Data Visualization—more about ggplot2

Chuchu Gao

```
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)
```

Introduction

In data analysis, there is a certain procedure to analyze the data and come up with a conclusion in a certain topic that we are interested in. Firstly, we collect our raw data and do some data cleaning. Secondly, we need to observe our data and come up with a hypothesis based on our observation. Thirdly, we need to test our hypothesis using the appropriate hypothesis testing mothod. Last but not the least, after we get our conclusion, we need to present what we believe to our audience or report readers. Data visualization is one of the necessary tools for us to archieve the analysis.

We have talked about two very commonly used ways to present our data; the plot() function from "base" library, and the "ggplot2" library. These two could be very helpful when we try to explore and present our data. According to Selva Prabhakaren in "Top 50 ggplot2 Visualizations- The Master List", an effective chart should meet the following criteria.

- Conveys the right information without distorting facts
- Is simple but elegant. It should not force you to think much in order to get it.
- Aesthetics supports information rather that overshadow it.
- Is not overloaded with information.

I would like to dive into ggplot2 since there are now more and more people using it instead of plot(). I will also be showing some interesting examples of the functions that we have not touched yet.

Layers in ggplot2

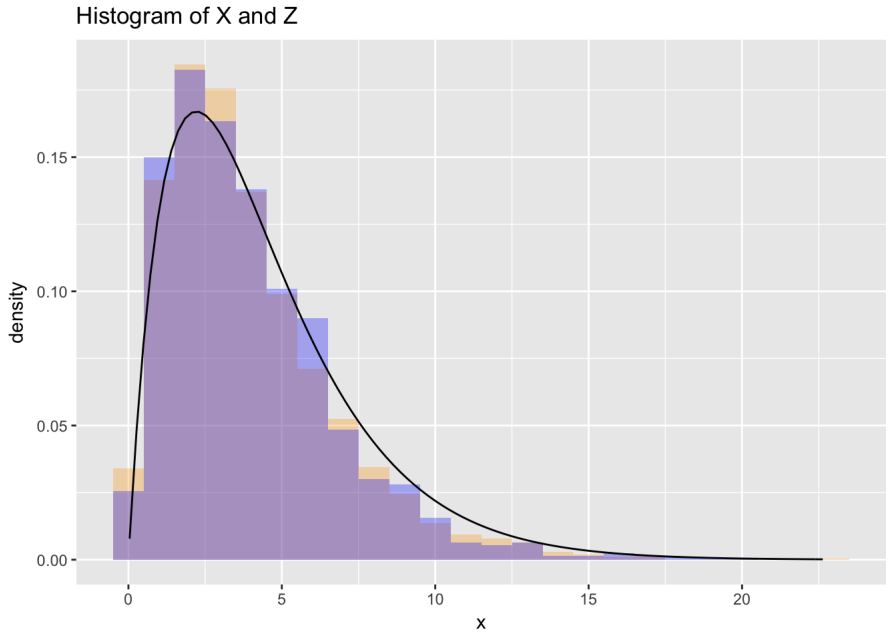
First, as many of us may have noticed that plot() creates a chart as one layer. We modify the graph by changing what we put inside plot(), while ggplot2 actually plots graphs with multiple layers. According to Hadley Wickham in "Build a plot layer by layer", one of the advantages of ggplot2 is that it allows us to build up a layer at a time and form a complicated plot. Each layer can even use a different dataset and have a different aesthetic mapping so that we can create far more sophisticated plots that display data from multiple sources.

For example, I had three large data sets shown in below, In the graph, I am going to create a graph using all of three data sets. In another word, the datasets I use for the histograms and the curve are independent.

```
#following is my sampling result and calculation of the testing statistics. The concept is from stat135, and you c
ould directly see the next chunk of code for my graph.
set.seed(123)
findts <- function(p){
  sample <-rbinom(1000,5,p)
  observations <- as.vector(table(sample))
  p_mle <- mean(sample)/5
  prob <- function(x){
  b <- choose(5,x)*(p_mle^x ) * ((1-p_mle)^(5-x))
}
  i <- c(prob(0),prob(1),prob(2),prob(3),prob(4),prob(5))
  expectations <- chisq.test(observations, p=i)$expected
  ts <- 2 * sum(observations * log(observations/expectations))
  ts
}

x <- c(replicate(2000, findts(p=0.4)))
X <- data.frame(x)
y <- c(replicate(2000, findts(p=0.6)))
Y <- data.frame(y)
z <- c(replicate(500, findts(p=0.4)))
Z <- data.frame(z)
dat <- mutate(X,Y)
```

```
#In this graph, I made a comparison between my dataset X and Y, where the histogram of .
ggplot()+geom_histogram(data=X,aes(x, y=..density..), binwidth = 1, alpha=0.3,fill="orange")+labs( title="Histogra
m of X and Z")+geom_histogram(data=Y ,aes(Y, y=..density..), binwidth = 1, alpha=0.3,fill="blue")+stat_function(da
ta=Z,fun = dchisq, n=101,args = list(0.4, df=4), color="black")
```



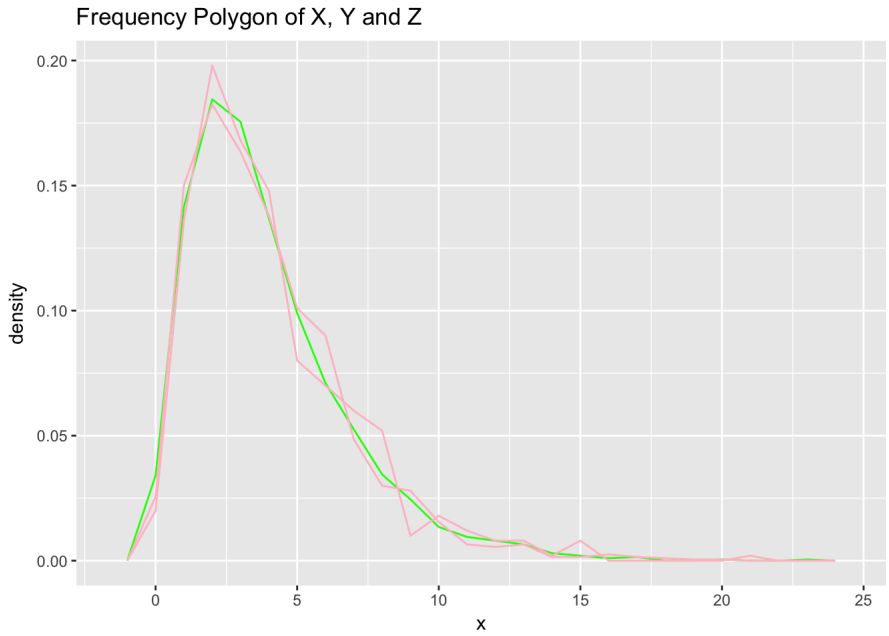
This is one of the illustration of ggplot2 working with multiple sources of data. From this example, we can see that comparing with “plot()”, ggplot2 works better when we have complicated data. This example just added two layers to my original data set X. In real life, we can add more layers just by simply “+” another function if needed.

More about the grammatical elements

Next I would like to explore some grammatical elements. In previous homework and labs, we practice with the three essential grammatical elements: Data, Aesthetics and Geometrics, and one optional grammatical element: facets. There are three more optional functions that could also be very helpful when we use ggplot2 later in the real life. They are “statistics”, “coordinates” and “themes”. “Statistics” is the representations of our data to aid understanding, binning, smoothing, descriptive and inferential. In my previous graph, I have include a descriptive Chi-square statistics curve using data set Z. “Coordinates” is the space on which the data will be plotted. For example, we can choose Cartesian, fixed, polar, or limits. “Themes” is all non-data ink.

We have practiced using ggplot2 to graph histogram, scatterplot, loess smooth, boxplot and etc. I would like to mention couple more types of graphs. (For more data profiling related information, you could visit “R Graphics: Introduction to ggplot2”, from UCLA Institute For Digital Research and Education)

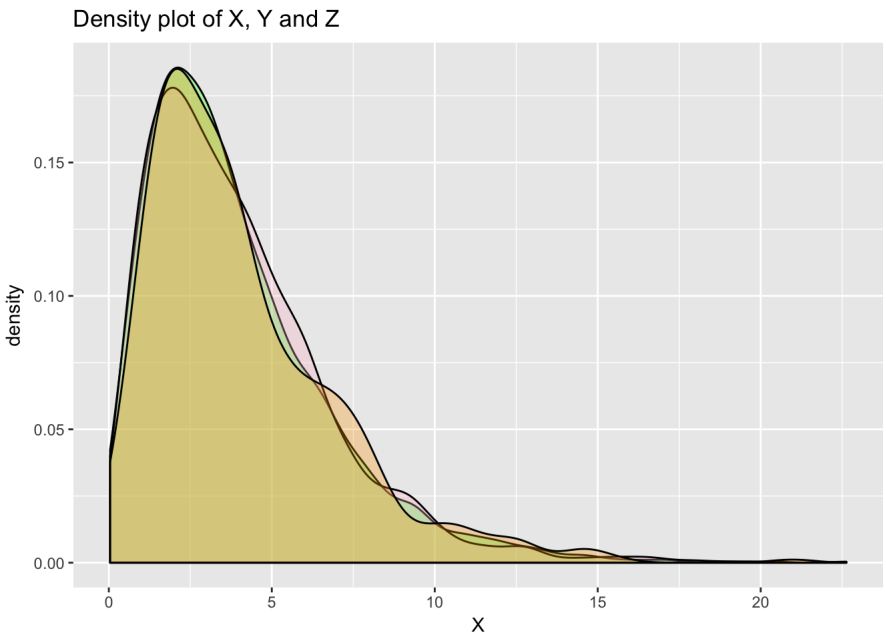
```
#1) Frequency polygon
ggplot()+labs(title="Frequency Polygon of X, Y and Z")+geom_freqpoly(aes(x, y=..density..),binwidth=1, color="gree
n")+geom_freqpoly(aes(Y, y=..density..),binwidth=1, color="pink")+geom_freqpoly(aes(Z, y=..density..),binwidth=1,
color="pink")
```



According to David Lane, frequency polygons are used to understand the shapes of distributions, they are very similar to histograms, however they are very helpful for comparing different data sets and displaying cumulative frequency distributions.

```
#2) Density plot
ggplot()+labs(title="Density plot of X, Y and Z")+geom_density(aes(X, y=..density..), alpha=0.3, fill="green")+geo
m_density(aes(Y, y=..density..), alpha=0.3, fill="pink")+geom_density(aes(Z, y=..density..), alpha=0.3, fill="oran
ge")
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
```



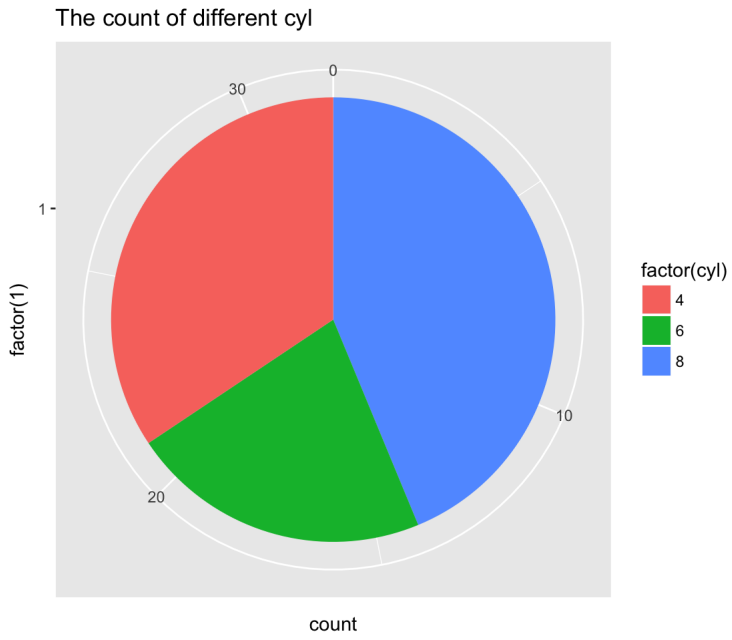
A Density Plot visualizes the distribution of data over a continuous interval or time period. The peaks of a Density Plot help display where values are concentrated over the interval. An advantage Density Plots have over Histograms is that they're better at determining the distribution shape because they're not affected by the number of bins used (each bar used in a typical histogram).

In addition to the basic grammatical elements we have played with. I would like to show more examples using Coordinates and Theme.

Coordinates.

```
#A pie chart = stacked bar chart + polar coordinates, mtcars is one of the dataset installed in R
pie <- ggplot(mtcars, aes(x = factor(1), fill = factor(cyl))) +
 geom_bar(width = 1) + labs(title="The count of different cyl")
pie
```

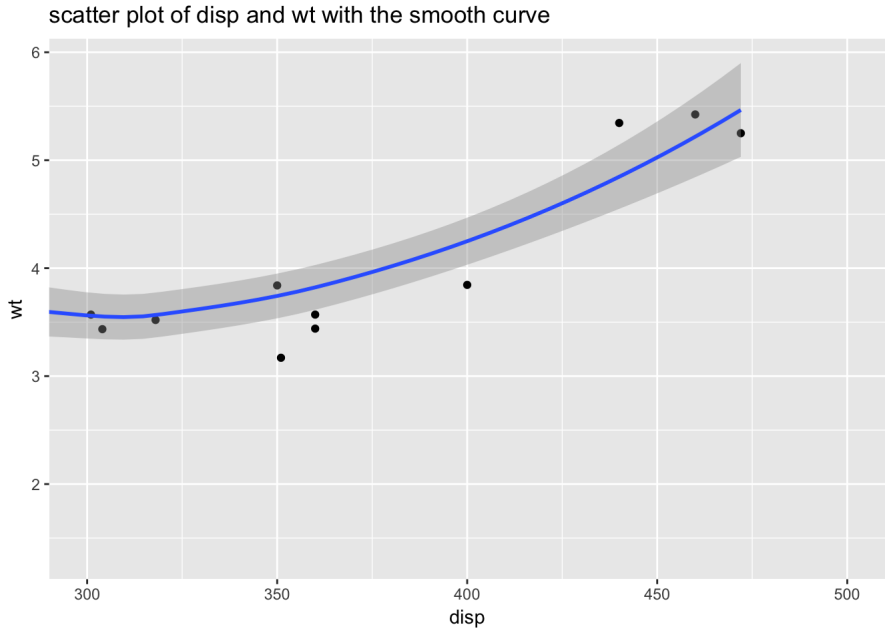

```
pie + coord_polar(theta = "y")+labs(title="The count of different cyl")
```



Inside the “coord_polar()” we set theta equals to the variable to map angle to. We also can set “start” and “direction”, where set “start” to offset of starting point from 12 o’clock in radians, “direction = 1” is clockwise, “direction=-1” is anticlockwise. The default set is start = 0, direction = -1.

```
# mtcars is a dataset installed in R
p <- ggplot(mtcars, aes(disp, wt)) +
  geom_point() +
  geom_smooth() +coord_cartesian(xlim = c(300, 500))+labs(title= "scatter plot of disp and wt with the smooth curv
e")
p
```

```
## `geom_smooth()` using method = 'loess'
```



Cartesian Coordinate can be used to zoom the plot display. I set x-axis in range of (300,500) by using xlim, we can do the same thing for y-axis. Another function to mention is “expand”, if it equals to TRUE, the default, it adds a small expansion factor to the limits to ensure that data and axes do not overlap. If it is False, limits are taken exactly from the data or xlim/ylim.

These are just a part of applications of Coordinates. More about Coordinates can be viewed Cartesian and Polar

Theme.

Themes is all non-data ink. We can choose different backgrounds for our graphs by working with their theme. The features that can be altered are the plot background elements, plot margin, major and minor grid, axis title, text and ticks. For many of the cases in real life, we want to present our graphs in a more eyes appealing way, we could change the color of background, the grid lines, the margin of graph and the size of girds to achieve it.

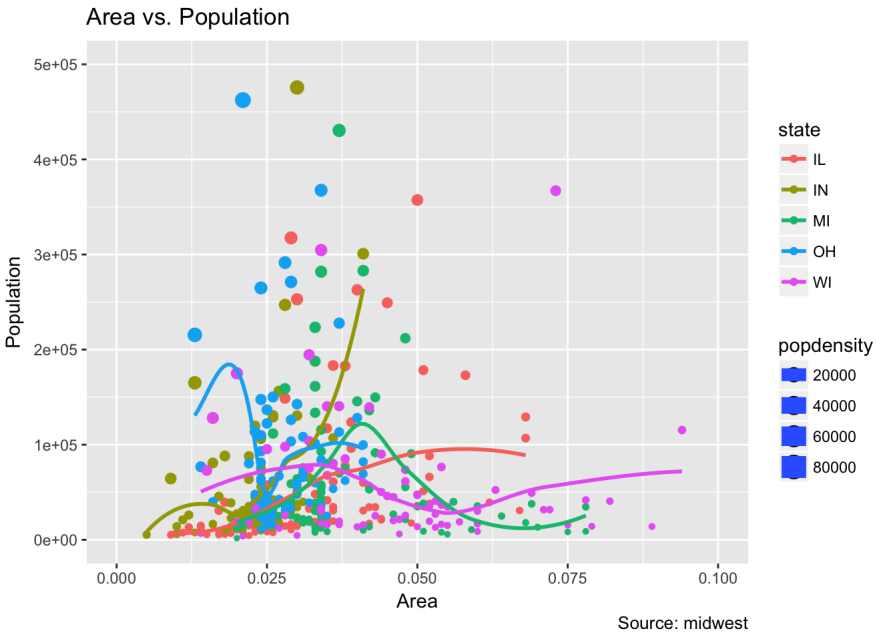
The default theme of ggplot2 is theme_gray. there are also some other pre-made theme. For example, theme_bw changes background color to white. theme_classic changes the theme to a classic looking with x, y axis lines and no gridlines. Furthermore, we can also design our own theme. The following are some interesting examples.

```
# oringinal graph
gg <- ggplot(midwest, aes(x = area, y = poptotal,
                          color = state, size = popdensity)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  xlim(0, 0.1) +
  ylim(0, 500000) +
  labs(title = "Area vs. Population",
       x = "Area",
       y = "Population",
       caption = "Source: midwest")
gg
```

```
## `geom_smooth()` using method = 'loess'
```

```
## Warning: Removed 15 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```

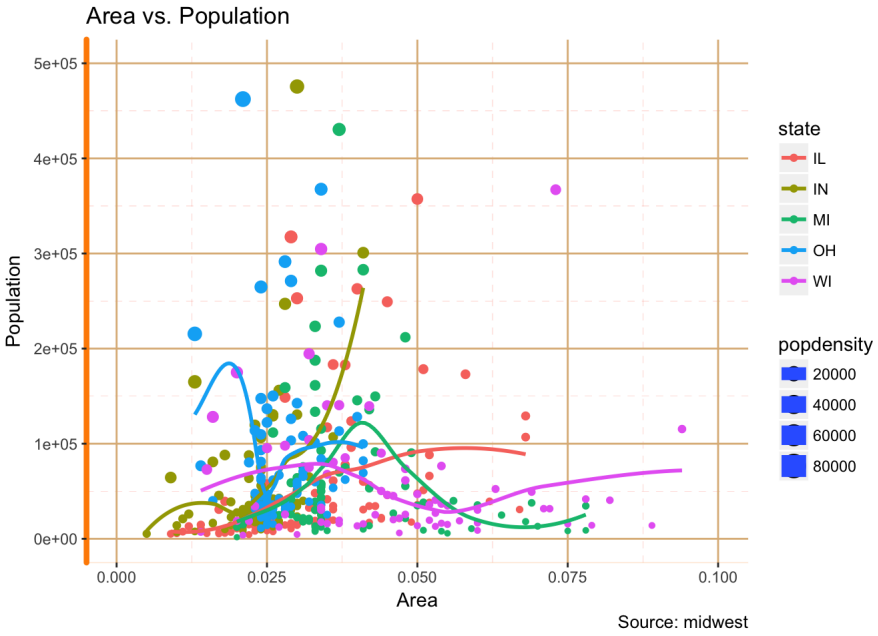



```
#1)change the plot background
gg +
  theme(
    panel.background = element_rect(fill = 'white'),
    panel.grid.major = element_line(color = "burlywood", size = 0.5),
    panel.grid.minor = element_line(
      colour = "tomato",
      size = .05,
      linetype = "dashed"
    ),
    panel.border = element_blank(),
    axis.line.x = element_line(
      colour = "darkorange",
      size = 0.05,
      lineend = "butt"
    ),
    axis.line.y = element_line(color = "darkorange",
                               size = 1.5)
  )
```

```
## `geom_smooth()` using method = 'loess'
```

```
## Warning: Removed 15 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```

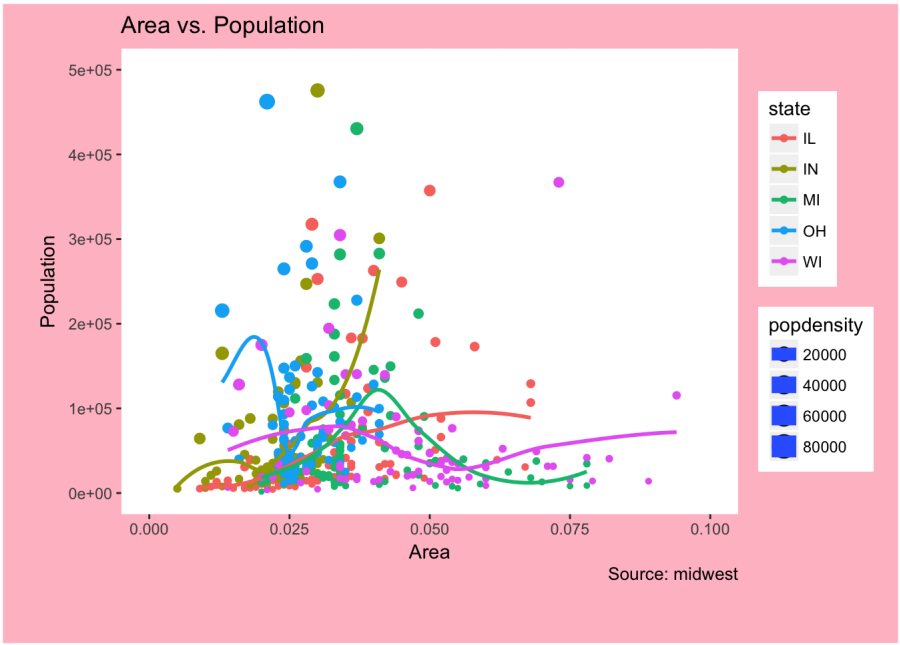



```
#2)Change plot margin
gg +
  theme(panel.background = element_rect(fill = 'white '),
    plot.background = element_rect(fill = "pink"),
        plot.margin = unit(c(0.1, 0.2, 0.5, 0.3), "in")) +
  labs(title = "Area vs. Population")
```

```
## `geom_smooth()` using method = 'loess'
```

```
## Warning: Removed 15 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```



For more information, I encourage you to check "Graphing tips for ggplot2 (life)" In real life, data could get very complicated. Theme could be very helpful when we are trying to show the characteristics the data.

Takeaways:

We have various ways to present the information, and our goal is to find the most efficient way. As I mentioned before, there are three basic grammatical elements and four optional grammatical elements available in ggplot2. These elements construct the frame of graphs. It is the programmer to decide which function to use and the final appearance of the graphs. The purpose to use these function is to get the most effective graphs possible.

List of My Reference

1)"Top 50 ggplot2 Visualizations- The Master List"

2)"Build a plot layer by layer"

3)"R Graphics: Introduction to ggplot2", UCLA Institute For Digital Research and Education

4)"Graphing tips for ggplot2 (life)"

5)"Cartesian"

6)"Polar"

7)"Frequency Polygons" by David M. Lane

8)"Visualizing data with ggplot2"

9)"Density Plot"