Creating Boxplots with ggplot2 and ggvis

Motivation

In my first post, I talked about how to interpret boxplots and how to create boxplots using R base functions. Now that we know more about data visualization, I believe that we are able to render our boxplots more nicely. Since we haven't learnt much about boxplots in ggplot2 and ggvis, in this post, I would like to talk about how to create nice boxplots using these packages in R.

Instruction

Before you continue reading, here's some information for you to follow the content of this post better: The tools used in this post are Rstudio version 1.1.383 and R version 3.4.2. And the required packages are ggplot2 version 2.2.1 and ggvis version 0.4.3. With all these tools, you will be able to replicate everything in this post!

R base Boxplots

First, let's see how a boxplot created by R base functions looks.

Here, I use the same sample data from my post 1: Say there are 30 students in Discussion 101 of Stat133, and their midterm scores form a vector *midterm_101* :

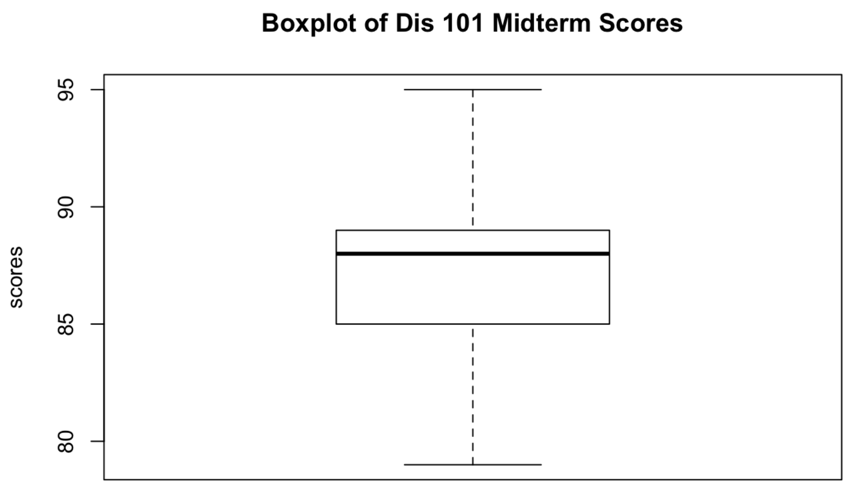
```
# midterm scores for Dis 101 students
midterm_101 <- c(79, 80, 81, 82, 82, 82, 84, 85, 85, 86, 86, 87, 88, 88, 88,
                 88, 88, 89, 89, 89, 89, 89, 89, 89, 90, 92, 92, 93, 95, 95)

# summary of midterm_101
summary(midterm_101)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    79.0    85.0    88.0    87.3    89.0    95.0
```

To create a boxplot by R base function, we apply the function boxplot() to the vector *midterm_101*:

```
# create a boxplot of Dis 101 midterm scores
boxplot(midterm_101,
        main = "Boxplot of Dis 101 Midterm Scores",
        ylab = "scores")
```



Such boxplot looks clear, but it's a bit boring, isn't it?

To make a fancier one, let's start using our R packages.

Boxplots with the package ggplot2

1. Create a basic boxplot

First, remember to load the package:

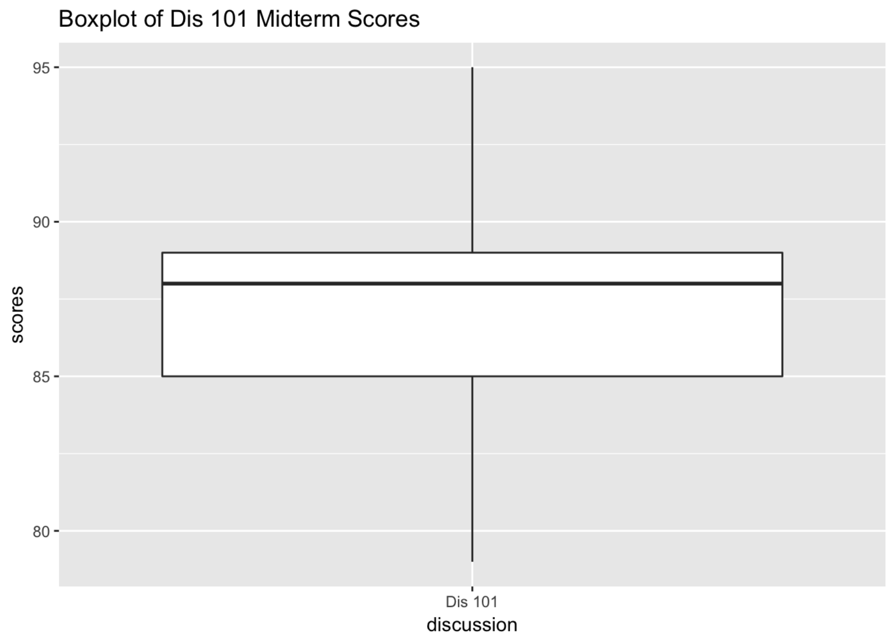
```
# importing the ggplot2 library
library(ggplot2)
```

Now we can start creating boxplots! However, this is not as easy as the R base case. First, we need to convert the vector *midterm_101* into a data frame *dat101* in order to use it as an argument for the function ggplot(). Also, we will need to specify the x variable in ggplot(), so let's create another variable *discussion* representing the discussion name.

```
# create a data frame "dat101"
dat101 <- data.frame(
  scores = midterm_101,
  discussion = rep("Dis 101", 30)
    )
```

The function for boxplots is *geom_boxplot*:

```
# create a boxplot
ggplot(dat101, aes(x = discussion, y = scores)) +
  geom_boxplot() +
  ggtitle("Boxplot of Dis 101 Midterm Scores")
```

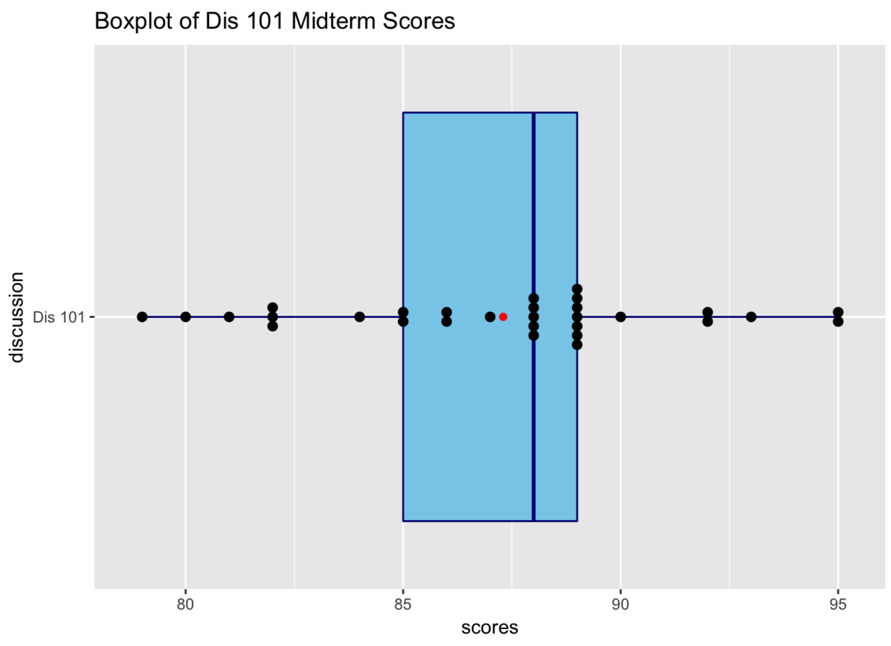


2. Create a fancy boxplot

This is still a bit plain, right? Let's add some codes to create a fancier plot:

```
# create a fancier boxplot
ggplot(dat101, aes(x = discussion, y = scores)) +
  coord_flip() +
  geom_boxplot(fill = "sky blue", color = "navy") +
  ggtitle("Boxplot of Dis 101 Midterm Scores") +
  geom_dotplot(binaxis='y', stackdir='center', dotsize = 0.4) +
  stat_summary(fun.y = mean, geom = "point", color = "red")
```

```
## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```



Let’s look at the function again and make some observations:

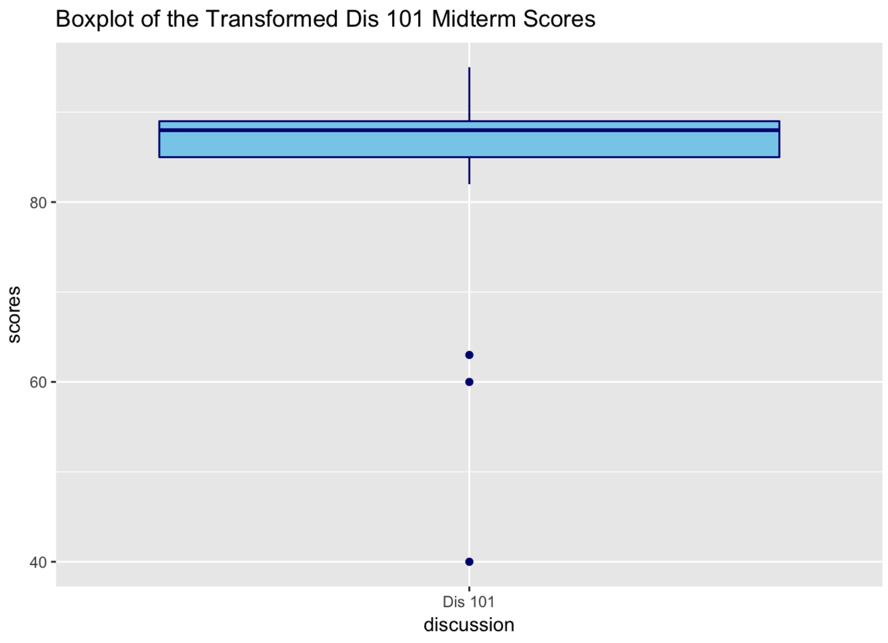
- The function coord_flip() allows us to create a horizontal boxplot.
- The arguments **fill** and **color** refer to the color inside the box and the border color, respectively.
- The function geom_dotplot() allows us to show explicitly each score on the plot. for example, the vertical line with 5 dots corresponds to the mode score (88).
- The function stat_summary() allows us to display a point (the red dot) indicating the average score (87.3).

3. Boxplots with outliers

To see how outliers look in ggplot2 boxplots, let’s change the lowest three midterm scores in Discussion 101 to 40, 60, and 63, and make a new boxplot of the transformed data:

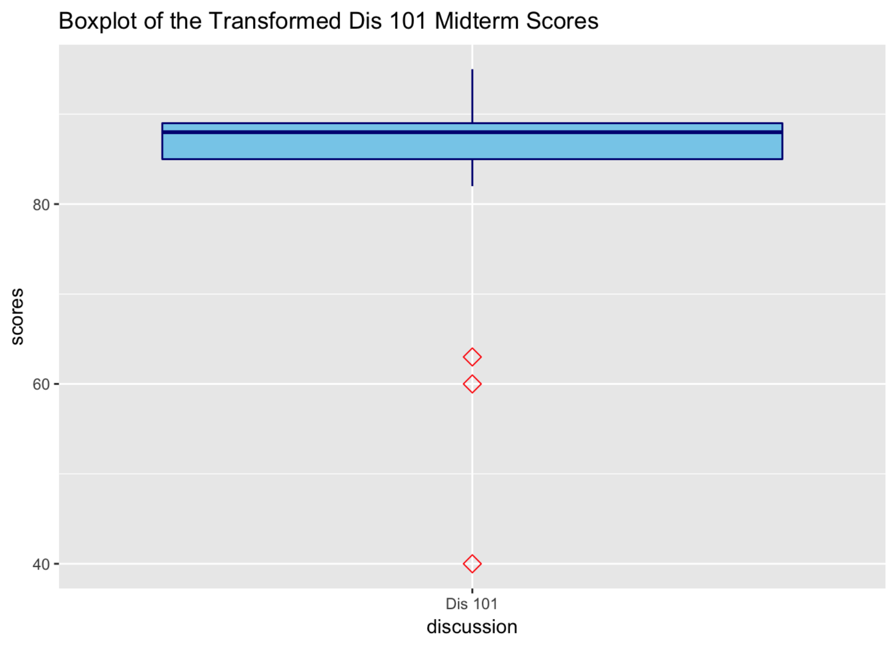
```
# transform midterm scores for Dis 101 students
midterm_101_outlier <- c(40, 60, 63, 82, 82, 82, 84, 85, 85, 86, 86, 87, 88, 88, 88,
                         88, 88, 89, 89, 89, 89, 89, 89, 89, 90, 92, 92, 93, 95, 95)
# create a new data frame dat101_outlier
dat101_outlier <- data.frame(
  scores = midterm_101_outlier,
  discussion = rep("Dis 101", 30)
    )

# create a new boxplot
ggplot(dat101_outlier, aes(x = discussion, y = scores)) +
  geom_boxplot(fill = "sky blue", color = "navy") +
  ggtitle("Boxplot of the Transformed Dis 101 Midterm Scores")
```



The lowest three scores are represented by dots on the boxplot because they are considered as outliers. That is, they are too low compared to other scores that we might have to ignore them while doing statistical analysis. Now, let’s try to change the appearance of these outliers:

```
# change the appearance of the outliers
ggplot(dat101_outlier, aes(x = discussion, y = scores)) +
  geom_boxplot(fill = "sky blue", color = "navy",
               outlier.shape = 5, outlier.color = "red", outlier.size = 3) +
  ggtitle("Boxplot of the Transformed Dis 101 Midterm Scores")
```



As shown above, the outliers and the box can be in different colors. We can change the appearance of outliers by the arguments **outlier.color**, **outlier.fill**, **outlier.shape**, **outlier.size**, **outlier.alpha**, etc.

4. Comparing different categories in one boxplot

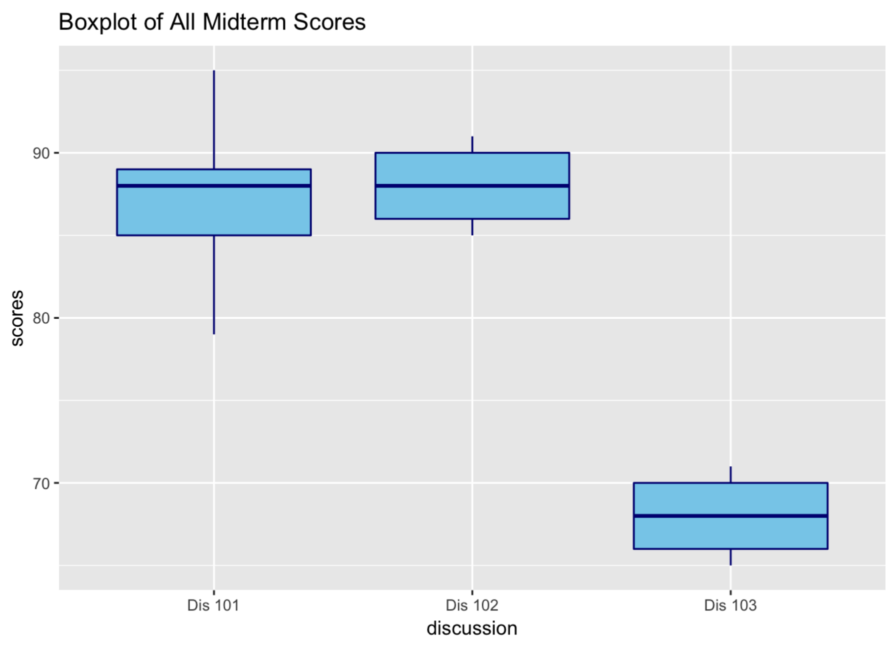
Now we know how to create boxplots of datasets with a single category. To see how to compare different categories in one boxplot, we need more data. Again, I will reuse the sample data from my post 1. Suppose there are also 30 students in Discussion 102 and 103, and their midterm scores form the vectors *midterm_102* and *midterm_103*:

```
# midterm scores for Dis 102 & 103 students
midterm_102 <- c(85, 85, 85, 85, 85, 86, 86, 86, 86, 87, 87, 87, 87, 87, 88,
                 88, 89, 89, 89, 89, 89, 90, 90, 90, 90, 91, 91, 91, 91, 91)
midterm_103 <- c(65, 65, 65, 65, 65, 66, 66, 66, 66, 67, 67, 67, 67, 67, 68,
                 68, 69, 69, 69, 69, 69, 70, 70, 70, 70, 71, 71, 71, 71, 71)
```

In order to plot a boxplot, we have to include all midterm scores in one data frame *dat_101_102_103*:

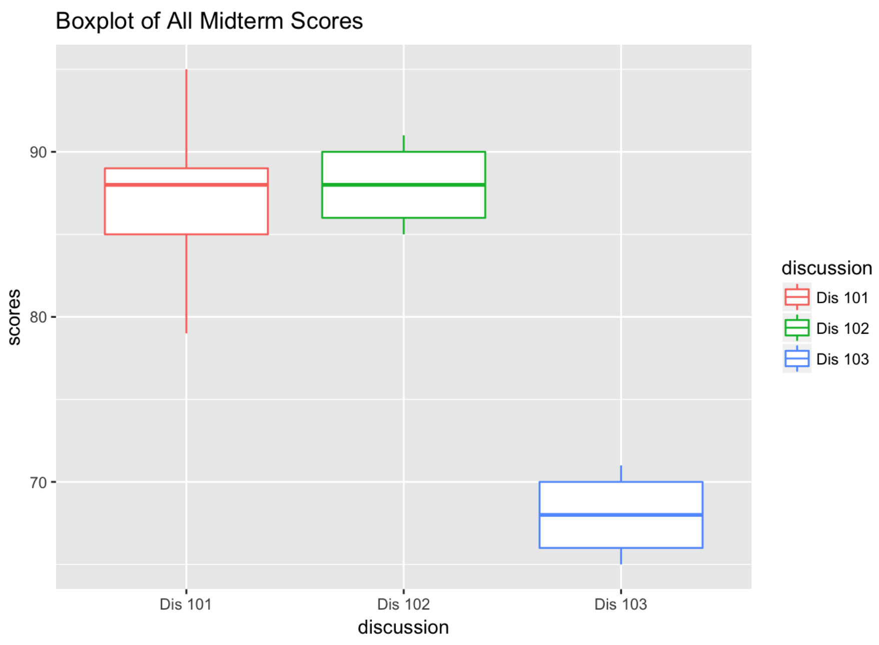
```
# create a data frame "dat_101_102_103"
dat_101_102_103 <- data.frame(
  scores = c(midterm_101, midterm_102, midterm_103),
  discussion = rep(c("Dis 101", "Dis 102", "Dis 103"), each = 30)
    )
dat_101_102_103$discussion <- as.factor(dat_101_102_103$discussion)

# create a boxplot from "dat_101_102_103"
ggplot(dat_101_102_103, aes(x = discussion, y = scores)) +
  geom_boxplot(fill = "sky blue", color = "navy") +
  ggtitle("Boxplot of All Midterm Scores")
```

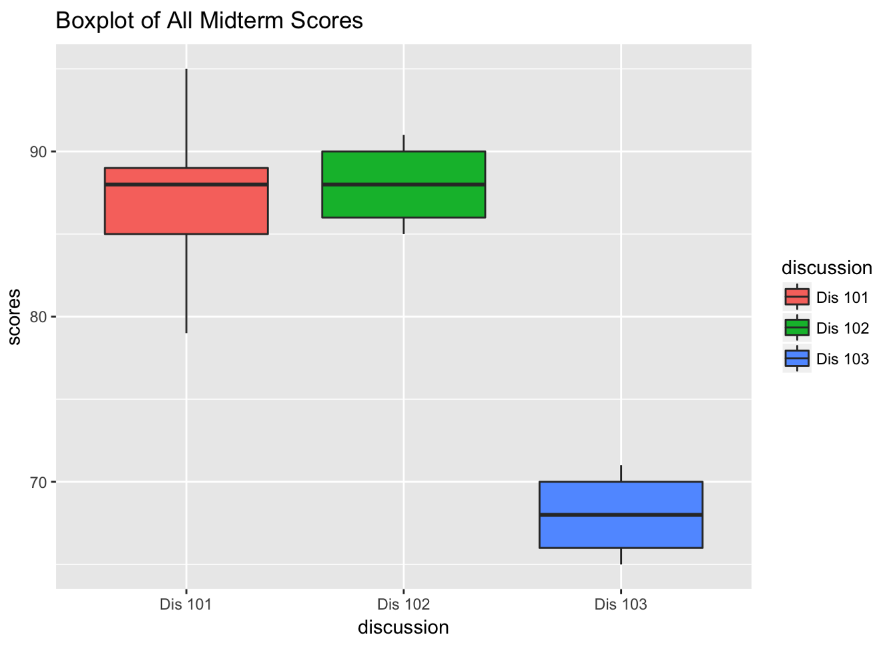


If we want the plots of different discussions to have different colors, we set the **color** or the **fill** argument as the categorical variable ("discussion" in this case).

```
# different discussions with different border colors
ggplot(dat_101_102_103, aes(x = discussion, y = scores, color = discussion)) +
  geom_boxplot() +
  ggtitle("Boxplot of All Midterm Scores")
```



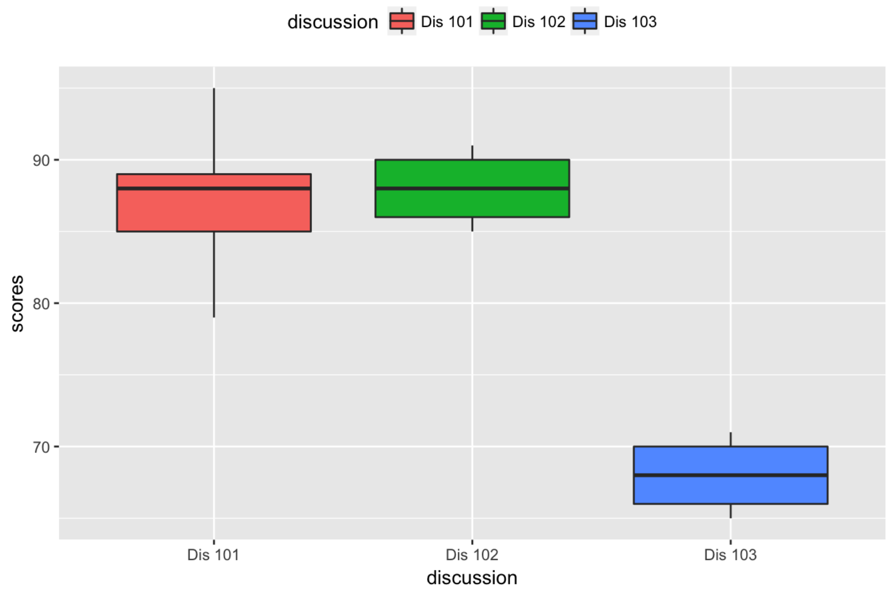
```
# different discussions with different colors inside the box
ggplot(dat_101_102_103, aes(x = discussion, y = scores, fill = discussion)) +
  geom_boxplot() +
  ggtitle("Boxplot of All Midterm Scores")
```



Notice that if we set different colors by groups, we will see the **legend** on the right of the graph. We can adjust the position of it or hide it by the function theme().

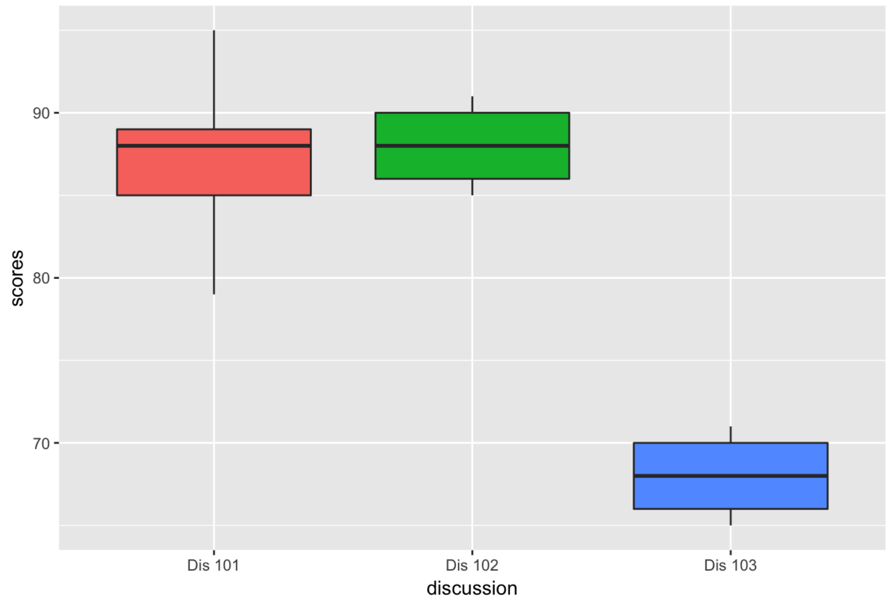
```
# Adjust the legend to the top
ggplot(dat_101_102_103, aes(x = discussion, y = scores, fill = discussion)) +
  geom_boxplot() +
  ggtitle("Boxplot of All Midterm Scores") +
  theme(legend.position = "top")
```

Boxplot of All Midterm Scores



```r
# Hide the legend
ggplot(dat_101_102_103, aes(x = discussion, y = scores, fill = discussion)) +
  geom_boxplot() +
  ggtitle("Boxplot of All Midterm Scores") +
  theme(legend.position = "none")
```

Boxplot of All Midterm Scores



Now that we know how to create nice boxplots using "ggplot2", let's turn to the package "ggvis".

Boxplots with the package ggvis

1. Create a basic boxplot

First, remember to load the package:

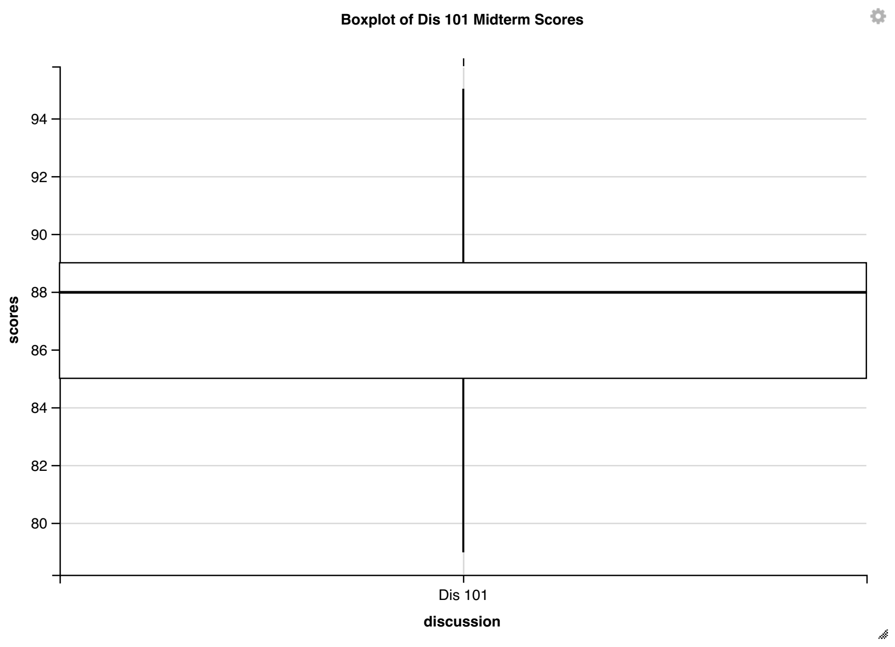
```r
# importing the ggvis library
library(ggvis)
```

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

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

Life is much easier this time because we've already created all the data frames required to draw our boxplots, and all we need is to change the ggplot2 syntax to the one for ggvis. Note that in ggvis, the function for boxplots is layer_boxplots(). Let's first create a boxplot for Discussion 101 midterm scores from the data frame *dat101*.

```
# create a basic boxplot
dat101 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots() %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of Dis 101 Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```



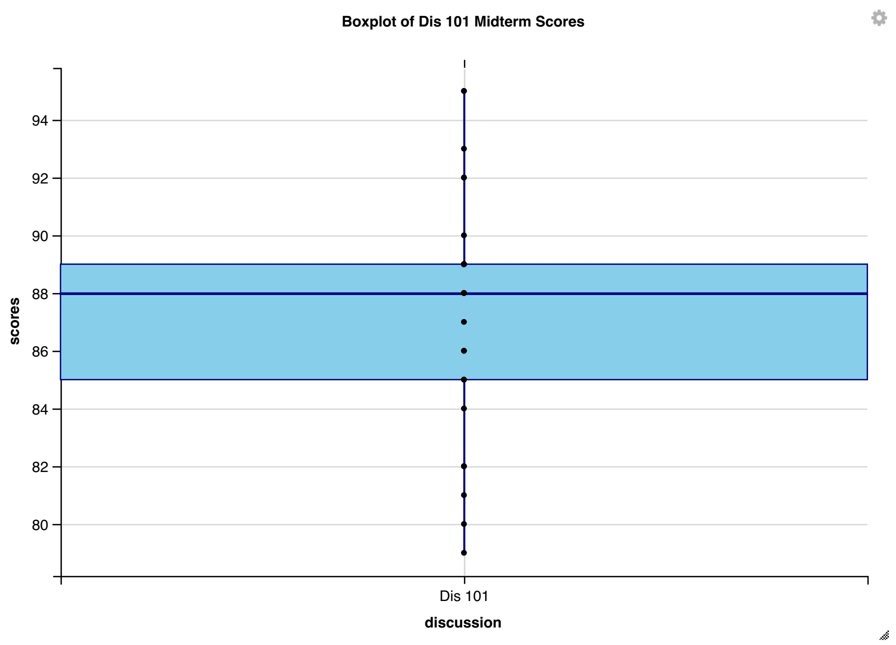
The codes are a bit long for such a plain plot. This is because that ggvis does not come with the helpful function ggtitle(), as this package is designed mainly for interactive plots, which can be combined with shiny app and do not require titles. Therefore, there are currently no simple ways to add titles to ggvis plots.

However, I used a trick to add a title: use the function add_axis to create another x axis on the top of the graph, change the axis title to the desired plot title “Boxplot of Dis 101 Midterm Scores”, and then use the argument **properties** to set the stroke color to white and the font size of the label to 0, so that only the axis title is visible.

2. Create a fancy boxplot

As in the ggplot2 case, we can add some codes to make the boxplot look better:

```
# create a fancier boxplot
dat101 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill := "skyblue", stroke := "navy") %>%
  layer_points(size := 15, prop("x", ~discussion, scale = "xcenter")) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of Dis 101 Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```



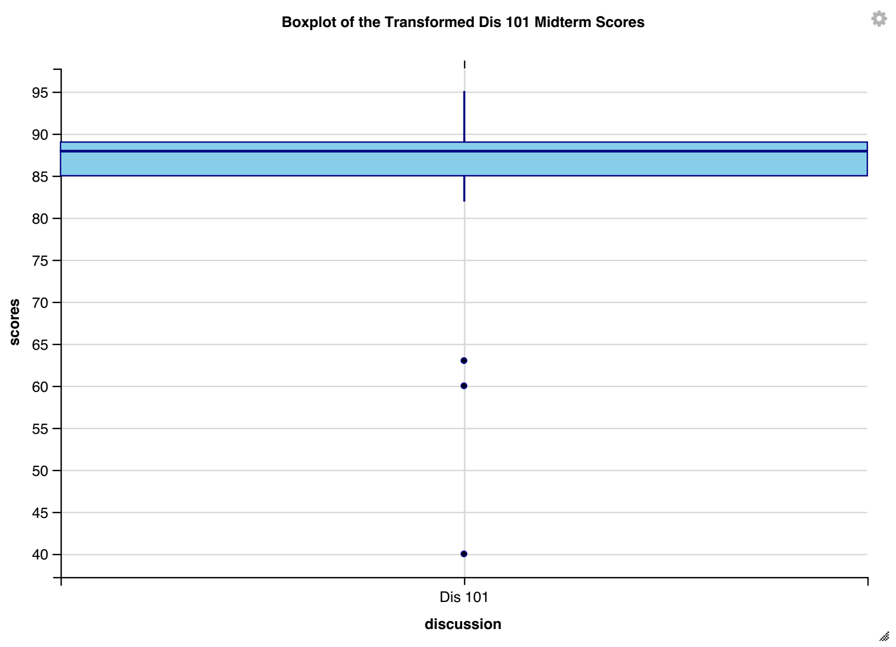
In ggvis, we can use the arguments **fill** and **stroke** to change the color of boxplots.

Notice that ggvis also lacks the function coord_flip(), so we can't create horizontal boxplot easily. Also, although the function layer_points() works similarly as geom_dotplot() in ggplot2, it doesn't display every data point on the plot. For example, 5 students got 88 on midterm, but we can only see one dot corresponding to the score 88.

3. Boxplots with outliers

As in the ggplot2 case, we are interested in how outliers look in ggvis boxplots. We can use the data frame dat101_outlier to make a new boxplot with outliers:

```
# create a boxplot with outliers
dat101_outlier %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill := "skyblue", stroke := "navy", size := 13) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of the Transformed Dis 101 Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```

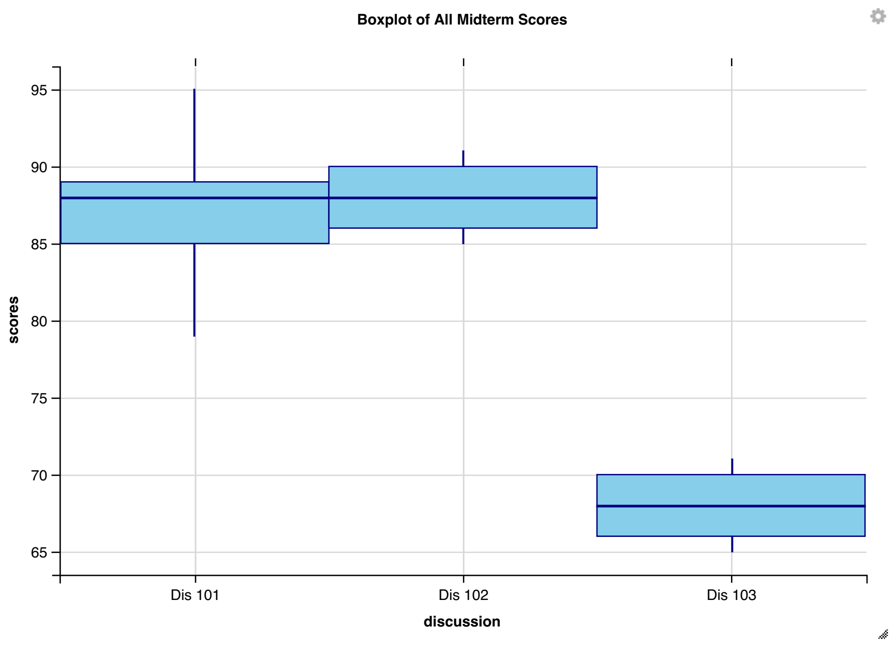


The argument **size** can be used to change the size of the dots, which represent the outliers. Yet, unlike ggplot2, ggvis does not have lots of arguments designed for changing the outlier appearance.

4. Comparing different categories in one boxplot

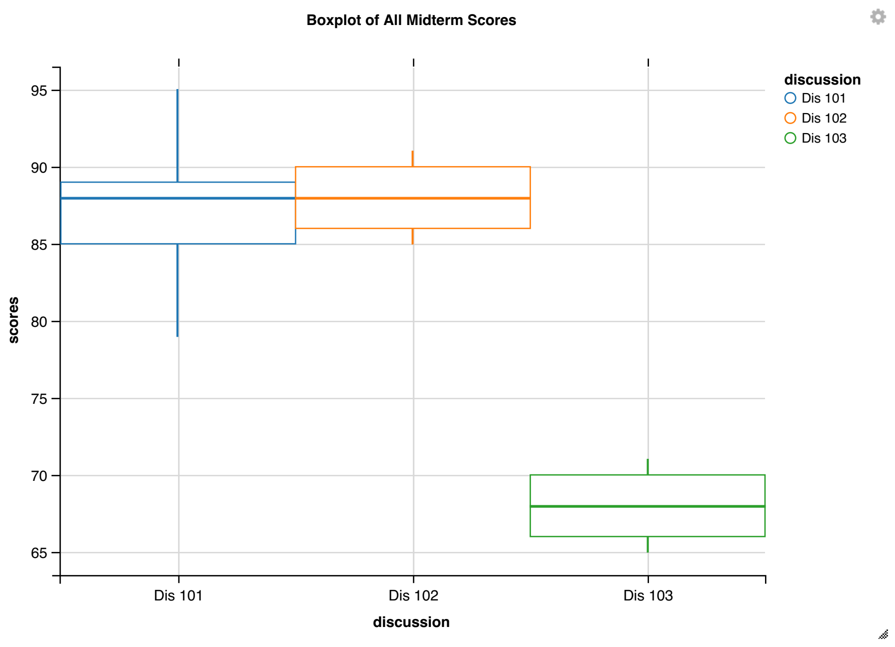
We can also compare different categories in one boxplot with ggvis. Let's create one such plot from our data frame *dat_101_102_103*:

```r
# create a boxplot from "dat_101_102_103"
dat_101_102_103 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill := "skyblue", stroke := "navy", width = 1) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of All Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```

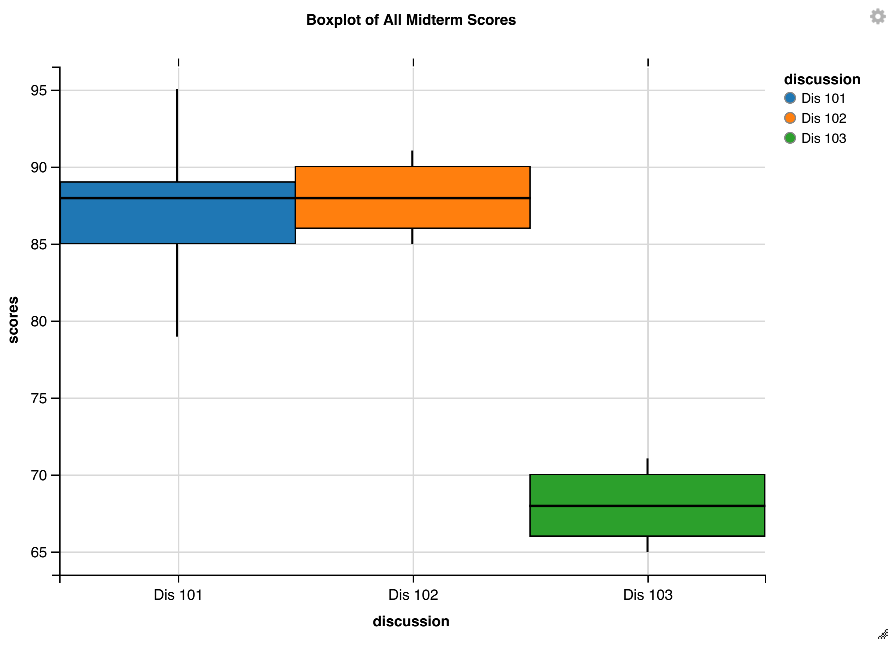


Let's try to set different colors for different discussions, as what we did in the ggplot2 case. For ggvis, this can be done by setting the **stroke** or the **fill** argument as the categorical variable ("discussion").

```r
# different discussions with different border colors
dat_101_102_103 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(stroke = ~discussion, width = 1) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of All Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```



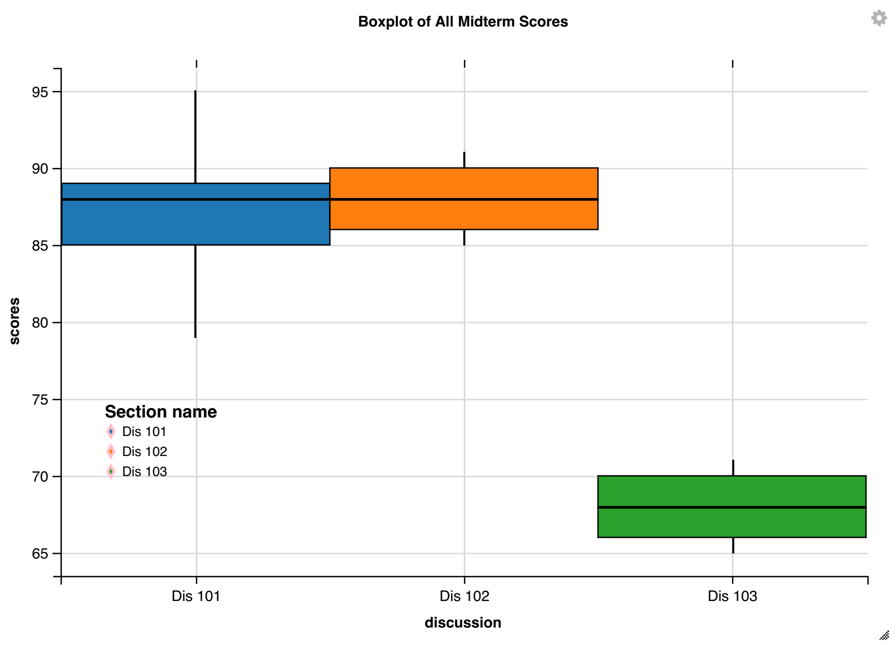
```r
# different discussions with different colors inside the box
dat_101_102_103 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill = ~discussion, width = 1) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of All Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           )
```



Be careful! When we set the color explicitly, we use the symbol **:=** before the color name. However, here we use the equal sign **=**, followed by **~** and the variable name. The symbol **~** indicates that the following variable is inside the data frame.

Notice that the **legend** is at the top-right corner. As in the ggplot2 case, we can adjust the position of it.

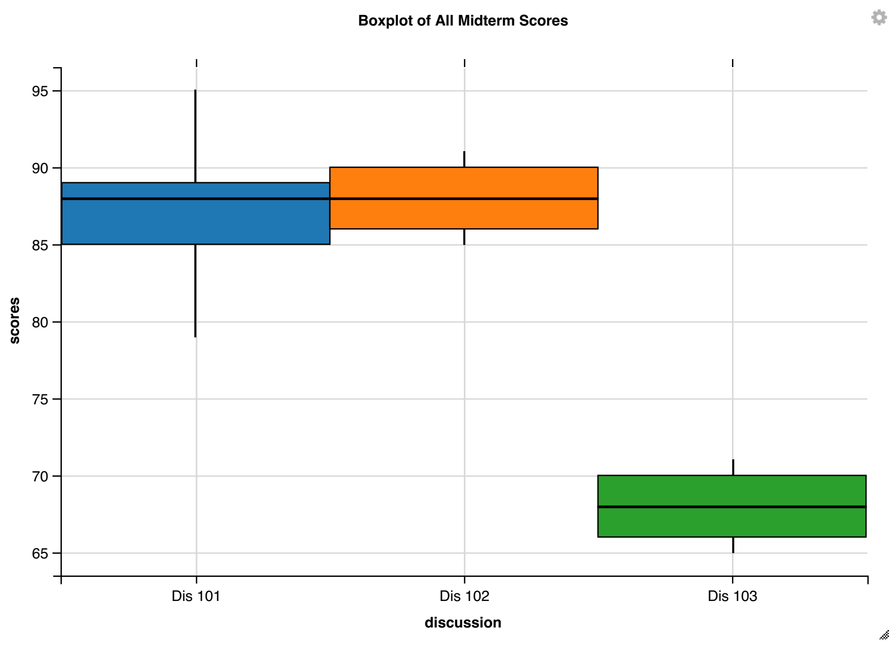
```r
# change the appearance and the position of the legend
dat_101_102_103 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill = ~discussion, width = 1) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of All Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           ) %>%
  add_relative_scales() %>%
  add_legend("fill", title = "Section name",
    properties = legend_props(
      legend = list(x = scaled_value("x_rel", 0.05),
                    y = scaled_value("y_rel", 0.35)
                    ),
      title = list(fontSize = 13),
      labels = list(fontSize = 10),
      symbol = list(stroke = "pink", strokeWidth = 2,
        shape = "diamond", size = 20)
    )
  )
```



As we see above, the argument **properties** inside the function add_legend() allows us to customize both the appearance and the position of the legend.

We can also hide the legend:

```
# Hide the legend
dat_101_102_103 %>%
  ggvis(x = ~discussion, y = ~scores) %>%
  layer_boxplots(fill = ~discussion, width = 1) %>%
  add_axis("x", title = "discussion") %>%
  add_axis("x", orient = "top", ticks = 0,
           title = "Boxplot of All Midterm Scores",
           properties = axis_props(
             axis = list(stroke = "white"),
             labels = list(fontSize = 0)
             )
           ) %>%
  hide_legend("fill")
```



Lastly…

As I showed in post 1, boxplots are more useful than you might have thought. But to present your findings via this special kind of plot, it's important to plot it nicely and clearly. So next time when you are creating boxplots, try to make use of the packages *ggplot2* and *ggvis*!

References

- ggplot2 box plot : Quick start guide - R software and data visualization http://www.sthda.com/english/wiki/ggplot2-box-plot-quick-start-guide-r-software-and-data-visualization
- Creating plots in R using ggplot2 - part 10: boxplots http://t-redactyl.io/blog/2016/04/creating-plots-in-r-using-ggplot2-part-10-boxplots.html

- R ggplot2 Boxplot https://www.tutorialgateway.org/r-ggplot2-boxplot/
- A box and whiskers plot (in the style of Tukey) http://ggplot2.tidyverse.org/reference/geom_boxplot.html
- Guides: Axes and Legends https://ggvis.rstudio.com/axes-legends.html
- Align multiple graphs using ggvis (layer_points over layer_boxplots) https://stackoverflow.com/questions/33631492/align-multiple-graphs-using-ggvis-layer-points-over-layer-boxplots
- Add a plot title to ggvis https://stackoverflow.com/questions/25018598/add-a-plot-title-to-ggvis
- Data Visualisation with R using GGVIS https://www.dezyre.com/data-science-in-r-programming-tutorial/ggvis