Post02-Tiantian-Fu

Data Visualization with R using GGVIS

Introduction:

The goal of ggvis is to make it easy to build interactive graphics for exploratory data analysis. We have already learned how to use ggvis to build interactive graphics and we were able to connect it to shiny app. But I still think that there are more interesting parts of ggvis for us to learn about. In this post, what I want to do is to explore into ggvis more deeply thoroughly.

This post is divided into three main sessions

1. Dive into interactive controls.
2. Create more types of graphic by controlling the layer type.
3. Build up rich graphics with multiple layers.

Bacis interaction controls

The following example using the built-in data set mtcars allows you to control the size and opacity of points with two sliders:

```
library(ggvis)
library(dplyr)
```

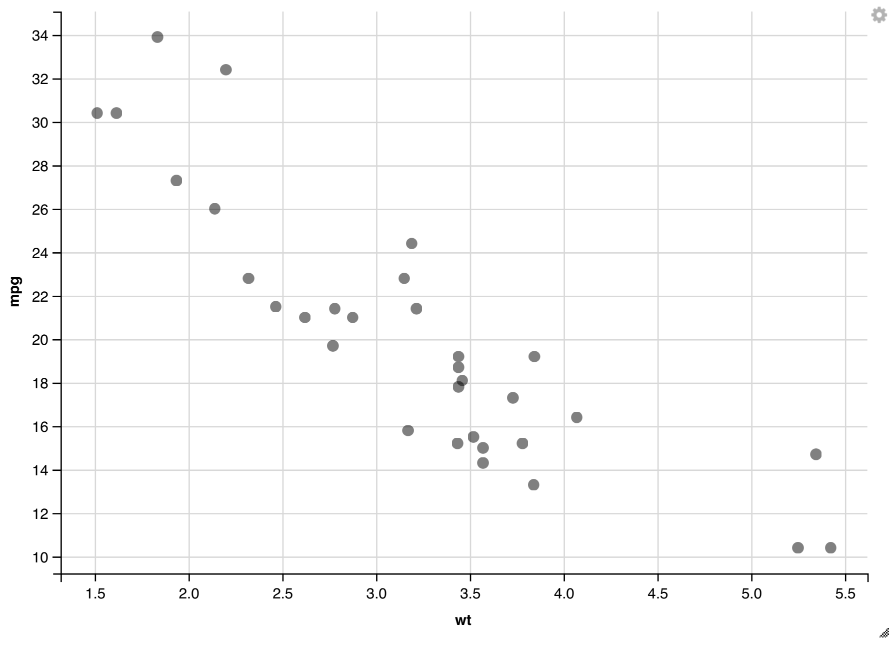
```
##
## Attaching package: 'dplyr'
```

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

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

```
mtcars %>%
  ggvis(~wt, ~mpg,
    size := input_slider(10, 100),
    opacity := input_slider(0, 1)
  ) %>%
  layer_points()
```

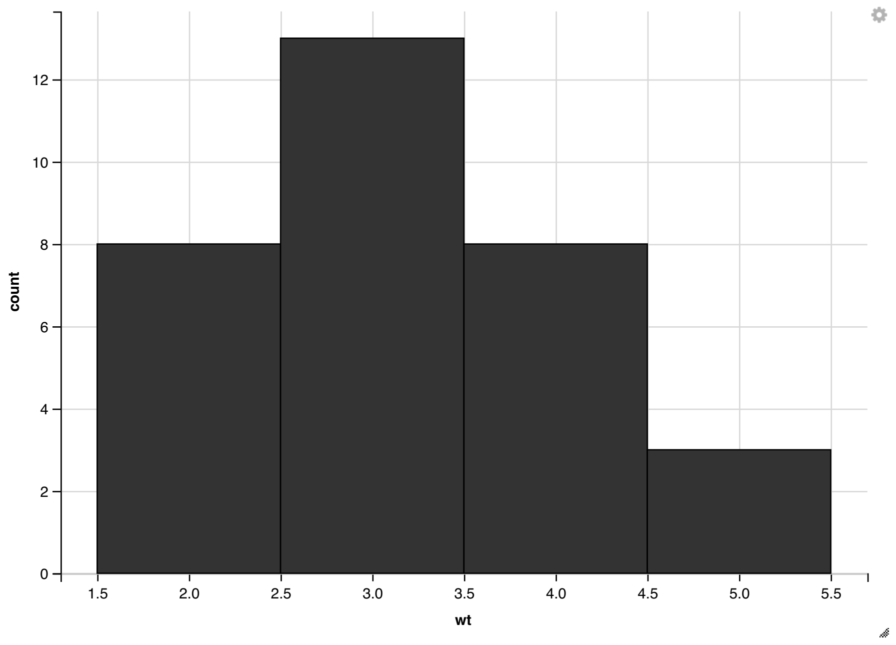
```
## Warning: Can't output dynamic/interactive ggvis plots in a knitr document.
## Generating a static (non-dynamic, non-interactive) version of the plot.
```



You can also connect interactive components to other plot parameters like the width and centers of histogram bins:

```
mtcars %>%
  ggvis(~wt) %>%
  layer_histograms(width =  input_slider(0, 2, step = 0.10, label = "width"),
                   center = input_slider(0, 2, step = 0.05, label = "center"))
```

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



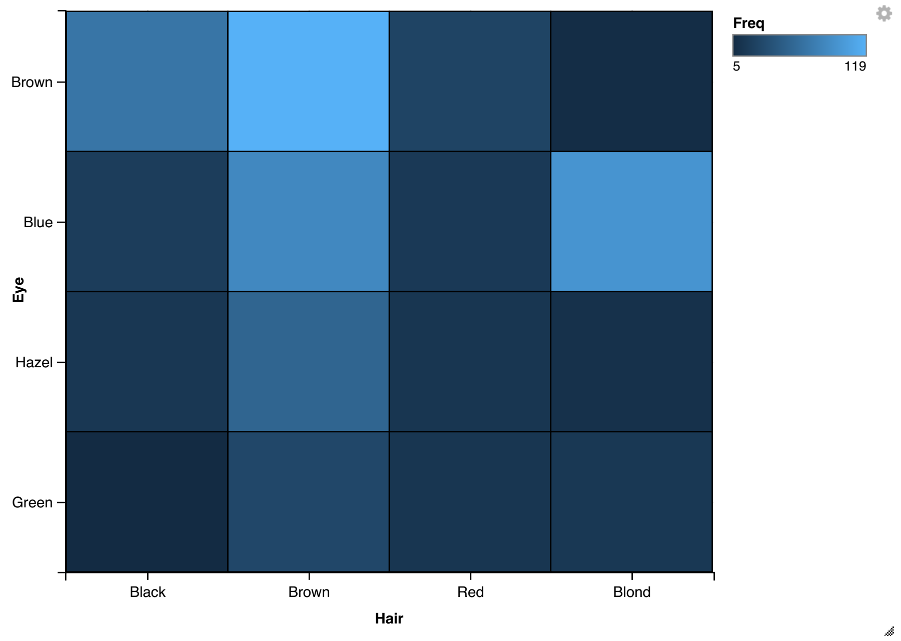
Layers

In ggvis, there are two types of layers. Simple layers correspond directly to vega marks and represent geometric primitives like points, lines and rectangles. Compound layers combine data transformations with one or more simple layers.

Adding a layer to a plot

the first layer I am going to introduce is layer_rects And for **layer_rects()**, you must set two of x, x2, and width, and two of y, y2 and height.

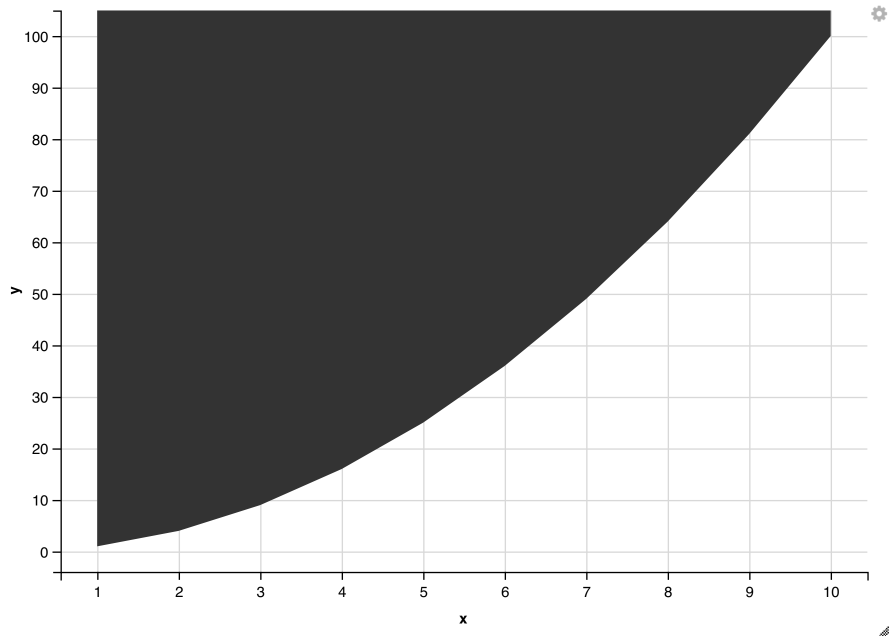
```
hec <- as.data.frame(xtabs(Freq ~ Hair + Eye, HairEyeColor))

hec %>%
  ggvis(~Hair, ~Eye, fill = ~Freq) %>%
  layer_rects(width = band(), height = band()) %>%
  scale_nominal("x", padding = 0, points = FALSE) %>%
  scale_nominal("y", padding = 0, points = FALSE)
```



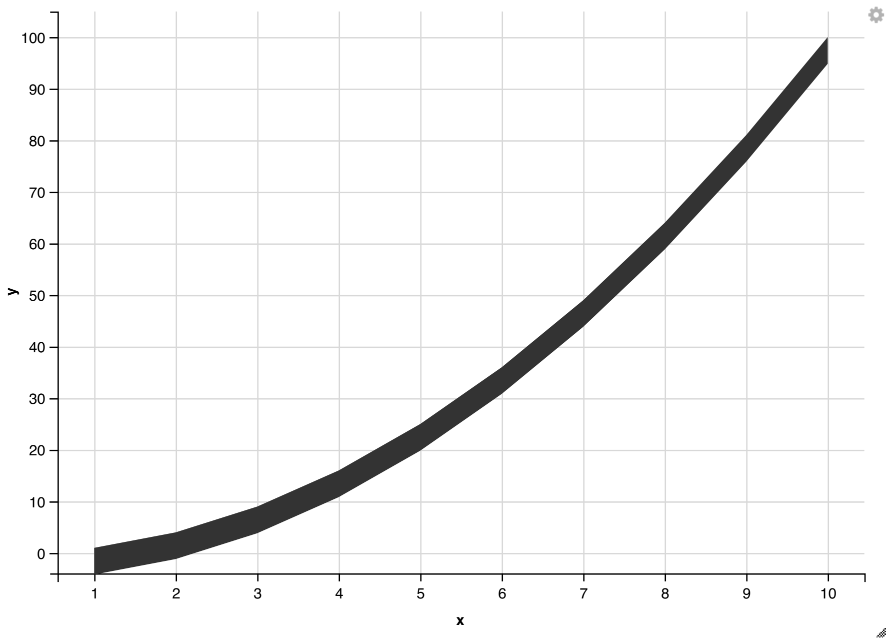
The graph here shows a frequency table of different kinds of colors of eyes and hair.And on the right top of the graph, there is a rectangular bar indicating that the less blue, the more frequency it has. From the graph, we can tell the most amount of people are those whom with brown eyes and brown hair. And the least amount of people are those with brown eyes and blond hair.

Next, I am going to introduce another layer funcion which is called **layer_ribbons**. You need to set two of y, y2 and height:

```
df <- data.frame(x = 1:10, y = (1:10) ^ 2)
df %>% ggvis(~x, ~y, y2 := 0) %>% layer_ribbons()
```

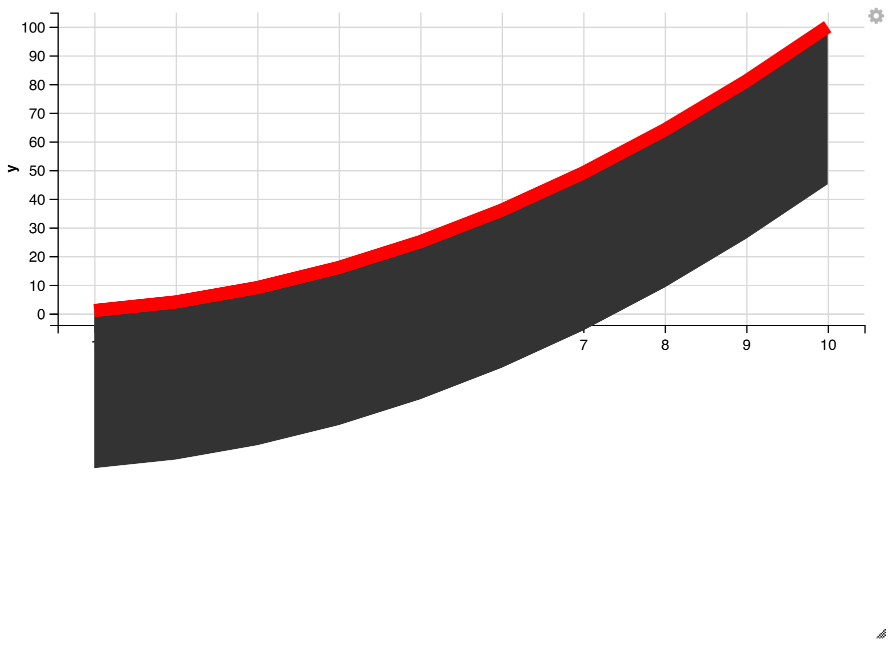


```
# Set height in pixels
df %>% ggvis(~x, ~y, height := 20) %>% layer_ribbons()
```



```
df <- data.frame(x = 1:10, y = (1:10) ^ 2)
df %>% ggvis(~x, ~y) %>%
  layer_ribbons(prop("height", input_slider(0, 100), scale = "y")) %>%
  layer_paths(stroke := "red", strokeWidth := 10)
```

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

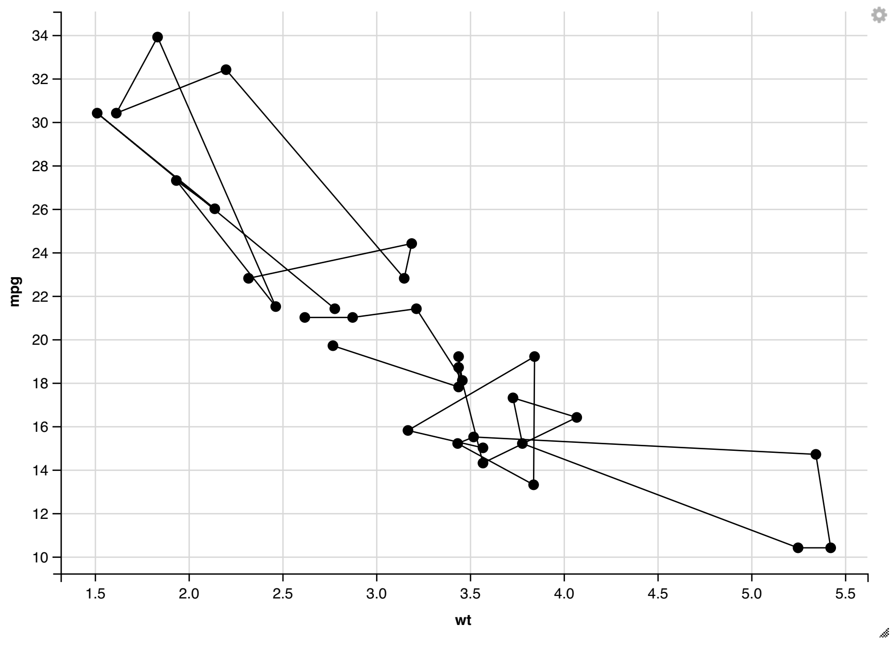


Height can only be mapped to a constant, because it does not have an obvious scale associated with it. You could force height to use the y scale, but that doesn't work - the area hangs below the y line, and increasing the value of height makes the area narrower! What's going on is that the underlying graphics device has (0, 0) in the top-left corner, and so the y-scale is upside down. As you increase height, it's mapped like a y variable so bigger values are further away.

Grouping

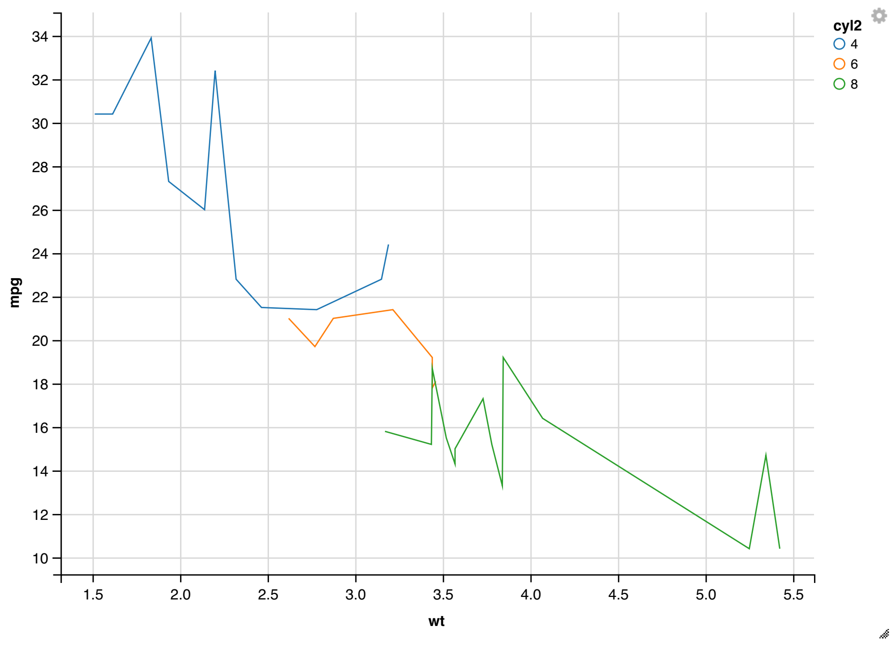
In ggplot2, grouping was a special aesthetic; in ggvis, grouping is a data transformation: use group_by() to split your data up into pieces given a specified variable, or **auto_split()** to split up by any categorical variable included in the plot:

```
mtcars %>% ggvis(~wt, ~mpg) %>%
  layer_points() %>%
  group_by(cyl) %>%
  layer_paths()
```



Some layers, like **layer_line()**, include auto_split() so will split automatically:

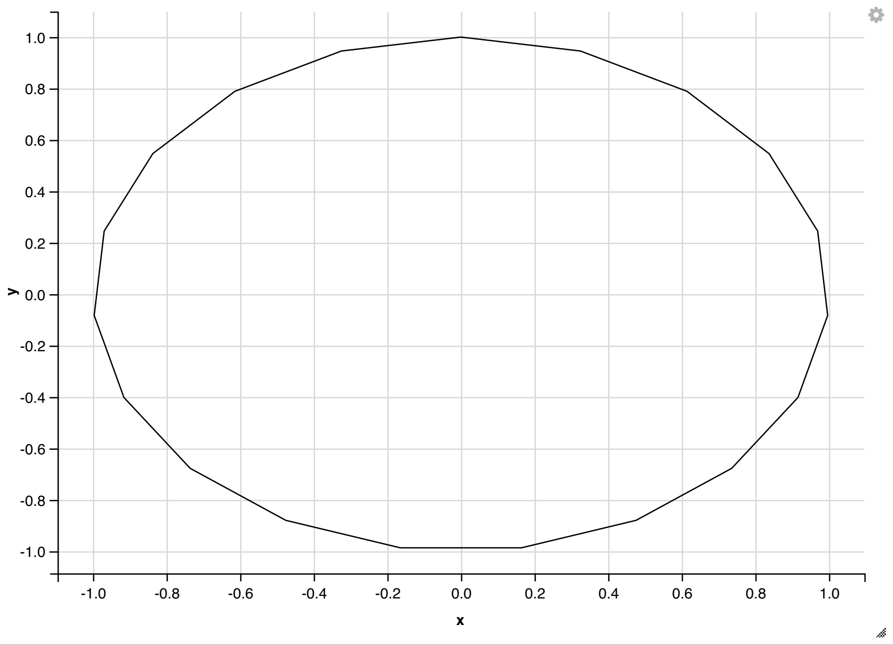
```
mtcars %>%
  dplyr::mutate(cyl2 = factor(cyl)) %>%
  ggvis(~wt, ~mpg, stroke = ~cyl2) %>%
  layer_lines()
```



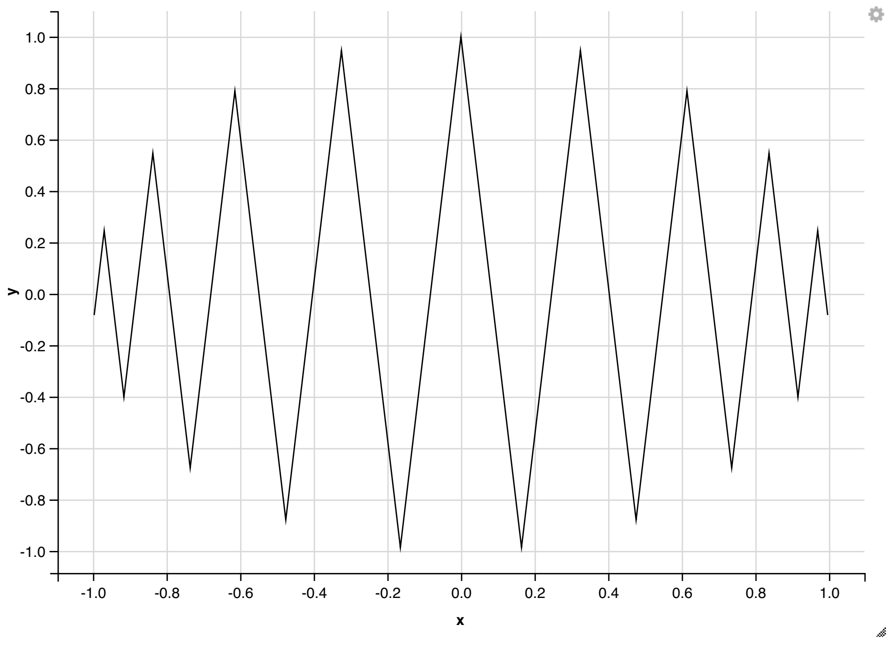
Compound layers

The four most common compound layers are: **layer_lines()** which automatically orders by the x variable:

```
t <- seq(0, 2 * pi, length = 20)
df <- data.frame(x = sin(t), y = cos(t))
df %>% ggvis(~x, ~y) %>% layer_paths()
```



```
df %>% ggvis(~x, ~y) %>% layer_lines()
```

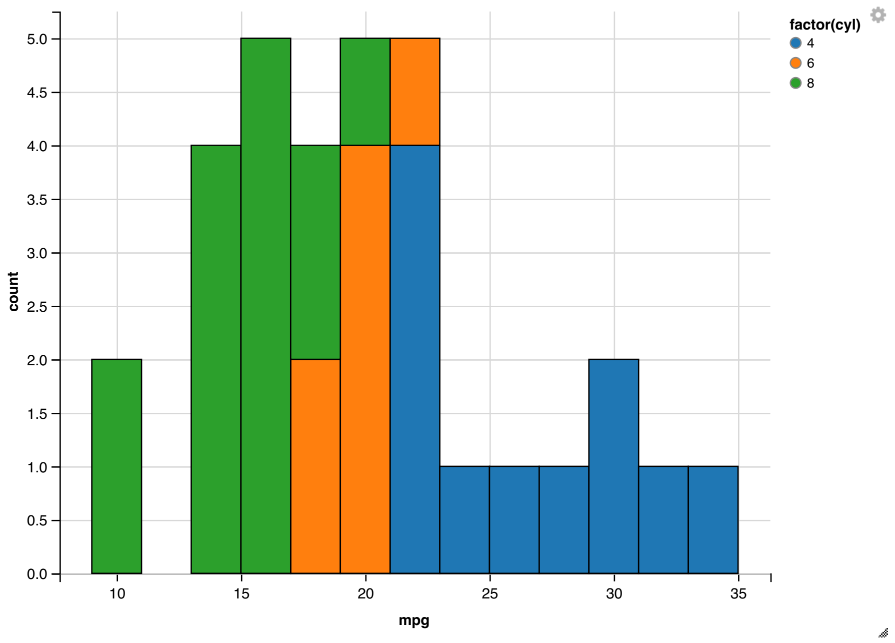


layer_histograms() and **layer_freqpolys()** which allows you to explore the distribution of continuous. Both layers first bin the data with compute_bin() then display the results with either rects or lines.

```
mtcars %>%
  ggvis(~mpg, fill = ~factor(cyl)) %>%
  group_by(cyl) %>%
  layer_histograms(width = 2, stack=TRUE)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```



Conclusion

The ggvis package is used to make interactive data visualizations. The fact that it combines shiny's reactive programming model and dplyr's grammar of data transformation make it a useful tool for data scientists. This package may allows us to implement features like interactivity, but on the other hand every interactive ggvis plot must be connected to a running R session.

Take home messages

I hope you gained a deeper understanding about ggvis through this post. Package ggvis is actrually a very useful tool used to turn a dataset into a visualisation, setting up default mappings between variables in the dataset and visual properties.

Resources

http://ggvis.rstudio.com

https://ggvis.rstudio.com/ggvis-basics.html#interaction

https://cran.r-project.org/web/packages/ggvis/README.html

https://www.rdocumentation.org/packages/ggvis/versions/0.4.3

https://www.datacamp.com/courses/ggvis-data-visualization-r-tutorial

https://www.youtube.com/watch?v=g4n2B-akEgI

https://www.dezyre.com/data-science-in-r-programming-tutorial/ggvis