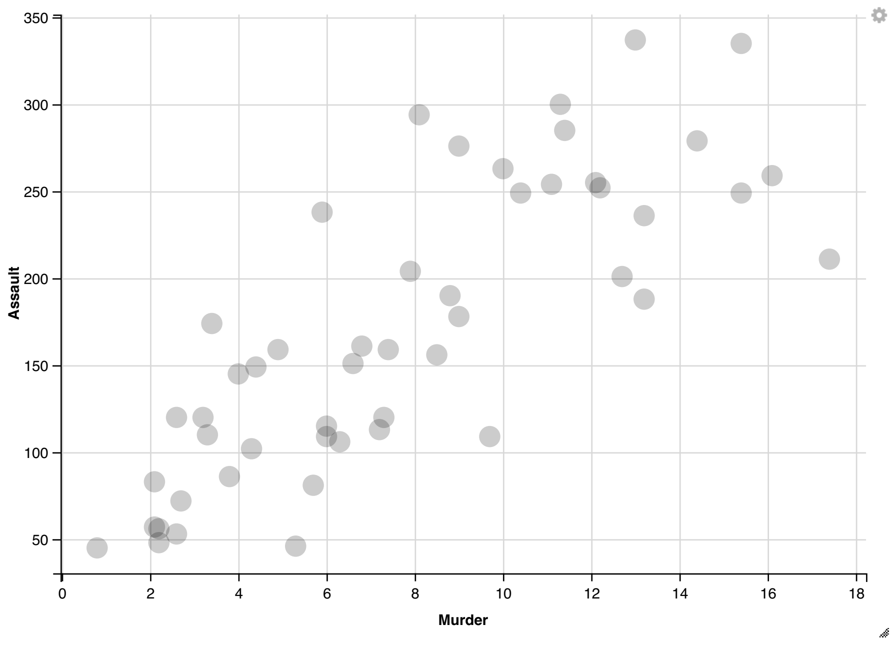
```
##         Murder Assault UrbanPop Rape
## Alabama   13.2     236       58 21.2
## Alaska    10.0     263       48 44.5
## Arizona    8.1     294       80 31.0
```

Interactivity

hover

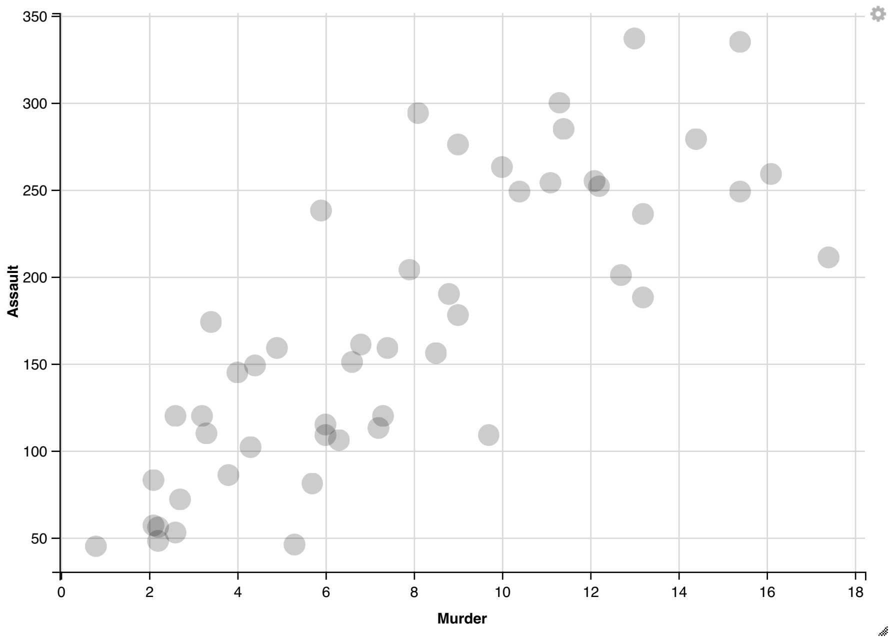
The first interesting interactivity which I am going to introduce is `hover`.
Let's plot one example to see what `hover` is.

```
USArrests %>%   # let's set x-axis = Murder and y-axis = Assault
  ggvis(x = ~Murder, y = ~Assault) %>%   # we are going to use fillOpacity.hover
  # The argument size:= is about the point size (200)
  # I will explain about fillOpacity and fillOpacity.hover below
  layer_points(size := 200, fillOpacity := 0.2, fillOpacity.hover := 1)
```



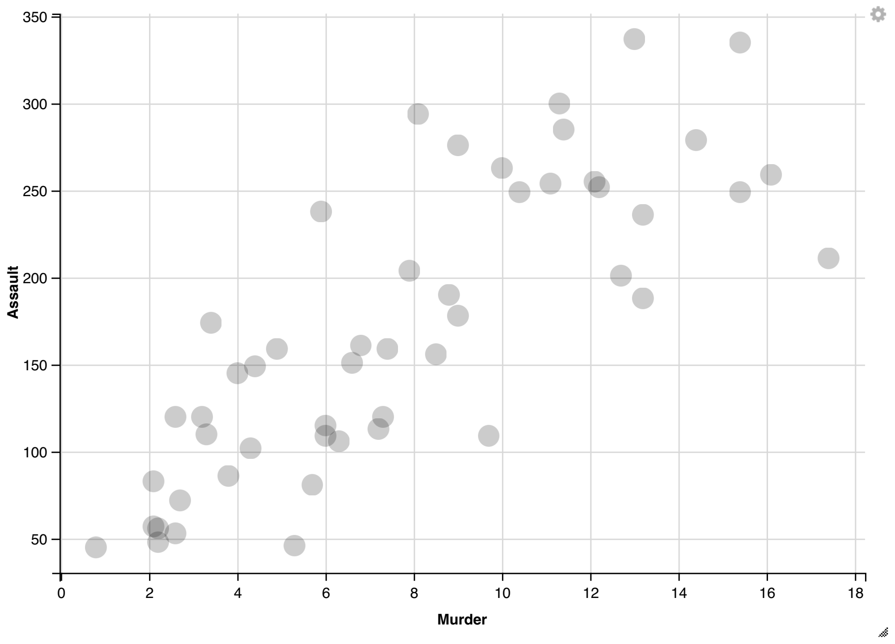
Could you notice what `hover` does?
Yes, when you put your mouse cursor over a point on the above plot, the point's opacity is changed.
The value of `fillOpacity:=` decides the initial opacity of a point. And, the value of `fillOpacity.hover:=` decides the later opacity of the point after you put your pointer over it.
The interesting thing is you could just use `opacity:=` and `opacity.hover:=` to get the same plot. The thing you have to be careful is that the letter `o` in the `opacity` argument is a small letter not a capital letter `O` as in the `fillOpacity` argument.
Let's try!

```
USArrests %>%
  ggvis(x = ~Murder, y = ~Assault) %>%
  # Let's use opacity and opacity.hover instead.
  layer_points(size := 200, opacity := 0.2, opacity.hover := 1)
```



As we have expected, they give the same plot.
What about the point's size? Can we use `hover` to interact with its size?
Let's try it! We will keep our `opacity.hover`, and, we will add `size.hover`

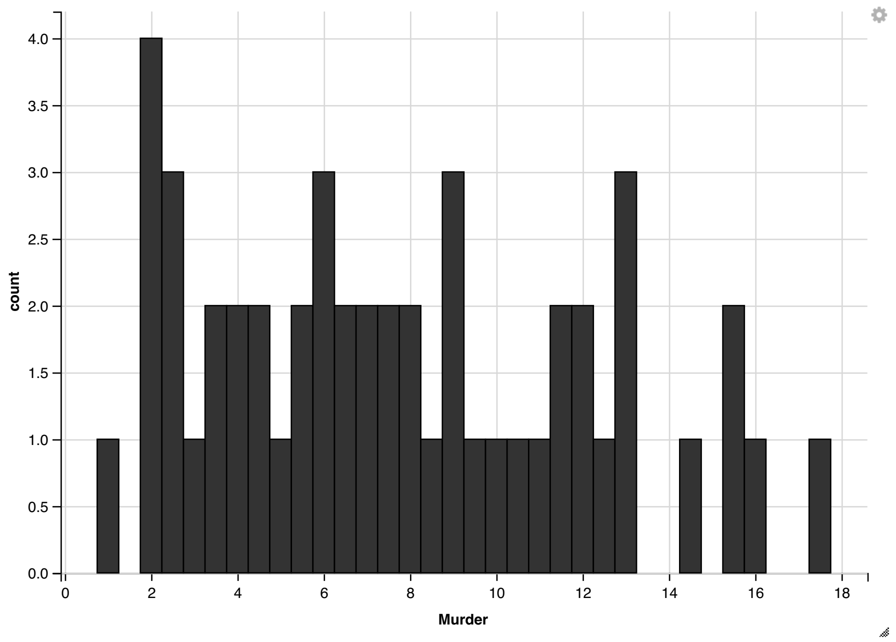
```
USArrests %>%
  ggvis(x = ~Murder, y = ~Assault) %>%
  # Let's add size.hover := 2000
  layer_points(size := 200, size.hover := 2000,
               opacity := 0.2, opacity.hover := 1)
```



What a dramatic change!
I have been curious what we can do more with the `hover`. And, I have fortunately found this link which contains interesting demos of hover.
According to the above link, we can play with more things such as `stroke`, `strokeWidth`, and `fill`.
Let's play one of them.

```
USArrests %>%   # This time let's draw a histogram
  ggvis(x = ~Murder) %>%     # Our x-axis is Murder
  # Let's play with strokeWidth.hover
  layer_histograms(strokeWidth := 1, strokeWidth.hover := 20)
```

```
## Guessing width = 0.5 # range / 34
```



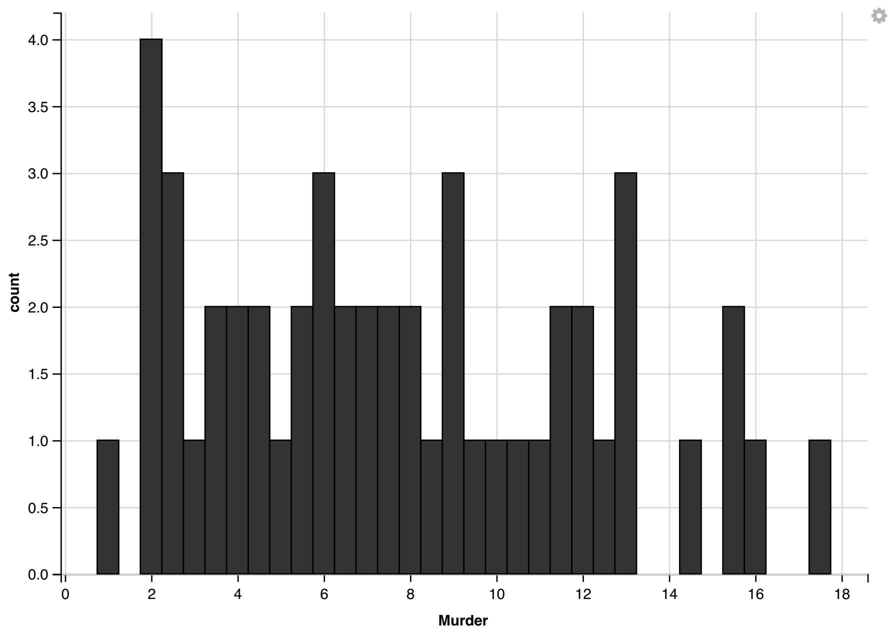
It's not that pretty, but we could see what `strokeWidth.hover` does clearly. It lets us interact with each bin's strokewidth of our histogram.
Let's move on to the next section, a sub-section of `hover`.

set_options

By adding a set_options *layer* to our `ggvis`, we could manipulate some options. For example, we could set the size of our plot or we could make a decision whether an user could resize our plot or not.
Let's try the permission part.

```
USArrests %>%   # The same histogram as the above one
  ggvis(x = ~Murder) %>%
  layer_histograms(strokeWidth := 1, strokeWidth.hover := 20) %>%
  # Let's add set_options layer with the argument resizable
  # The thing you would notice is, in the set_options we use = not :=
  set_options(resizable = FALSE)
```

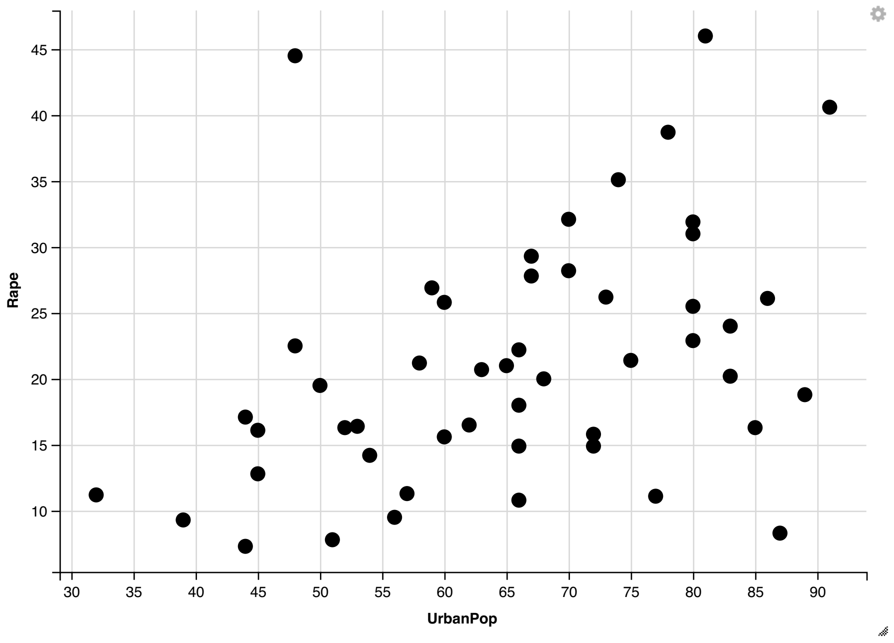
```
## Guessing width = 0.5 # range / 34
```



As we can see, this plot is just the same as above except one thing.
The bottom right interactive part for one to resize the plot is gone now. It's interesting, but, personally, removing an interactive part from `ggvis` would not be a good idea. It is like we are polluting its values.

Anyway, there is a more interesting reason why I chose the `set_options` as the sub-section of `hover`.
As we are focusing on interactivity of `ggvis`, this one argument - `hover_duration`, of set_options would be very useful in terms of interactivity.

```
USArrests %>%   # Let's go back to a scatter plot but with different axes
  ggvis(x = ~UrbanPop, y = ~Rape) %>%   # x-axis is UrbanPop and y-axis is Rape
  # Let's use size.hover to see what hover_duration does
  layer_points(size := 100, size.hover := 700) %>%
  # Let hover_duration = 1000
  set_options(hover_duration = 1000)
```



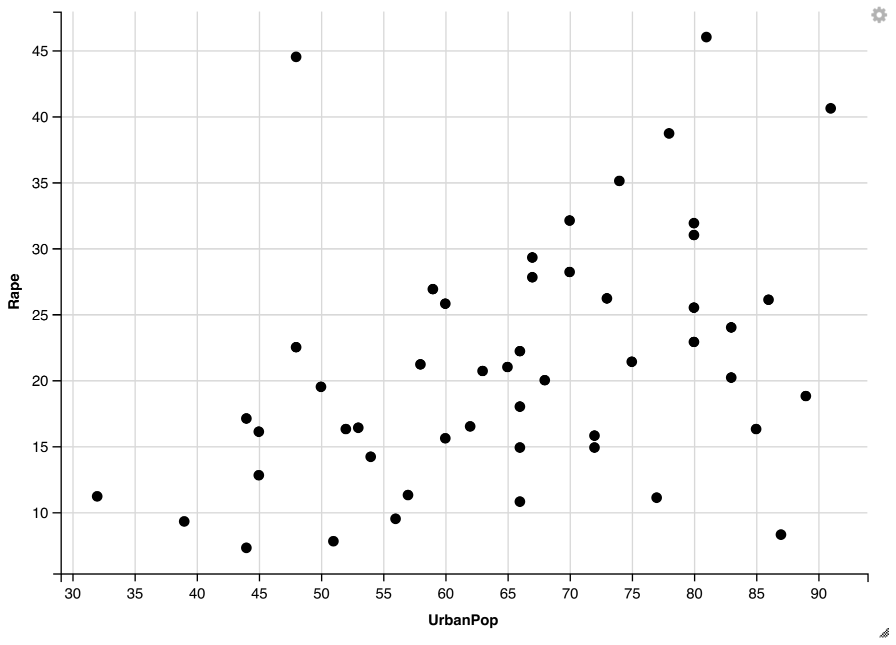
When you move your mouse cursor over a point, you would notice that the point doesn't change its size immediately like previous ones. It takes time for it to transform to a bigger one, and it also takes time to get back its original size.
Yes, `hover_duration` in the `set_options` manipulates hover transformation time. The unit of time `hover_duration` takes is milliseconds. One tip is that "don't underestimate milliseconds." I once put very large number for the `hover_duration` argument, and it took forever to see **one** complete transformation.
If you move around points, you could notice that transformation of one point can occur concurrently with other transformation. By setting reasonable time, one would be able to enhance interactive features of `ggvis`.
The information about `set_options` including `hover_duration` can be found in this link.

Big Warning

Sadly, from now on, following interactive features of `ggvis` are not going to work in our knitted html. In other words, you would not be able to see them in this knitted html.
To make this clear, let me show you what it means.

```
# I will explain what add_tooltip does very soon
# This example is for showing what 'interactive features doesn't work' means
USArrests %>%
  ggvis(x = ~UrbanPop, y = ~Rape) %>%
  layer_points() %>%
  add_tooltip(function (x) x$Rape, on = "hover")
```

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



Can you see this message just above our plot?

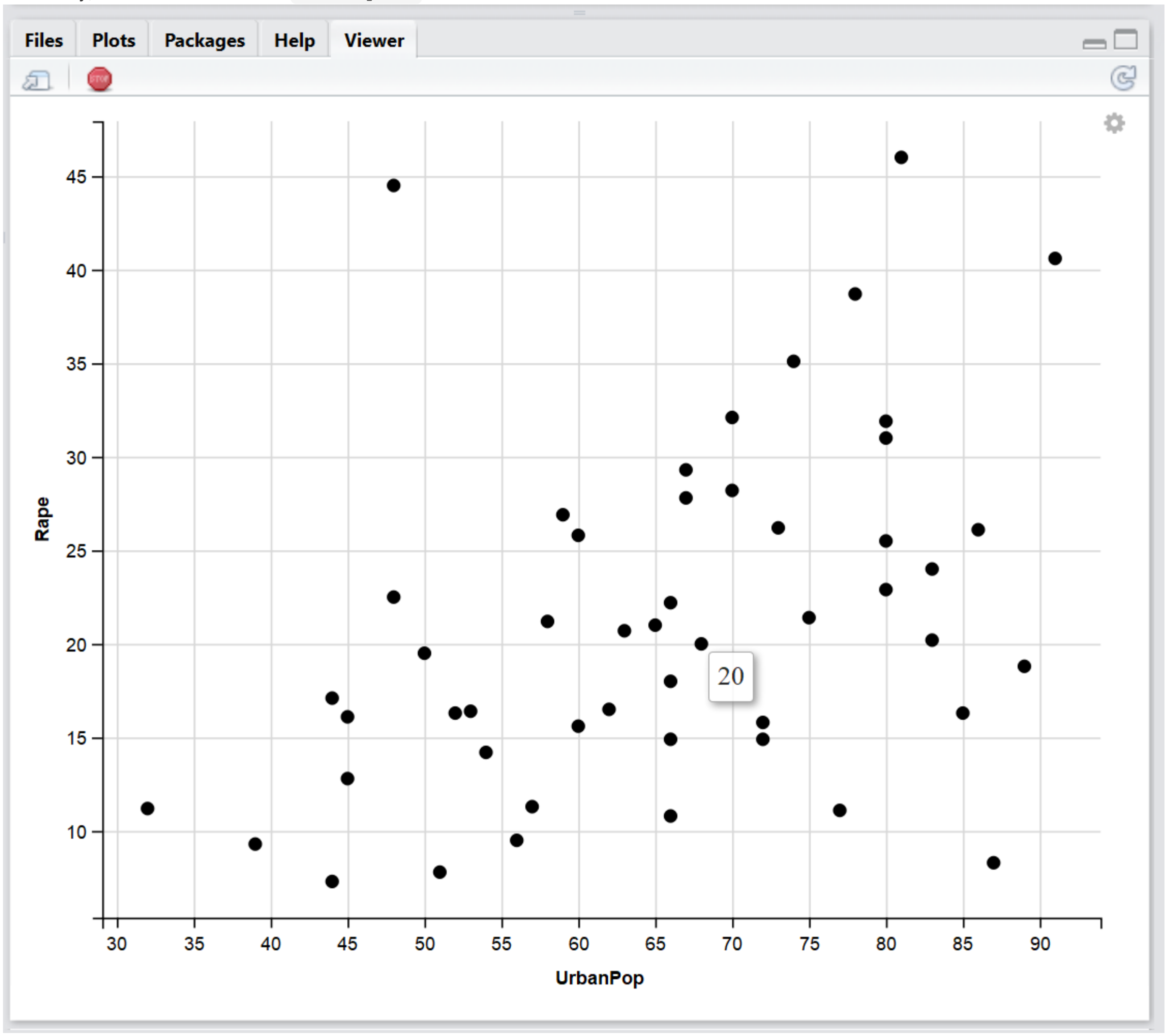
> ## Warning: Can't output dynamic/interactive ggvis plots in a knitr document.
> ## Generating a static (non-dynamic, non-interactive) version of the plot.

Our knitr html document just returns a *static* version of our plot. In other words, it cannot output interactive features.

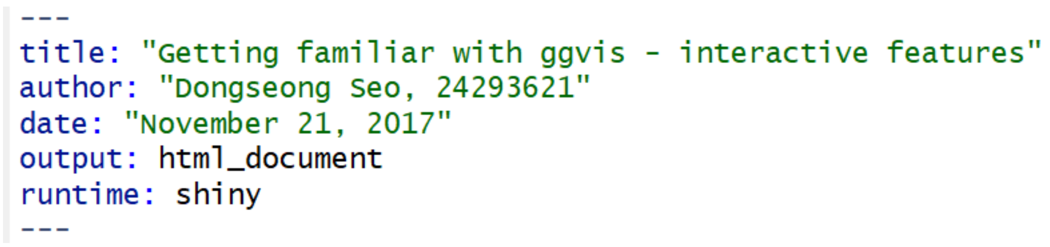
Solutions

In order to view how our intended interactive features work, there are basically three things we can do.

1. Thankfully, we can notice that our `viewer panel` in RStudio shows what interactive features do.



2. The simplest and best way, in my opinion, is putting `runtime: shiny` in the **header** of Rmd. I found this valuable solution in this link.



```
---
title: "Getting familiar with ggvis - interactive features"
author: "Dongseong Seo, 24293621"
date: "November 21, 2017"
output: html_document
runtime: shiny
---
```

However, the one thing you have to be careful is that if you have put `runtime: shiny` in the header of your Rmd, knitted documents, such as `md` or `html` will not be created. If you want to create knitted documents, you should remove `runtime: shiny` from your header.

3. The possibly most time consuming way is using `Shiny` to check. If your purpose is just checking how your code works in `ggvis`, this way would not be your choice. However, if your purpose is sharing or reporting your research with plots using `ggivis`, it would be your choice. In short, `ggvis` is intended to render well with `Shiny`.

In order to communicate with you

Due to the above explained knitting problem, it would be very **appreciated** if you kindly put `runtime: shiny` in your header of Rmd. This magic command would be helpful for you to enjoy the rest parts of this post.

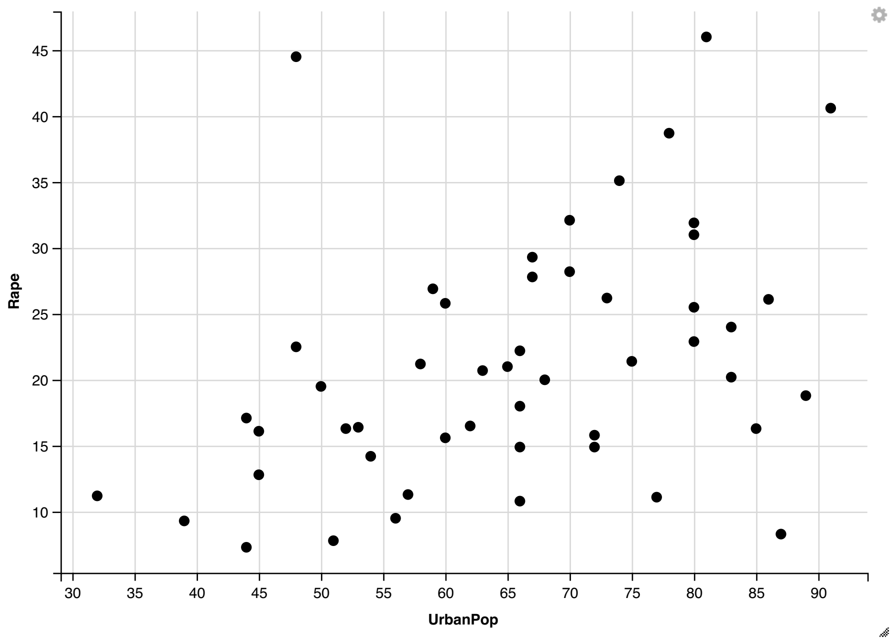
Also, owing to the same reason, allow me to use **images** to explain rest interactive features of `ggvis`.
Please ignore the same warning message in this knitted html document as they will be gone by the runtime command.

Tooltips

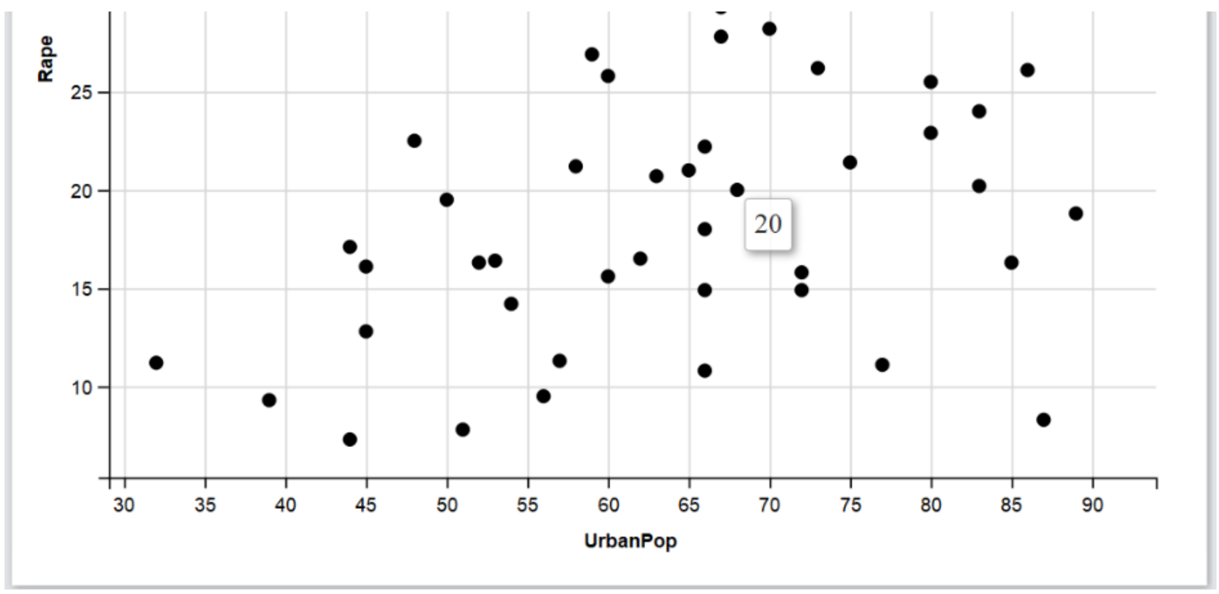
As I have mentioned above, now is the time to see what `add_tooltip` does to enhance interaction. Likely to `set_options`, we can just add one more layer `add_tooltip`. Also, we use `=` not `:=` for its arguments.
Let's plot a scatterplot.

```
USArrests %>%
  ggvis(x = ~UrbanPop, y = ~Rape) %>%   # Same axes
  layer_points() %>%
# Here we have let on = "hover"
# The function(x) is an auxiliary function likeley to the one in apply() families
  add_tooltip(function (x) x$Rape, on = "hover")
```

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



First of all, by using the `function(x) x$Rape`, we have set an outcome (a list). It means that we just have made an interactive feature which will return a list of values. In our case the list contains a corresponding single **Rape value** of a point.

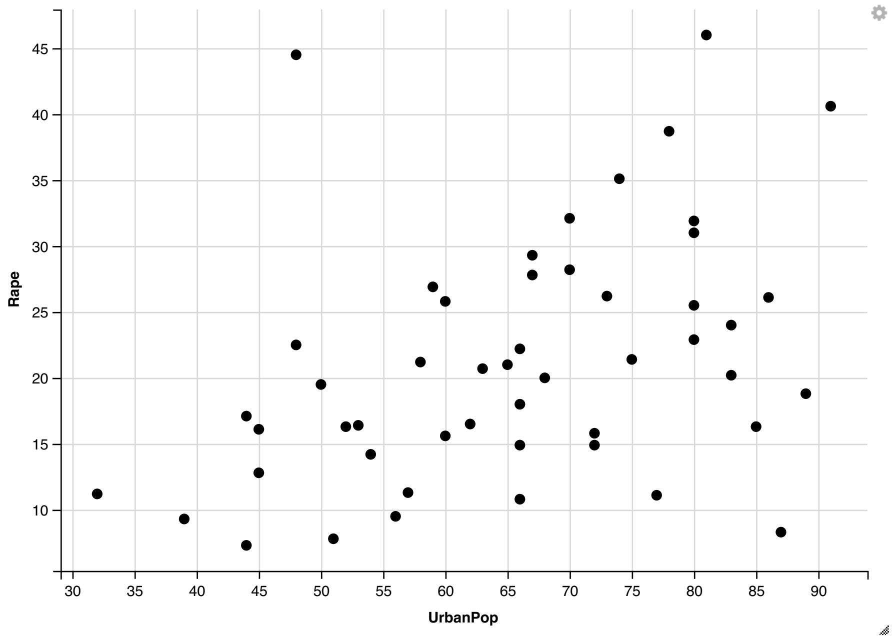


As you can see from this image, the point's y-axis(Rape) has a value, 20. And, this value is returned.

What about the `on` argument? The `on` argument is about how the interaction starts. As we have let `on = "hover"`, whenever mouse pointer moves over a point, the corresponding Rape value will pop up. The `on` argument can also take `"click"`. If we let `"click"` as its argument, the corresponding value will not pop up until we actually click the point.
Let's draw one more scatter plot with `"click"`.

```
USArrests %>%
  ggvis(x = ~UrbanPop, y = ~Rape) %>%   # same axes
  layer_points() %>%
  # Here we have let on = "click"
  # You can use any variable name istead of x for the function part
  # Also, you can put x-axis variable(UrbanPop) instead of y-axis(Rape)
  add_tooltip(function (hellothere) hellothere$UrbanPop, on = "click")
```

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



This time the corresponding **UrbanPop value** doesn't show until we actually click the point.

You can find the basic structure of `add_tooltip` in this link.

Rather than the function part, I have focused on the argument `on` of `add_tooltip` as I want to show you available interactive options of `ggvis`.
Therefore, if you think you are competent and want to do more, this link would be interesting.

Interactive control

The last part of this post is going to cover interactive input functions which do things similar to widgets in `Shiny`.
According to the ggvis .rstudio, there are 7 available interactive controls.

1. `input_slider` : A slider
2. `input_checkbox` : A check box

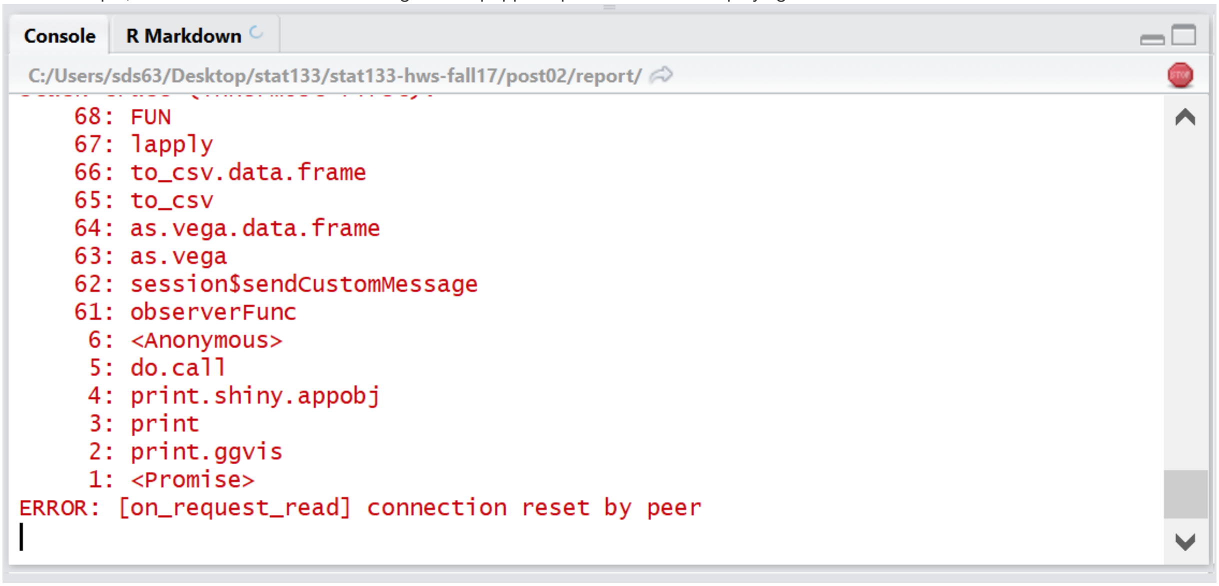
3. `input_checkboxgroup()` : A group of check boxes
4. `input_numeric()` : A spin box
5. `input_radiobuttons()` : A set of radiobuttons
6. `input_select()` : A drop-down text box
7. `input_text()` : A text input

They look and work very similar to `Shiny Widgets`. Also, the good news is that, coding of `ggvis` interactive controls is logically similar and is easier than that of `Shiny`.

However, before giving you examples of interactive controls, here is one important thing that we need to keep in mind.

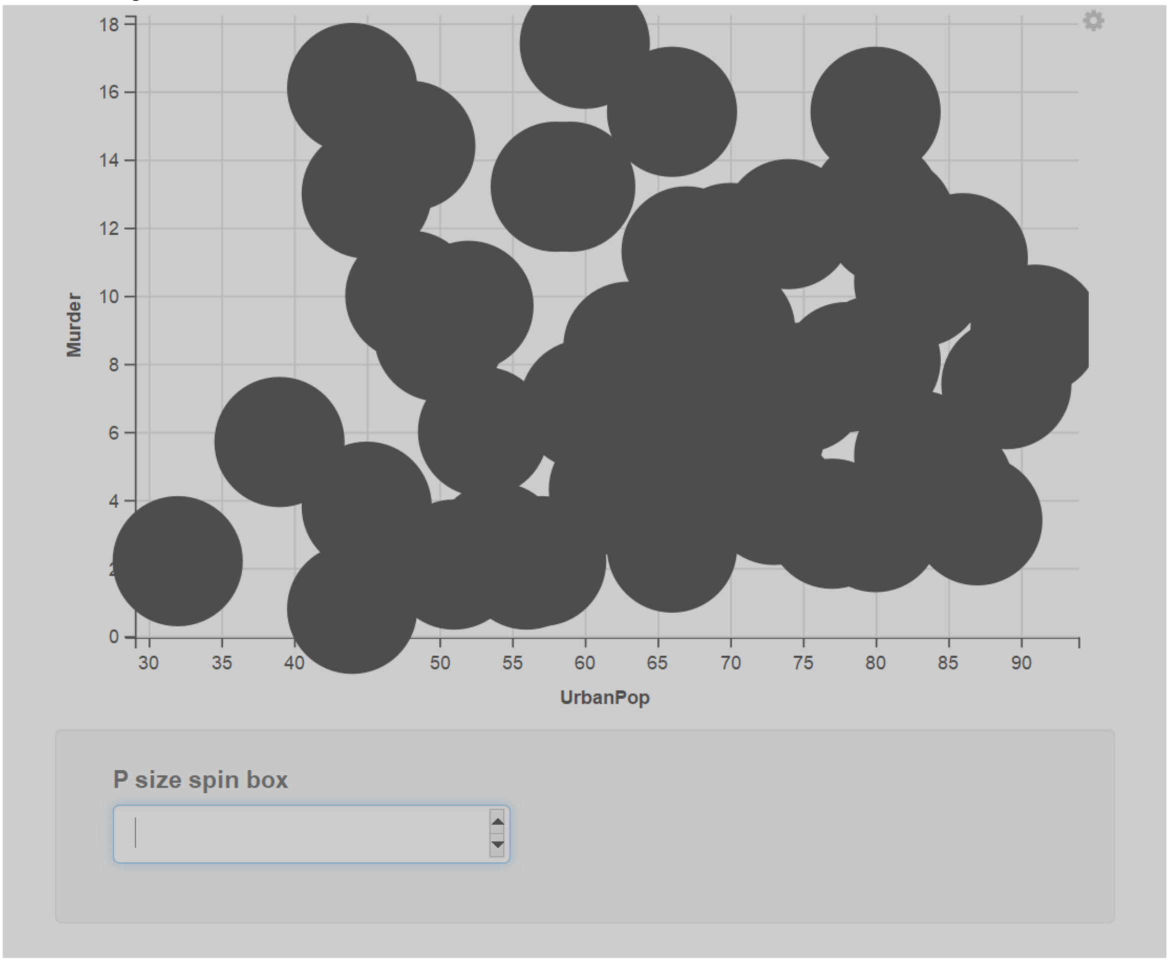
Not Stable!

As `ggvis` is an evolving package, I have noticed that many features are not stable yet. In other words, you are going to confront several errors which suddenly shut your plots.
As an example, this is the common error message which popped up while I have been playing with interactive controls.


Console
R Markdown
C:/Users/sds63/Desktop/stat133/stat133-hws-fall17/post02/report/

```
    68: FUN
    67: lapply
    66: to_csv.data.frame
    65: to_csv
    64: as.vega.data.frame
    63: as.vega
    62: session$sendCustomMessage
    61: observerFunc
     6: <Anonymous>
     5: do.call
     4: print.shiny.appobj
     3: print
     2: print.ggvis
     1: <Promise>
ERROR: [on_request_read] connection reset by peer
```

and its related image of error.



For this particular error, it seems that `ggvis` interactive controls do not know how to deal with the blank value. When we let the interactive control value as blank, we can see the plot suddenly turns to grey and refuses to work anymore.

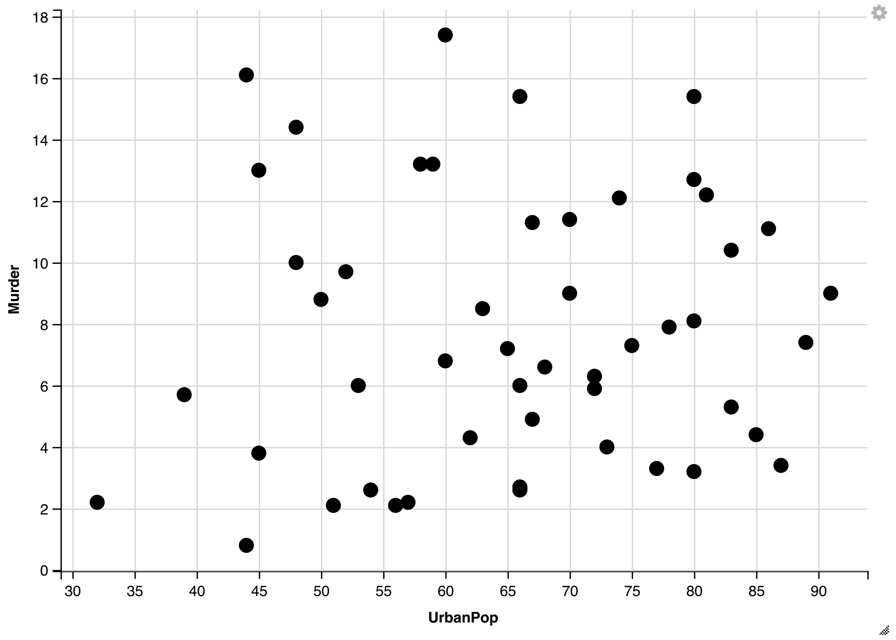
Therefore, while you are playing with interactive controls, **please be careful with the blank value!!**

Examples- Interactive Controls

With the above instability of `ggvis` in our minds, let's try one interactive control `input_numeric()`, a spin box.

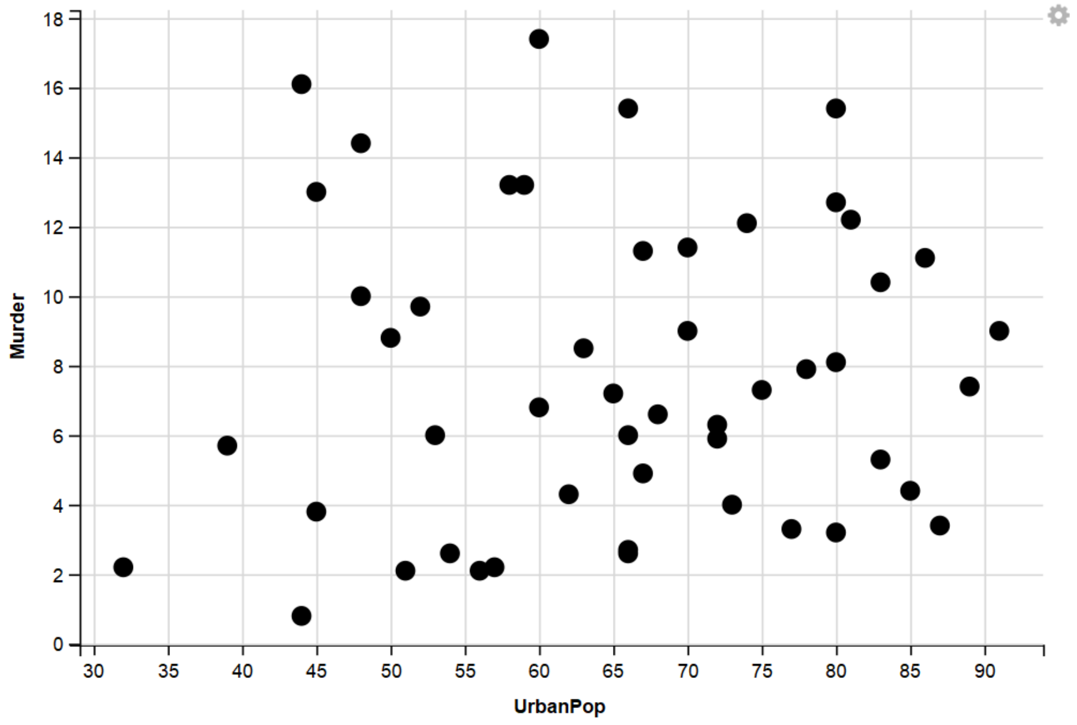
```
USArrests %>%   # This time our x-axis is UrbanPop; y-axis is Murder
  ggvis(x = ~UrbanPop, y = ~Murder) %>%
  # We are making a spin box to interact with the point size.
  # Let the point size argument, size := input_numeric()
  # 100 is the initial point size value
  # Label argument is the title of your interactive control, a spin box
  layer_points(
    size := input_numeric(100, label = "P size spin box")
  )
```

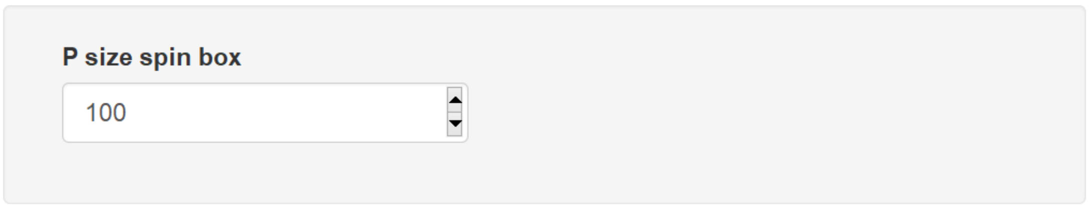
```
## Warning: Can't output dynamic/interactive ggvis plots in a knitr document.
## Generating a static (non-dynamic, non-interactive) version of the plot.
```



The codes to make the spin box controlling the point size seems quite easier than that of `Shiny` as we do not have to worry about panels, server, UI, and so on. In general, what we have to do is just letting things, which we want to interact, equal to one of interactive control functions.

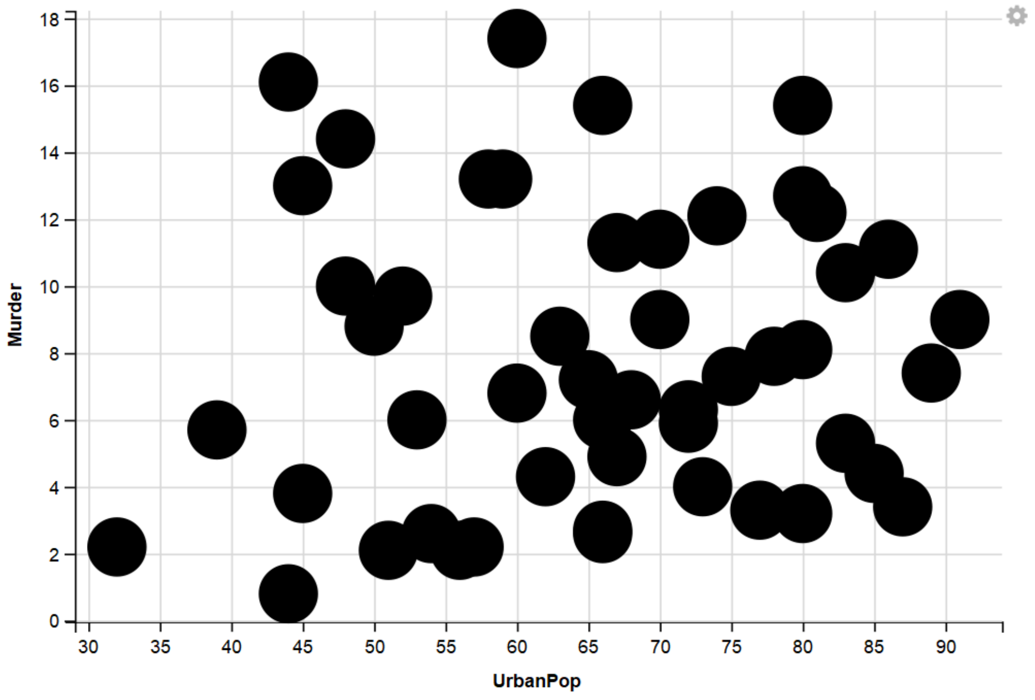
By the above code, you will get this plot with the spin box which allows us to control the point size.




P size spin box

100

As we have seen a similar widget in `Shiny`, one can change points' size by typing or interacting with up and down arrows. Let's make the point size as **955**. You would get this by letting the "P size spin box" equal to 955.



P size spin box

955

It's pretty interesting to make a plot with interactive features without using `Shiny`!

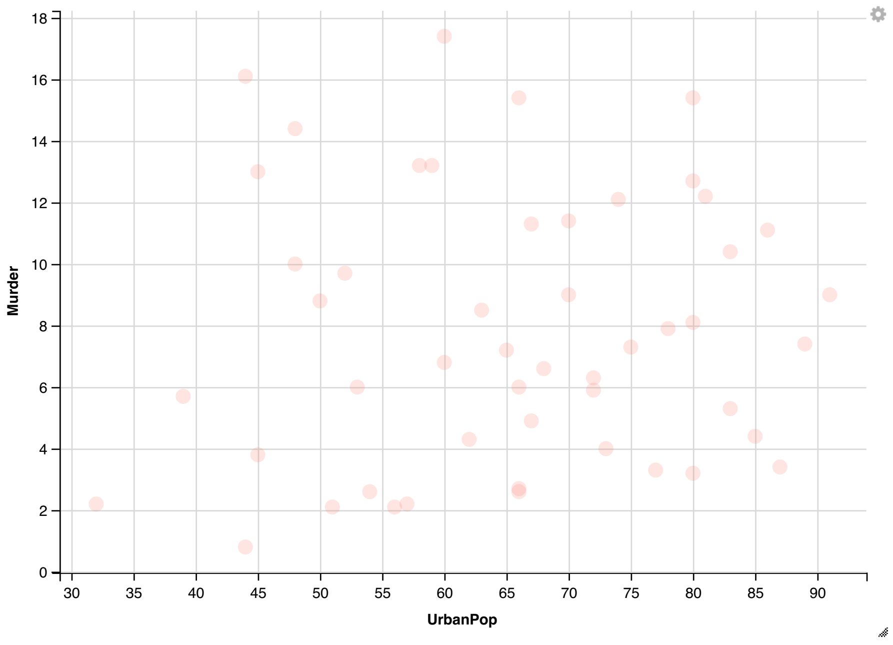
This time, let's add three more interactive controls (in total four interactive controls along with our original spin box).

- We will add `input_select()` for size.hover
  - selectable values will be 500, 1000, 2500, and 5000

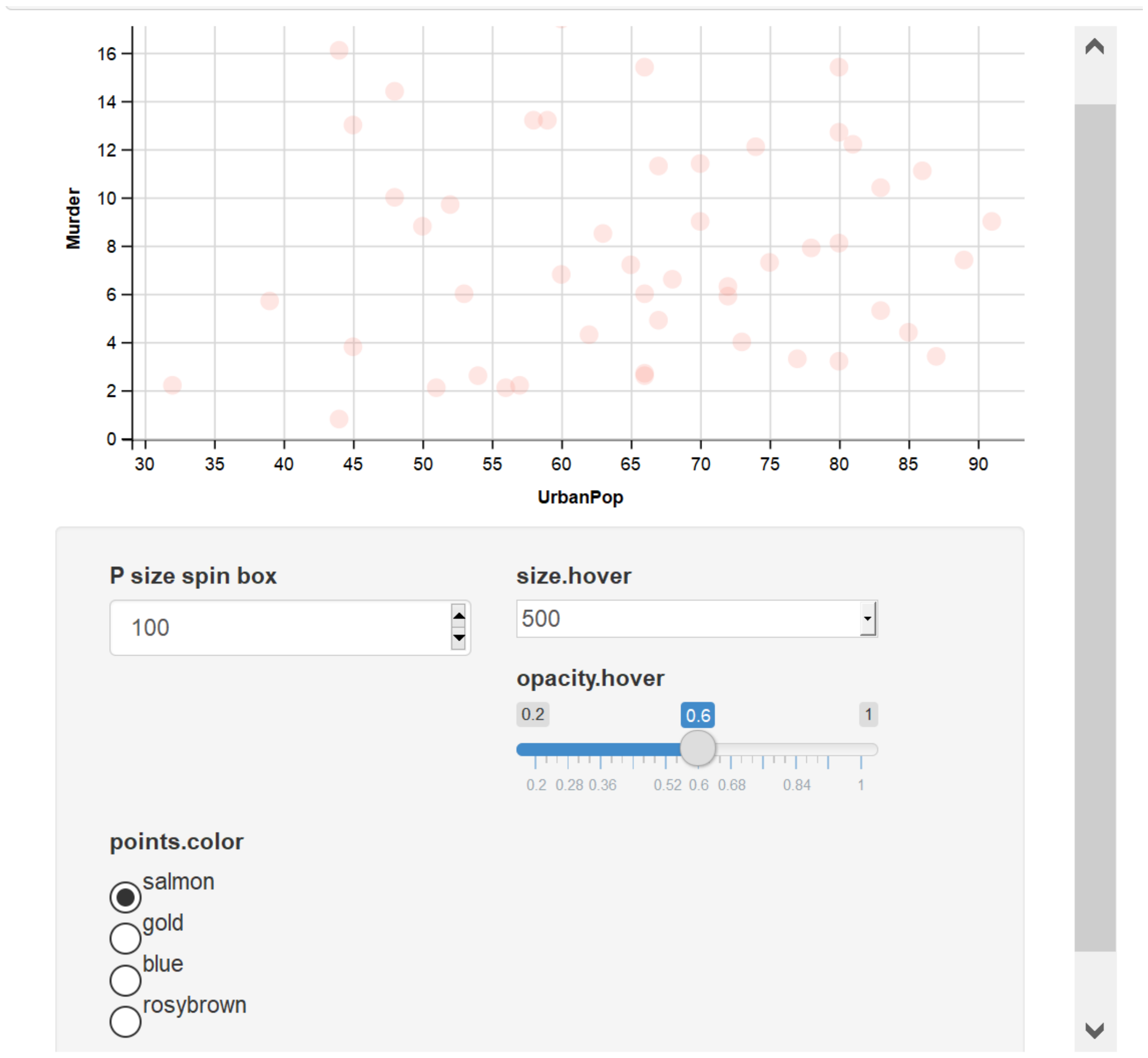
- We will add `input_slider` for fillOpacity.hover
    - we will let the initial fillOpacity as 0.2
    - and, one can interact fillOpacity.hover value from 0.2 to 1
- We will add `input_radiobuttons()` for points color
    - choices will be “salmon”, “gold”, “blue”, and “rosybrown”

```
USArrests %>%   # Our x-axis is UrbanPop; y-axis is Murder
  ggvis(x = ~UrbanPop, y = ~Murder) %>%
  layer_points(
# Let's keep our spin box
    size := input_numeric(100, label = "P size spin box"),
# input_select for size.hover, the argument choices take selectable values
# Let's name it, with the argument label, as "size.hover"
    size.hover := input_select(choices = c(500, 1000, 2500, 5000),
                               label = "size.hover"),
# Let the initial fillOpcaity value as 0.2
    fillOpacity := 0.2,
# input_slider for fillOpacity.hover, min is the minimum possible value
# max is the maximum and value takes the initially displayed hover value
# Let's name it, with the argument label, as "opacity.hover"
    fillOpacity.hover := input_slider(min = 0.2, max = 1, value = 0.6,
                                      label = "opacity.hover"),
# input_radiobuttons for points color, we can use c() for selectable values
# Let's name it, with the argument label, as "points.color"
    fill := input_radiobuttons(c("salmon", "gold", "blue", "rosybrown"),
                               label = "points.color")
  )
```

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



We have got this beautiful plot with four interactive controls.



P size spin box

100

size.hover

500

opacity.hover

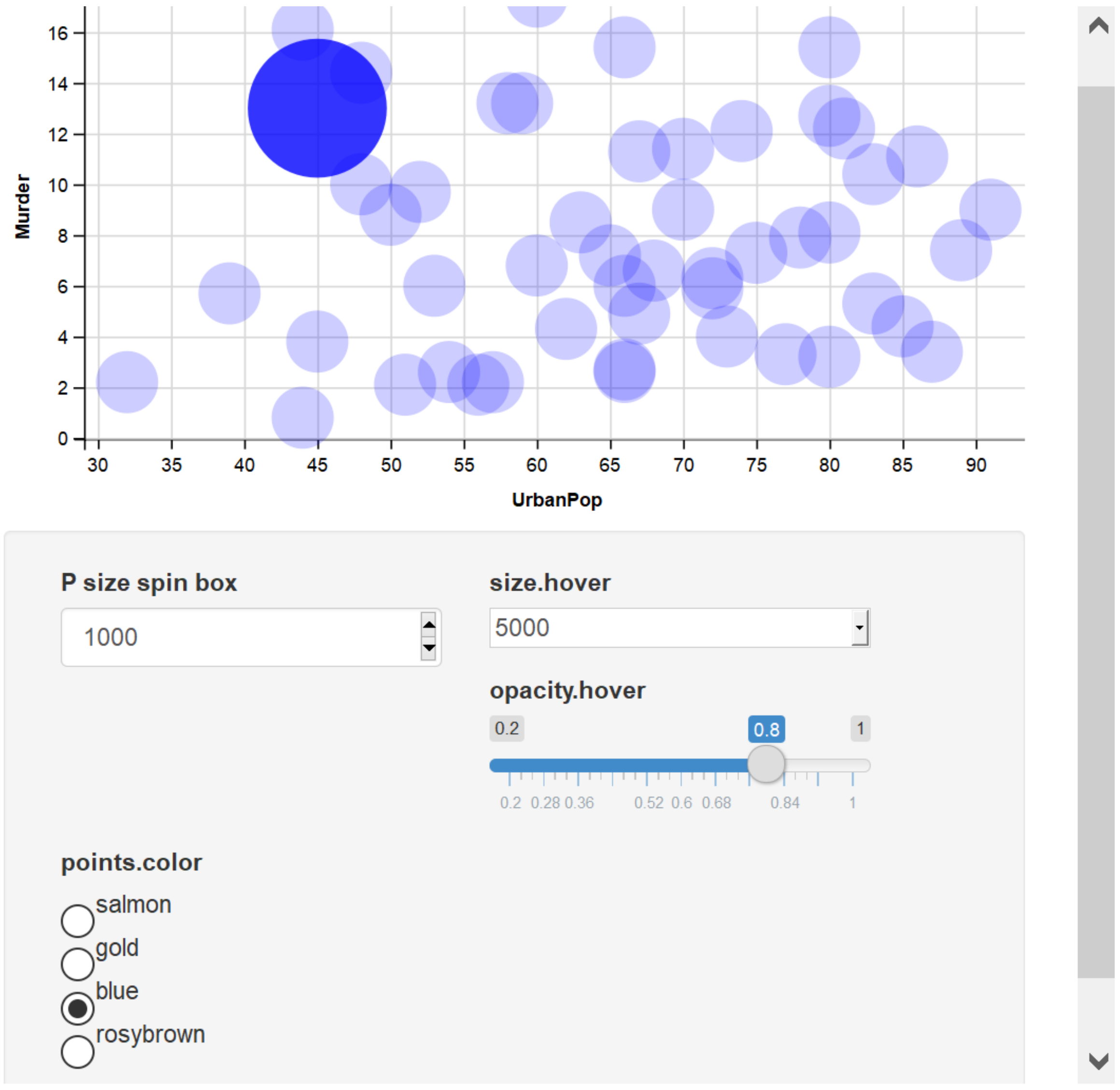
0.6

points.color

[✓ salmon] gold blue rosybrown

Let’s do some interaction whit our result.

1. “P size spin box” control equals to 1000
2. “size.hover” control equals to 5000
3. “opacity.hover” control equals to 0.8
4. “points.color” control equals to blue



P size spin box

1000

size.hover

5000

opacity.hover

0.2 0.8 1

0.2 0.28 0.36 0.52 0.6 0.68 0.84 1

points.color

salmon
gold
[✓ blue]
rosybrown

I know, we can play with it all day!
However, sadly, it is the time to wrap things up.

Before the Conclusion (more resources)

I have been trying hard to find valuable resources for us to make more sense of `ggvis` ' interactive features. However, it seems comparably harder to find adequate ones than those for other packages such as `ggplot2` and `shiny` . It could be because `ggvis` is an evolving package, in my best guess.

- I was not able to find an official `ggvis` cheat sheet like `ggplot2` one, but this one made by an user covers some important things of `ggvis` . It is not a perfect one, but, hopefully, it would be helpful.
- I have been quite excited when I have found the Datacamp website and its YouTube channel as they, especially the website, have fine information related to `ggvis` . Most of all, our **Hadley Wickham** is teaching about R in the Datacamp. However, my excitement has not lasted long as the Website keeps asking me to subscribe. However, as they let us peek their materials related to `ggvis` for some amounts of time, it would be valuable for you to peek and get away from it.
- This is another user's `ggvis` html with interesting examples. As I have mentioned earlier, interactive features will not work in the link. Therefore, we need to copy codes and put `runtime: shiny` in our Rmd header to see interactive features.
- In my opinion, this ppt from LondonR is the best resource among others. It explains `ggvis` from basic concepts to applications.

I hope above resources would be helpful.

Conclusion and take home message

Writing about interactive features of `ggvis` , I have sometimes been irritated owing to frustrating bugs and errors. However, with the fact that `ggvis` is one of the newest packages in R, it seems understandable and reasonable. Also, the fact, I have been learning about the up-to-date package, has compensated my suffering. Moreover, it has been glad to find more values in interactive features of `ggvis` .

This post has covered these interactive features

- hover
    - set_options: resizable and hover_duration
- add_tooltip
    - on
- interactive controls

Take home message

Data visualization is one of the most important parts in data science. It means that it would be beneficial for us to be competent in visualization part. Therefore, getting familiar with many packages related to visualization seems valuable. By this, we can broaden our options for visualization tasks in the Data Analysis Cycle. In other words, with knowledge in various graphical packages, we can allocate adequate packages for each cycle. For example, our `ggvis` seems to fit well with Reporting part. It is because the interactive features of `ggvis` will reduce boredom for both users and reporters. If we become competent in rendering `ggvis` into `Shiny`, the value of `ggvis` will be more than doubled!

In short, knowing various graphical packages including `ggvis` will benefit us in the near future.

References

1. https://rdrr.io/cran/ggvis/src/demo/hover.r
2. https://www.rdocumentation.org/packages/ggvis/versions/0.4.3/topics/set_options
3. https://stackoverflow.com/questions/40322713/how-to-embed-ggvis-interactive-charts-in-rmarkdown
4. https://www.rdocumentation.org/packages/ggvis/versions/0.4.3/topics/add_tooltip
5. https://stackoverflow.com/questions/31119328/r-ggvis-font-and-parameters-of-interactive-text-in-scatter-plot-hover
6. http://ggvis.rstudio.com/interactivity.html
7. https://www.cheatography.com/shanly3011/cheat-sheets/data-visualization-in-r-ggvis-continued/
8. https://campus.datacamp.com/courses/ggvis-data-visualization-r-tutorial/chapter-four-interactivity-and-layers?ex=2
9. https://www.youtube.com/channel/UC79Gv3mYp6zKiSwYemEik9A
10. http://www.di.fc.ul.pt/~jpn/r/GraphicalTools/ggvis.html
11. http://www.londonr.org/presentations/2014/11/LondonR_-_Introduction_to_ggvis_-_Aimee_Gott_-_20141125.pdf