The Wonders of ggplot2

Andrew Chen

October 31, 2017

Since its introduction into R in 2005, the ggplot2 package has been heralded for its ability to create beautiful data visualizations without having to make the user do much work. We've explored a couple of basic ggplot graphics in class, such as the barplot, scatterplot, boxplot, etc. But I would like to delve a little deeper into a few of the more extensive, more wondrous, and perhaps even stranger aspects of ggplot's powers.

The data I will use for this post is primarily the NBA data that we have been using for the past 2 months, particularly the version of the data that has all the game statistics for each player (points, blocks, assists, etc.) that we used in HW2 and HW3. But I will explore alternative representations of the data using ggplot's versatile graphing options.

Preliminaries

We'll start just by cleaning up the data a little bit, along with adding a couple of variables to the data. The code chunk below is basically copied from HW2.

```
#importing libraries
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(magrittr)

#reading nba2017-player-statistics.csv
data <- read.csv('~/Desktop/stat133/nba2017-player-statistics.csv', colClasses = c("character", "character", "fact
or", "character", "double", rep("integer", 19)))

#adding variables to the data
data <- mutate(data, Missed_FG = data$FGA - data$FGM,
               Missed_FT = data$FTA - data$FTM,
               PTS = data$Points3 * 3 + data$Points2 * 2 + data$FTM,
               REB = data$OREB + data$DREB,
               MPG = data$MIN / data$GP,
               )

#adding EFF to the data
data <- mutate(data, EFF = (data$PTS + data$REB + data$AST + data$STL + data$BLK - data$Missed_FG - data$Missed_FT
- data$TO) / data$GP)

#changing column "Experience" into integers
data$Experience[data$Experience == 'R'] <- 0
data$Experience <- as.integer(data$Experience)
```

Ranking Teams by EFF: Diverging Plots and z-scores

We've done team rankings by EFF before in the homework, but I'd like to use that same task to explore some of the aesthetic and practical applications of ggplot, while also using the statistical concept of z-score.

First, I grouped the data into teams, and created a data frame providing the average EFF for each team.

```
team_EFF <- data %>% group_by(Team) %>% summarise(avg_EFF = mean(EFF))
```

Then using these averages, I took their mean and standard deviation to find the z-score of each average, and created a column for those z-scores, called eff_z.

I also created a column called eff_type; a certain team's EFF would be categorized either as "below", for below average (z-score < 0), or above average, for above average (z-score > 0).

Lastly, I arranged the teams according to their EFF z-score.

```
team_EFF <- team_EFF %>% mutate(eff_z = round((avg_EFF - mean(avg_EFF)) / sd(avg_EFF), 2)) %>% mutate(eff_type = i
felse(eff_z < 0, "below", "above")) %>% arrange(eff_z)
```

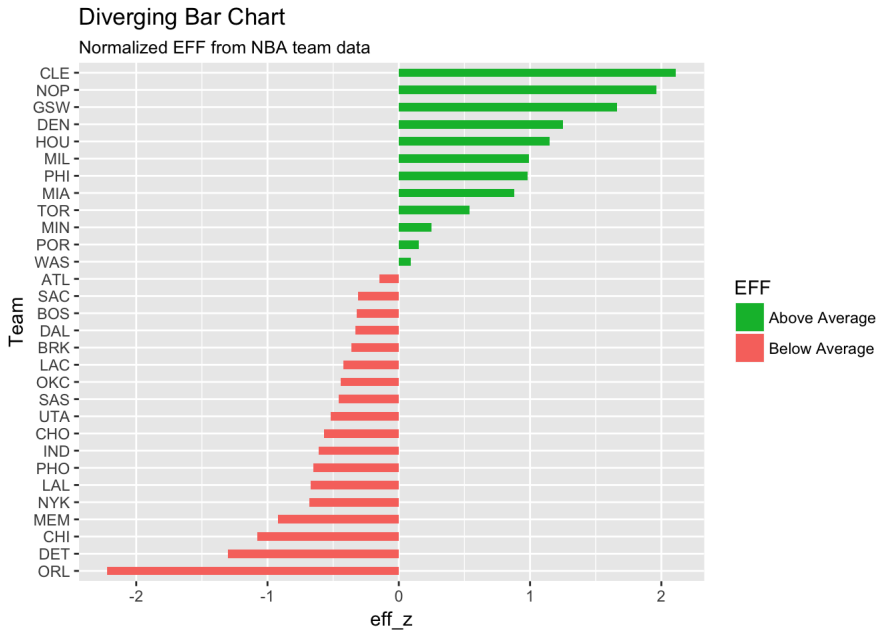
In order to retain such an order, I converted the column Team into a factor of 30 levels, one for each team.

```
team_EFF$Team <- factor(team_EFF$Team, levels = team_EFF$Team)
```

Finally, we can create a diverging bar chart! This is similar to the ones we created for HW3, but the difference is that the y-values are z-scores

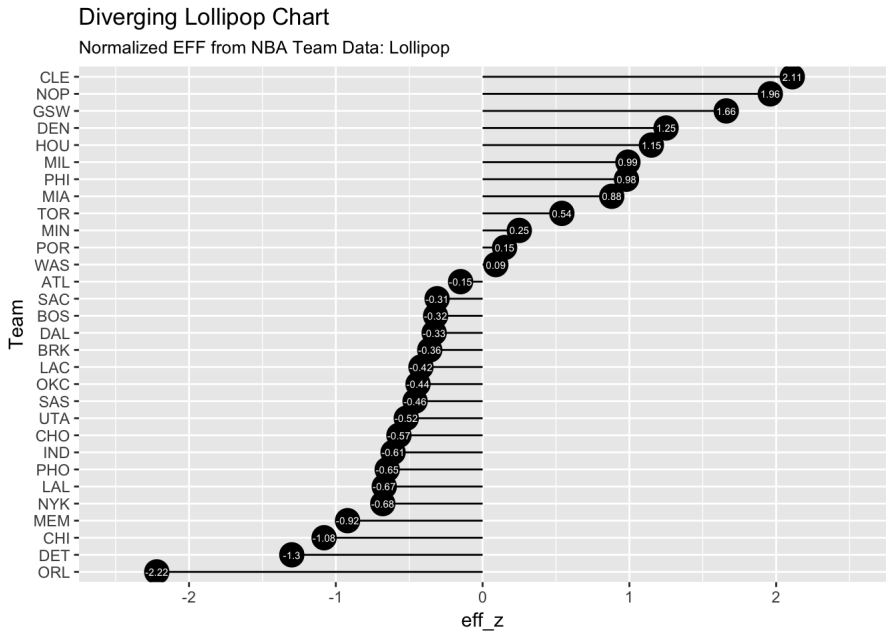
instead of average EFFs, allowing bars to point either in the positive or negative direction. And using the eff_type we created earlier, we can color-code the bars according to whether they are above or below average, and determine colors for the categories using “scale_fill_manual”.

```
ggplot(team_EFF, aes(x = Team, y = eff_z, label = eff_z)) +
  geom_bar(stat = 'identity', aes(fill = eff_type), width = .5)  +
  scale_fill_manual(name = "EFF",
                    labels = c("Above Average", "Below Average"),
                    values = c("above"="#00ba38", "below"="#f8766d")) + labs(subtitle = "Normalized EFF from NBA t
eam data",
       title = "Diverging Bar Chart") + coord_flip()
```



Another variation of this kind of plot is referred to as the diverging lollipop chart; you can perhaps see why from the graph below. This is created using geom_segment.

```
ggplot(team_EFF, aes(x = Team, y = eff_z, label = eff_z)) +
  geom_point(stat = 'identity', fill="black", size=6)  +
  geom_segment(aes(y = 0,
                   x = Team,
                   yend = eff_z,
                   xend = Team),
               color = "black") +
  geom_text(color = "white", size=2) +
  labs(title = "Diverging Lollipop Chart",
       subtitle = "Normalized EFF from NBA Team Data: Lollipop") +
  ylim(-2.5, 2.5) +
  coord_flip()
```



Yet another variation of this plot is the diverging dot plot, in which the plot points are color-coded as in the diverging bar chart example, and the lines are omitted, giving the viewer some sort of impression of a scatterplot.

```
ggplot(team_EFF, aes(x = Team, y = eff_z, label = eff_z)) +
  geom_point(stat='identity', aes(col = eff_type), size=6)  +
  scale_color_manual(name = "EFF",
                     labels = c("Above Average", "Below Average"),
                     values = c("above"="#00ba38", "below"="#f8766d")) +
  geom_text(color = "white", size=2) +
  labs(title = "Diverging Dot Plot",
       subtitle = "Normalized EFF from NBA team data: Dotplot") +
  ylim(-2.5, 2.5) +
  coord_flip()
```

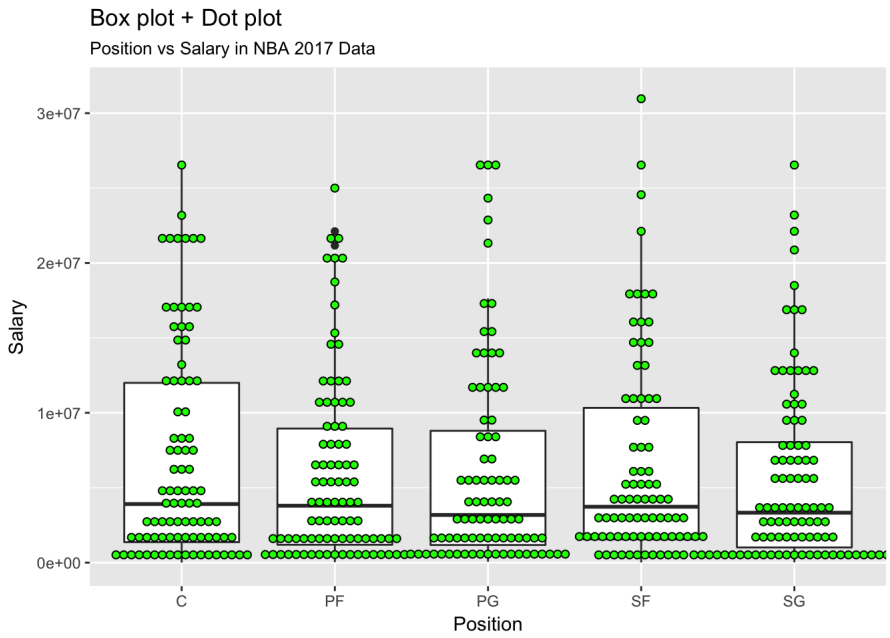
These clever manipulations of geom_bar and geom_plot are useful whenever you have data that you want to take the average of and compare each data point against the average, showing both positive and negative differences. Courtesy to this source.

Combining various plots (and variables) in ggplot2

Another wonder of ggplot is the ability to combine various basic plots together into one single graphic. An example of this is the combination of box plot and dot plot below, which depicts the salaries of the players grouped by their respective positions.

```
g <- ggplot(data, aes(Position, Salary))
g + geom_boxplot() +
  geom_dotplot(binaxis = 'y',
               stackdir = 'center',
               dotsize = .5,
               fill = "green") +
  labs(title = "Box plot + Dot plot",
       subtitle = "Position vs Salary in NBA 2017 Data",
       x = "Position",
       y = "Salary")
```

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

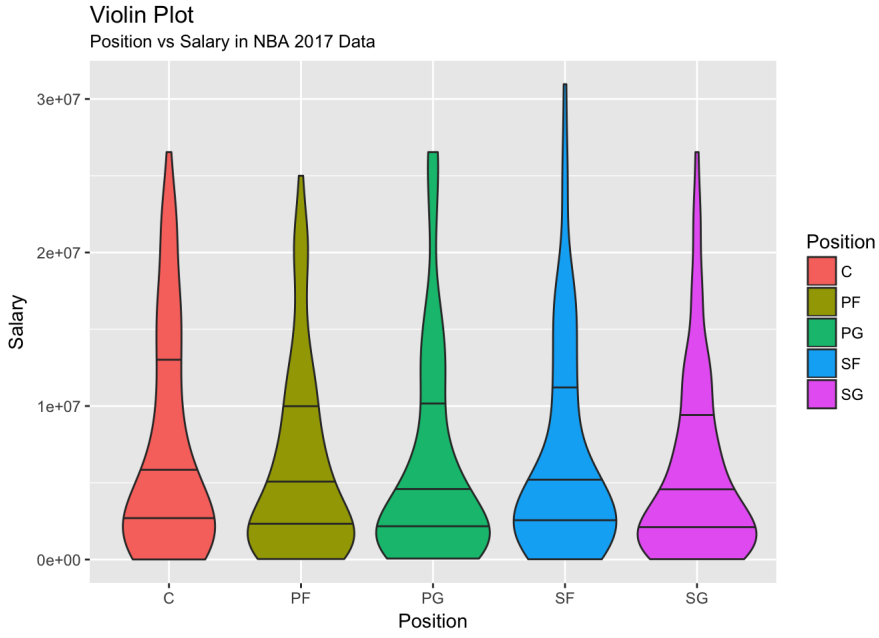


While this does effectively show the exact distribution of the data, I think it doesn't seem as aesthetically pleasing as you'd expect ggplot graphics to be. Fortunately, there's a more pleasant-looking ggplot that can be used: the violin plot! (As a practicing violinist myself, I am particularly drawn to this kind of plot.)

Essentially, the violin plot is a combination of the boxplot and the density plot. The density plot is mirrored against the vertical axis, creating the "sides" of the violin. So the fatter or skinnier the violin, the more or less data points there are within that interval. You can even use the 'draw-quantiles' parameter to mark the quantiles for each category, just like in a boxplot.

Here's the violin plot version of Position vs Salary:

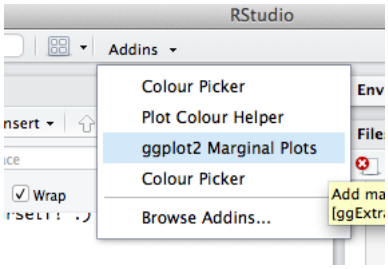
```
ggplot(data, aes(Position, Salary)) + geom_violin(aes(fill = Position), draw_quantiles = c(0.25, 0.5, 0.75)) + lab
s(title = "Violin Plot",
       subtitle = "Position vs Salary in NBA 2017 Data",
       x = "Position",
       y = "Salary")
```



Unfortunately, these look more like amateur pottery works than violins, but hey, if you have a more normally distributed set of data, you can witness the magic yourself! :)

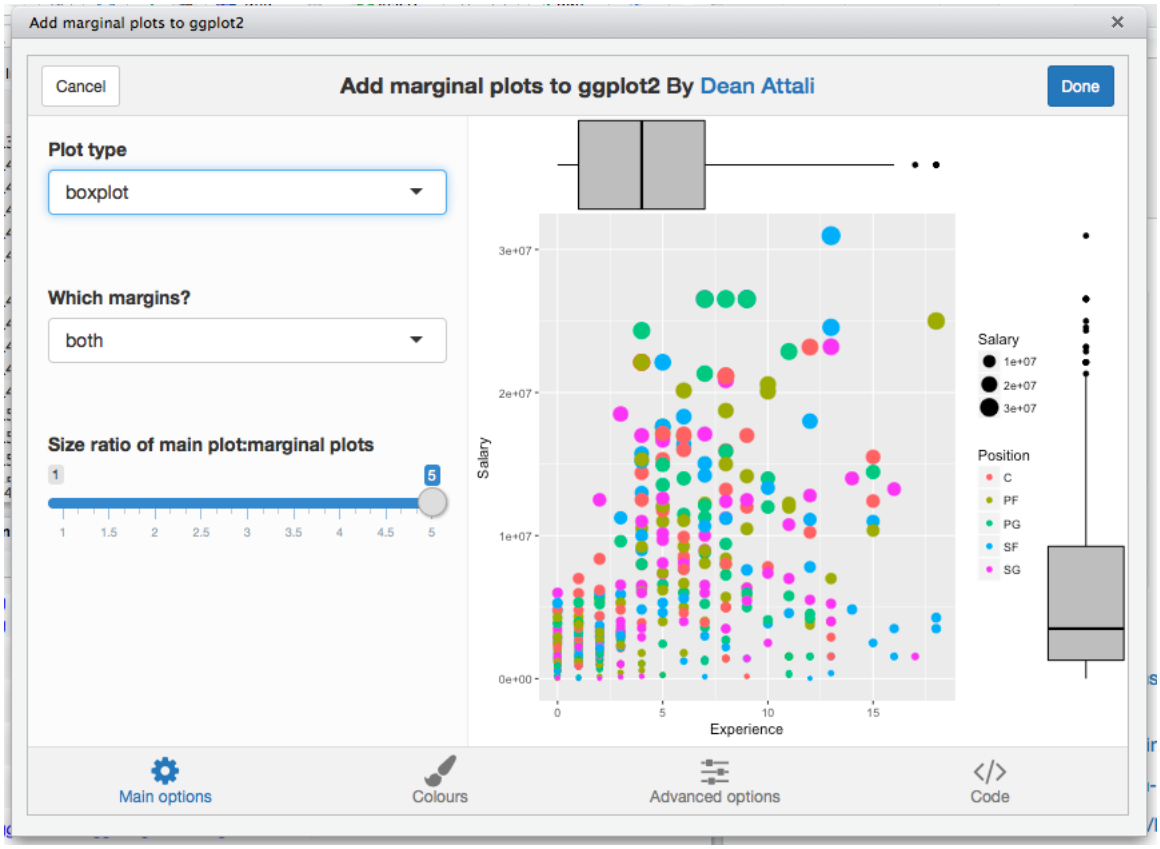
ggplot2 enhancer: ggExtra

Another example of combining plots involves the use of ggMarginal, a function that can be found within the ggExtra, a package that was created by Dean Attali in 2016. ggMarginal allows users to add histograms, boxplots, or density plots to the margins of a scatterplot in order to provide more information about the individual distributions of the variables used in the x and y axes of the scatterplot. Once the ggExtra package is installed, we can find the 'ggplot2 Marginal Plots' in the 'Addins' button located at the top of RStudio.



Addins has a new member!

After highlighting a certain ggplot command in our code, we can then select this 'ggplot2 Marginal Plots' option and customize whatever plots we wish to add to the margins of our scatterplot!



Customizing the marginal plots.

And once you click "Done", the code chunk will automatically be updated with the code for the enhanced plot!

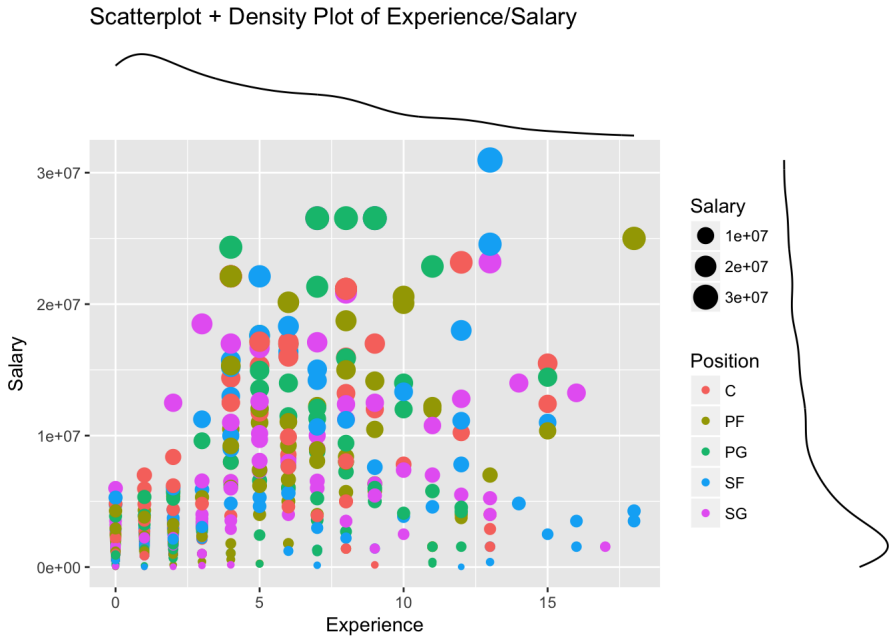
Here are the examples of ggMarginal, with a scatterplot of Experience and Salary, grouped by Position:

```
library(ggExtra)
p1 <- ggplot(data, aes(Experience, Salary, color = Position, size = Salary)) + geom_point() + ggtitle("Scatterplot
+ Boxplot of Experience/Salary")

ggExtra::ggMarginal(
  p = p1,
  type = 'boxplot',
  margins = 'both',
  size = 5,
  col = 'black',
  fill = 'gray'
)
```


```
p2 <- ggplot(data, aes(Experience, Salary, color = Position, size = Salary)) + geom_point() + ggtitle("Scatterplot + Density Plot of Experience/Salary")

ggExtra::ggMarginal(
  p = p2,
  type = 'density',
  margins = 'both',
  size = 5,
  col = 'black',
  fill = 'gray'
)
```



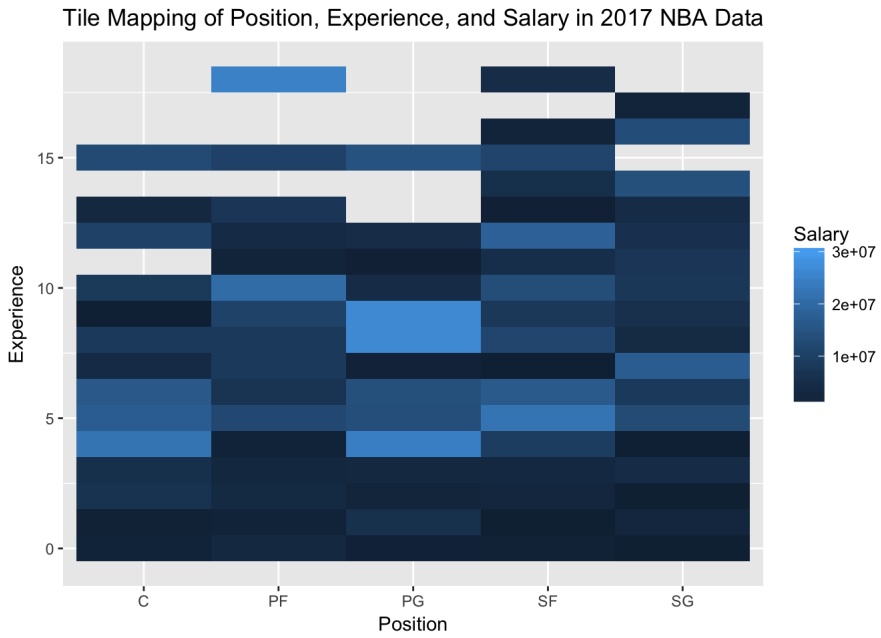
Three Variables: geom_raster, geom_contour, and geom_tile

My last tidbit involves ggplot's ability to map three variables in 2D space. We've done a little bit of that during lab using the function geom_density2d, but I'd also like to further introduce the concept of mapping 2D probability densities using ggplot. (Stat 134 nightmares intensify…)

For this last portion, I will be using the 'faithfuld' data set within R, which gives 2D probability density estimates of the 'faithful' data set, containing all the eruption times of the Old Faithful geyser and the waiting times between eruptions.

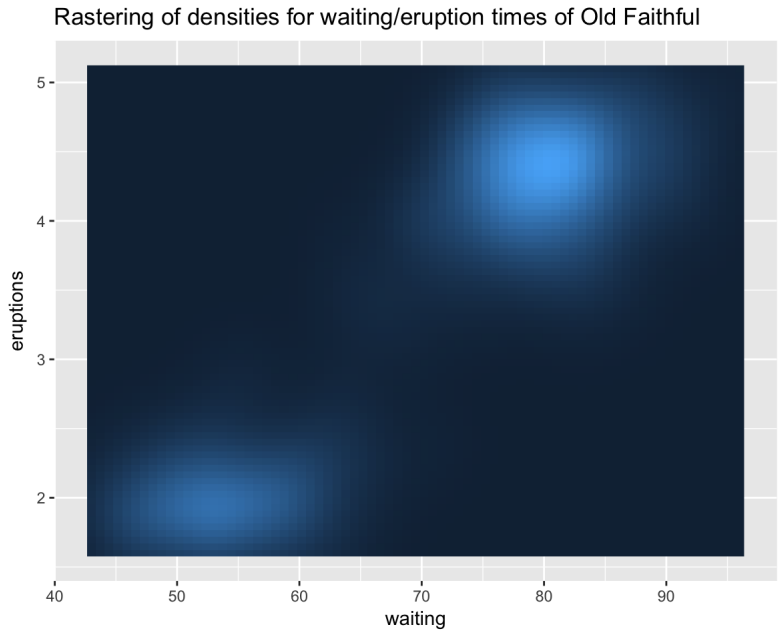
But first, just to get our feet wet in these functions using the NBA data we've been so familiar with all these weeks, here's an example of using geom_tile to plot the player's positions, experience, and salary all together. This shows the distribution of salaries for varying levels of experience and different positions, with the color shades of the tiles indicating varying salaries.

```
ggplot(data, aes(Position, Experience, Salary)) + geom_tile(aes(fill = Salary)) + ggtitle("Tile Mapping of Position, Experience, and Salary in 2017 NBA Data")
```



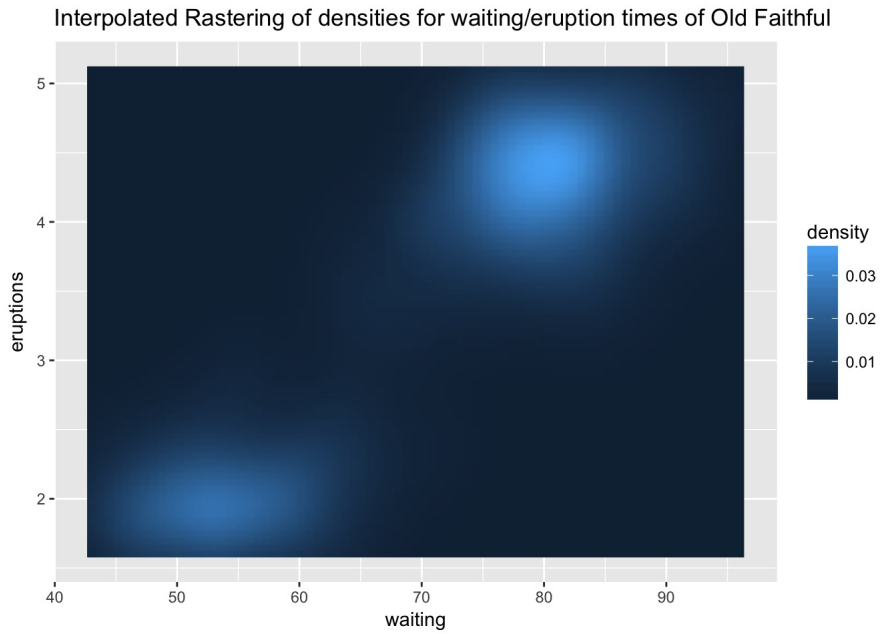
As the grey gaps above may show, these functions work best when the x and y axes of the data can be roughly spread throughout a rectangular grid, as will be shown through the plotting of the 'faithfuld' data. But in the following case, we will use geom_raster since it is much more efficient in rendering a graph of a large data set like 'faithfuld' (5000+ rows).

```
ggplot(faithfuld, aes(waiting, eruptions)) +
 geom_raster(aes(fill = density)) + ggtitle("Rastering of densities for waiting/eruption times of Old Faithful")
```



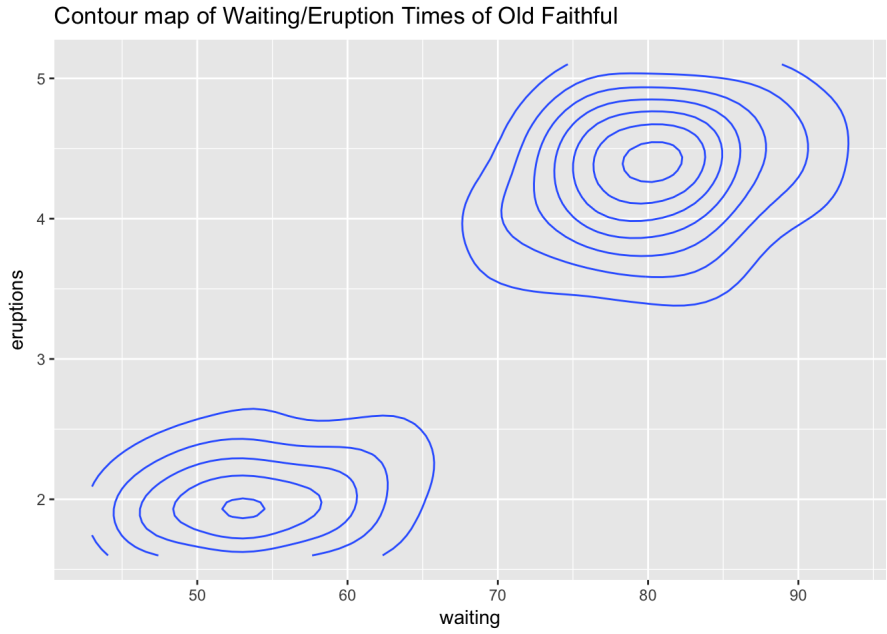
TRUE" to further smooth the density graph:

```
ggplot(faithfuld, aes(waiting, eruptions)) +
 geom_raster(aes(fill = density), interpolate = TRUE) + ggtitle("Interpolated Rastering of densities for waiting/e
ruption times of Old Faithful")
```



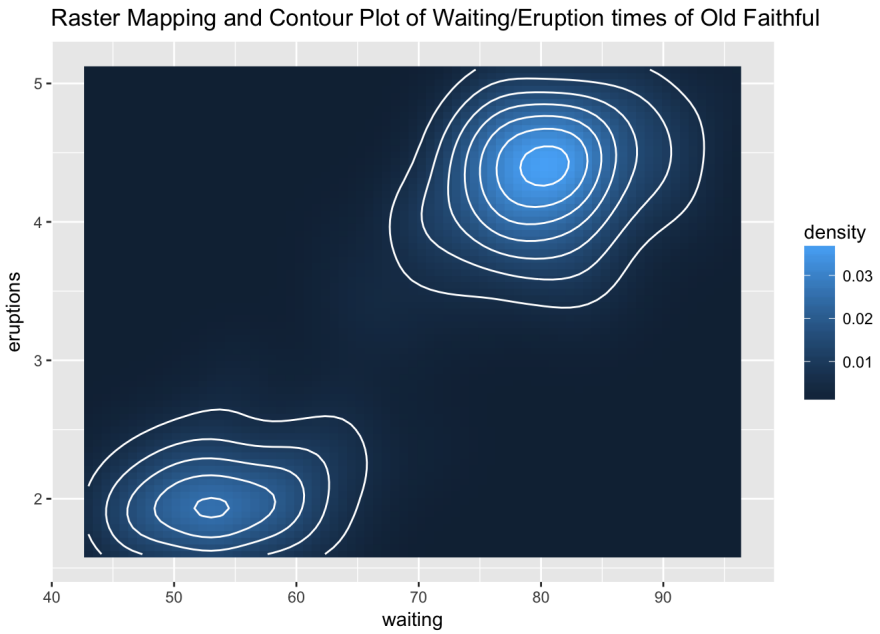
We can also use geom_contour to create clear outlines for the density contours:

```
ggplot(faithfuld, aes(waiting, eruptions, z = density)) + geom_contour() + ggtitle("Contour map of Waiting/Eruption Times of Old Faithful")
```



We can even combine the raster map and the contour map together! (Just make sure the colors are contrasting.)

```
ggplot(faithfuld, aes(waiting, eruptions, z = density)) + geom_raster(aes(fill = density)) + geom_contour(colour = "white") + ggtitle("Raster Mapping and Contour Plot of Waiting/Eruption times of Old Faithful")
```



Conclusion

By no means is this a comprehensive overview of the versatility of ggplot2, but I hope this post reinforces the idea that even just one subset of data can yield so many visual representations, depending on what type of visualization you consider best for the occasion. Among such visualizations are even options to combine various ggplot graphs together into a single image (histogram + scatterplot, boxplot + dotplot, coutour plot + raster map, etc.) to give the audience multiple facets or interpretations to a single data subset. In addition, statistical concepts from the data such as z-scores and probability distributions can also be represented and displayed through the plots.

Sources

http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

http://ggplot2.tidyverse.org/reference/geom_violin.html

http://www.r-graph-gallery.com/79-levelplot-with-ggplot2/

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

http://www.r-graph-gallery.com/277-marginal-histogram-for-ggplot2/

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

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

http://ggplot2.tidyverse.org/reference/geom_tile.html

http://ggplot2.tidyverse.org/reference/geom_contour.html