Additional Plots With ggplot2

Introduction

In class, we have studied how to use `ggplot2` to create simple graphs like scatterplots, boxplots, and histograms. In this post I will explore additional plots which can be implemented with `ggplot2` and features which can enhance some of the simpler plots. I will also explore additional packages which can be used in conjuction with `ggplot2` to create more advanced plots. It is important to understand these uses of ggplot2 to be able to decipher how to best visualize certain aspects of a chosen dataset based on what you would like to emphasize in the data. First, I will begin with expanding upon scatterplots by explaining the additional feature of encircling. I will then introduce new plots called jitter plots and count plots and which both provide different ways to deal with data containing overlapping points. Furthermore, I will explore the correlogram and a new package to allow us to easily display correlations within a dataset. I will also demonstrate a few variations of the pie chart which is one of the more visually unique graphs relying heavily on the use of color. Lastly, I will show how to implement two different types of line charts which are useful to visualize changes in quantitative data.

Scatterplots

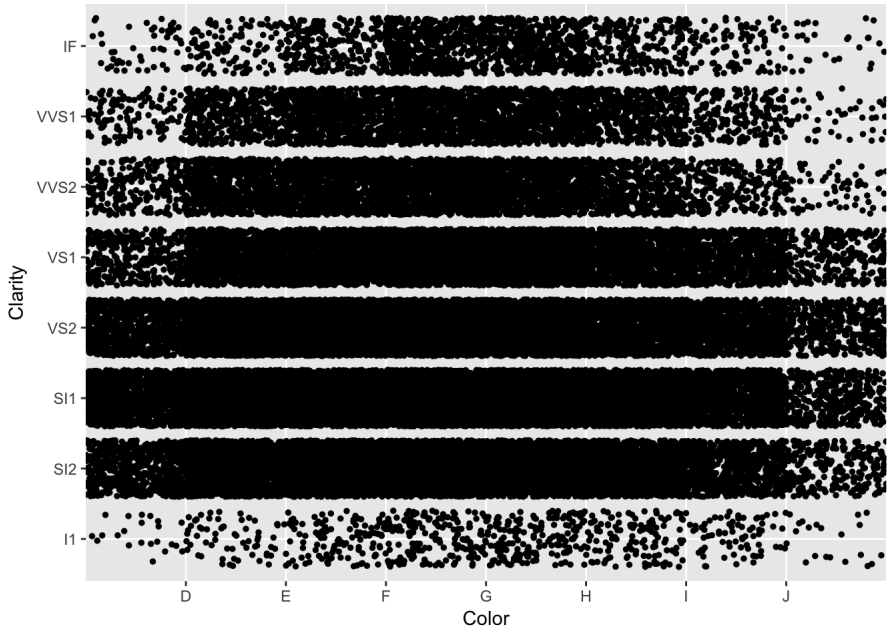
In class, we learned how to use `geom_point()` to create a scatterplot of our data. One of the features which can be applied to the scatterplot is an encircling of data. With `geom_encircle()` from the `ggalt` package, a circle can be drawn around select data points for emphasis. This circling can be heavily customized in terms of thickness, color, and size with inputs to the `geom_encircle()` function. Below is an example use of `geom_circle()` with the public diamonds dataset describing the relationship between the price, carat, and cut of the diamonds. In order to reproduce this graph, it is necessary to load the `ggalt` package as well as `ggplot2` with the first two commands shown. The `ggalt` package offers additional features to utilize with `ggplot2` such as coordinate systems, scales, statistical transformations, and fonts.

```
#Install the 'ggalt' package with install.packages('ggalt')

library(ggplot2)
library(ggalt)

data("diamonds", package = "ggplot2")
circle <- diamonds[diamonds$price > 1000 &
                     diamonds$price < 15000 &
                           diamonds$carat <= 5 &
                           diamonds$carat > 2, ]

#Scatterplot with encircle
ggplot(diamonds, aes(x=price, y=carat)) +
  geom_point(aes(col=cut)) +
  geom_encircle(aes(x=price, y=carat),
                data=circle,
                color="purple",
                size=2,
                expand=0.08) +
  labs(y="Carat",
       x="Price")
```

Jitter Pots and Count Plots

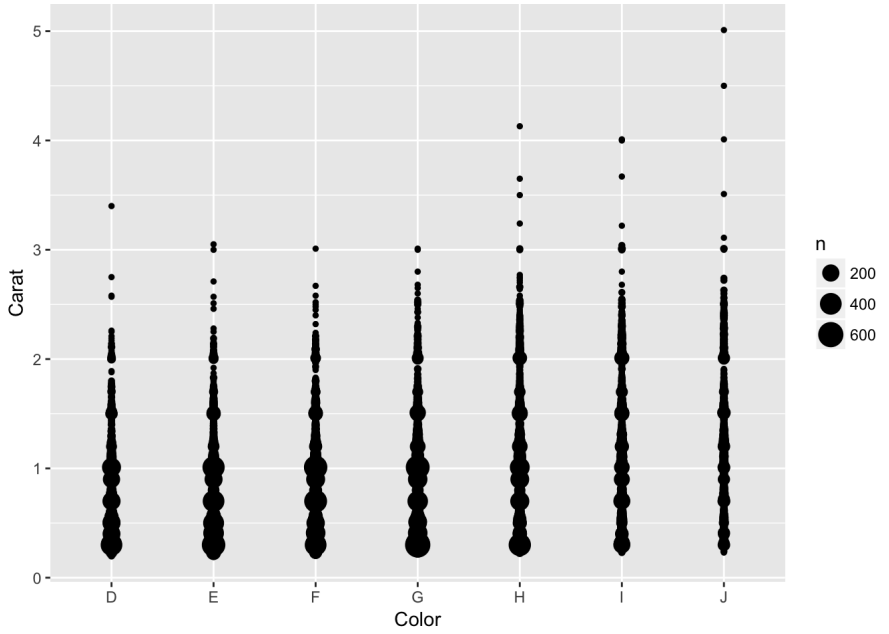
Another plot which is similar to the scatterplot is the jitter plot. The jitter plot is useful for dealing with data containing multiple overlapping points and ultimately causing a lack of visibility of all of the points in the graph. With `geom_jitter()`, the width parameter can be used to space out these points along the x-axis to ensure that all points are visible in the graph while the size parameter allows you to choose the size of the points. The code below demonstrates how `geom_jitter()` is used to space out overlapping points along the graph, demonstrating the relationship between the clarity and color of the diamonds from the public diamonds dataset.

```
library(ggplot2)

data("diamonds", package = "ggplot2")

#Jitter plot
graph <- ggplot(diamonds, aes(color, clarity))
graph + geom_jitter(width = 1, size=1) +
  labs(y="Clarity",
       x="Color")
```



One alternate method for visually indicating overlapping points is to utilize `geom_count()` which enlarges the size of the points to show that there are multiple points at specific locations. The larger points indicate that there are more overlapping points at that location. It is not necessary to pass in any arguments to the `geom_count()` function to produce a counts graph. An example of the relationship between color and carat of the diamonds from the diamonds dataset is shown below with overlapping points displayed larger than the others.

```
library(ggplot2)

data("diamonds", package = "ggplot2")

#Count plot
graph <- ggplot(diamonds, aes(color, carat))
graph + geom_count() +
  labs(y="Carat",
       x="Color")
```



Correlograms

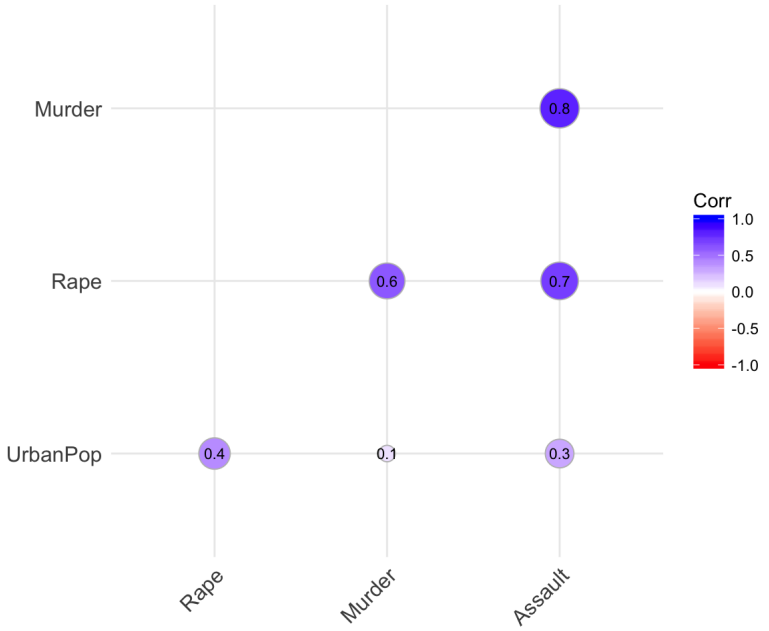
The correlogram is useful to indicate the correlation between variables. In order to implement this feature, it is necessary to load the `ggcorrplot` package as shown in the code below. With this package, the correlogram can be made with differing sized and colored points relating to how

large or small the correlation is. These points may also be labelled with the relevant correlation value. In the code below, the the `cor()` function has been applied to the public USArrests dataset and passed into the `ggcorrplot()` function which also takes in inputs to determine the size of the points and the labels of the correlation values on each of the points. This plot shows the correlations between murder, rape, assault, and the urban population in the United States.

```
#Install the 'ggcorrplot' package with install.packages('ggcorrplot')

library(ggcorrplot)
library(ggplot2)

data(USArrests)
corr <- round(cor(USArrests), 1)
corr
```

```
##          Murder Assault UrbanPop Rape
## Murder      1.0     0.8      0.1  0.6
## Assault     0.8     1.0      0.3  0.7
## UrbanPop    0.1     0.3      1.0  0.4
## Rape        0.6     0.7      0.4  1.0
```

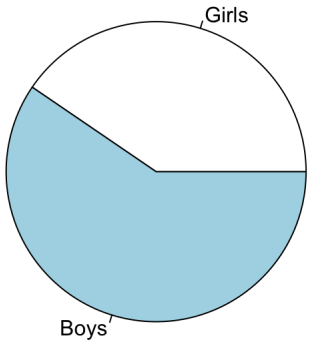
```
#Correlogram plot
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("red", "white", "blue"))
```



Pie Charts

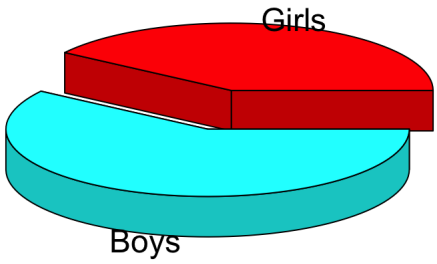
The pie chart is a method of visualizing data which contrasts more heavily to the previous examples in this post. With `ggplot2` we can use the `pie()` function to easily produce a circular pie chart. The `pie()` function takes in a vector containing the area of each of the portions of the chart as well as the corresponding labels, if applicable. This is a convenient way to show the overall composition of data and can be more effective when different colors are used to distinguish different variables. An example visualization of the amount of girls and boys enrolled in a class is shown in the graph below.

```
library(ggplot2)

slices <- c(17, 25)
lbls <- c("Girls", "Boys")
pie(slices, labels = lbls)
```



It is also possible to create 3D versions of these pie charts using the `pie3D()` function from the `plotrix` package. The `pie3D()` function takes in the same parameters as the `pie()` function as well as an explode parameter denoting how far apart the sections of the chart should appear from each other in the 3D visualization. The code to load the package and implement the 3D pie chart with the data from the previous example is shown below.

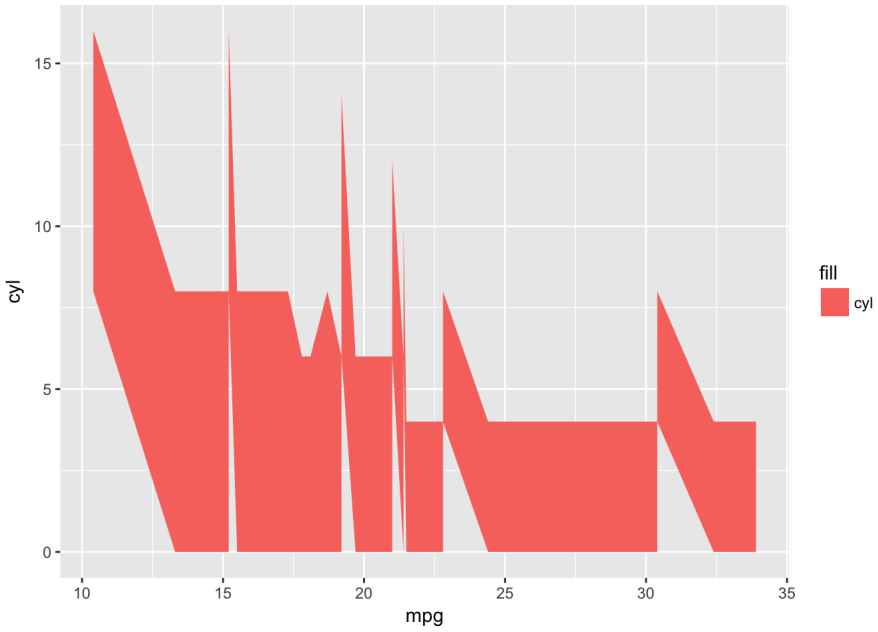
```
#Install the 'plotrix' package with install.packages('ggalt')

library(plotrix)
slices <- c(17, 25)
lbls <- c("Girls", "Boys")
pie3D(slices,labels=lbls,explode=0.2)
```



Line Charts

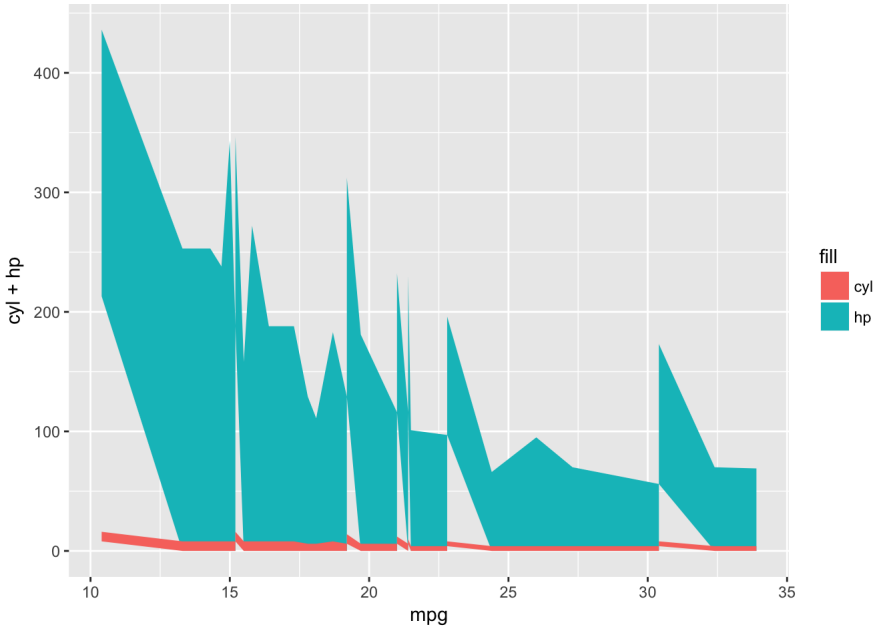
To create slightly more advanced line charts than those we have seen before, it is possible to fill the area under the lines with color. This is useful when we would like to display how a quantity changes over time. In addition to `geom_line()`, we also use `geom_area()` to produce the colored region under the lines in the graph. This is done by passing in a value to the fill parameter of the `geom_area()` function. An example describing the relationship between miles per gallon and number of cylinders from the public mtcars dataset is shown in the code below.

```
library(ggplot2)

data(mtcars)

#Line Chart
ggplot(mtcars, aes(x=mpg)) +
  geom_area(aes(y=cyl, fill="cyl"))
```



To expand upon the line chart we have just created, it is possible to implement a stacked line chart which displays more than one line on the same graph. The regions below the lines are colored as in the previous area chart but the fill color below the topmost line stops when it reaches the lower line. We can implement this feature by using two `geom_area()` functions. One function is implemented as in the previous example and one function takes in the sum of two values and a different value for the fill parameter. This is useful to visualize the change of more than one quantity over time in order to observe their similarities and differences. Below is an example of the relationship between miles per gallon, number of cylinders, and gross horsepower from the public mtcars dataset.

```
library(ggplot2)

data(mtcars)

#Stacked line chart
ggplot(mtcars, aes(x=mpg)) +
  geom_area(aes(y=cyl+hp, fill="hp")) +
  geom_area(aes(y=cyl, fill="cyl"))
```



Conclusion

For data visualization, there are countless ways in which the programmer may choose to display their data. While there may be a number of different ways to show certain relationships in a dataset, I feel that it is important to be knowledgeable about the many different functions of packages like `ggplot2` in order to choose the visualizations which best convey the intended message to the viewer. Moreover, since some of the same plots and charts may be implemented in different ways, mastery of data visualization packages in R allows the programmer to decipher which method will help them to implement the chart most efficiently. Therefore, it is necessary to understand the many possibilities of the `ggplot2` package as well as the packages which can be used in conjunction with `ggplot2` like `ggalt`, `ggcorrplot`, and `ggplotrix` as we have explored in this post.

Sources

1. https://www.rdocumentation.org/packages/datasets/versions/3.4.1/topics/mtcars

2. http://ggplot2.tidyverse.org/reference/diamonds.html
3. https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/USArrests.html
4. http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
5. https://cran.r-project.org/web/packages/plotrix/plotrix.pdf
6. https://www.statmethods.net/graphs/pie.html
7. http://www.sthda.com/english/wiki/r-built-in-data-sets#iris
8. https://cran.r-project.org/web/packages/ggalt/index.html
9. http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2