Exploring More of ggplot2

Juliana Gilman

December 2, 2017

Introduction

This post will primarily cover ggplot2 graphs and applications not discussed or gone over in Stat 133. Ggplot2 is an extremely useful package for visualizing data, so I felt that it was important to learn more about the wide range of graphs that it is able to produce. In order to ensure reproducibility of the graphs and data frames produced within this post, I have used set.seed() whenever random numbers are generated.

Applications

Before we delve into any examples, let's make sure that we have loaded ggplot2 into our Rmd file. If you haven't yet installed ggplot2, make sure to do so in your console!

```
# Load ggplot2

library(ggplot2)
```

For our first ggplot2 example, we'll be going over a density plot. But *first*, we'll need to go over what data we'll be using!

For this first example, we'll be using data provided by R. If you are interested in looking at a list of data sets provided by R, you can simply type `data()` into your console. The data set that we'll be using for this first example is listed as "UCBAdmissions." In order to obtain this data, convert it into a data frame, and assign it to an object you will want to run the following code:

```
UCB <- as.data.frame(UCBAdmissions, stringsAsFactors = FALSE)
```

Let's look at the first few columns of the data frame.

```
head(UCB)
```

```
##      Admit Gender Dept Freq
## 1 Admitted   Male    A  512
## 2 Rejected   Male    A  313
## 3 Admitted Female    A   89
## 4 Rejected Female    A   19
## 5 Admitted   Male    B  353
## 6 Rejected   Male    B  207
```

Because I intend to demonstrate how to generate a density graph with overlapping density plots as the first example of this post, we will need data for more than one university. For the purpose of this example, We will be creating data frames for UCLA, UCI, and UCSD from the data frame for UCB.

In order to generate new admissions data for UCI, UCLA, and UCSD we will be altering the frequency ("Freq") column data that we will extract from the original UCB data frame. The code below will serve as the new entries for the column Freq in the UCI, UCLA, and UCSD data frames that we will create next.

```
#As can be seen below, for all three universities we have created two vectors, one representing the number of stud
ents admitted and the other the number of students rejected.

# Number of students admitted to UCI
set.seed(10)
UCI_admitd_freq <- floor(runif((length(UCB$Freq))/2, min = 20, max = 40))

# Number of students rejected by UCI
set.seed(20)
UCI_reject_freq <- floor(runif((length(UCB$Freq)/2), min = 500, max = 600))

# Number of students admitted to UCLA
set.seed(100)
UCLA_admitd_freq <- floor(runif((length(UCB$Freq))/2, min = 5, max = 15))

# Number of students rejected by UCLA
set.seed(50)
UCLA_reject_freq <- floor(runif((length(UCB$Freq))/2, min = 300, max = 800))

# Number of students accepted to UCSD
set.seed(150)
UCSD_admitd_freq <- floor(runif((length(UCB$Freq))/2, min = 100, max = 500))

# Number of students rejected by UCSD
set.seed(25)
UCSD_reject_freq <- floor(runif((length(UCB$Freq))/2, min = 50, max = 70))
```

```r
# Here we are creating data frames for UCI, UCLA, and UCSDusing the UCB data frame and then modifying the "Freq""
column of each of those three data frames by using the vectors we created in the code chunk above.

# UCI data frame
UCI <- UCB
UCI$Freq[UCI$Admit == "Admitted"] <- UCI_admitd_freq
UCI$Freq[UCI$Admit == "Rejected"] <- UCI_reject_freq

#UCLA data frame
UCLA <- UCB
UCLA$Freq[UCLA$Admit == "Admitted"] <- UCLA_admitd_freq
UCLA$Freq[UCLA$Admit == "Rejected"] <- UCLA_reject_freq

#UCSD data frame
UCSD <- UCB
UCSD$Freq[UCSD$Admit == "Admitted"] <- UCSD_admitd_freq
UCSD$Freq[UCSD$Admit == "Rejected"] <- UCSD_reject_freq
```

```r
# Add a column to each data frame called "University" that lists the name of the university associated with that d
ata frame

UCB$University <- rep("UCB", length(UCB$Freq))

UCLA$University <- rep("UCLA",length(UCLA$Freq))

UCI$University <- rep("UCI",length(UCI$Freq))

UCSD$University <- rep("UCSD",length(UCSD$Freq))

# Create 1 data frame called "universities" that contains the data from all 4 data frames by merging them

merge1 <- merge(UCB,UCLA, all = TRUE)
merge2 <- merge(UCI,UCSD, all = TRUE)
universities <- merge(merge1, merge2, all = TRUE)
```

Density Plots

Ok, great! Now that we have our data, we can move onto the actual density plot. Our goal is to make a density plot for the number of admitted females at each university. In order to be able to obtain a density plot for this subset group, we'll use "dplyr" functions to perform some data frame manipulation.

```r
# Let's first use the library() function to load the "dplyr" package. Make sure you have aleady typed install.pack
ages("dplyr") into your console!

library(dplyr)
```

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

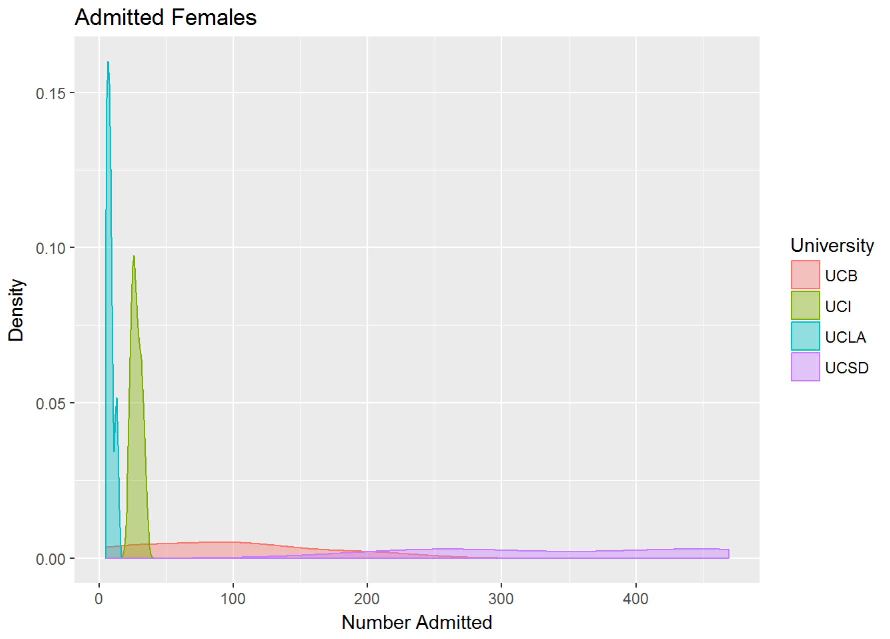
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```r
admitted_females <- filter(universities, Gender == "Female", Admit == "Admitted")
```

Note that in order to generate a density plot, we must use **geom_density()**.

```r
# Generate a density plot

ggplot(admitted_females, aes(Freq, color = University, fill = University)) + geom_density(alpha = 0.40) + ggtitle(
"Admitted Females") + labs(x= "Number Admitted", y = "Density")
```

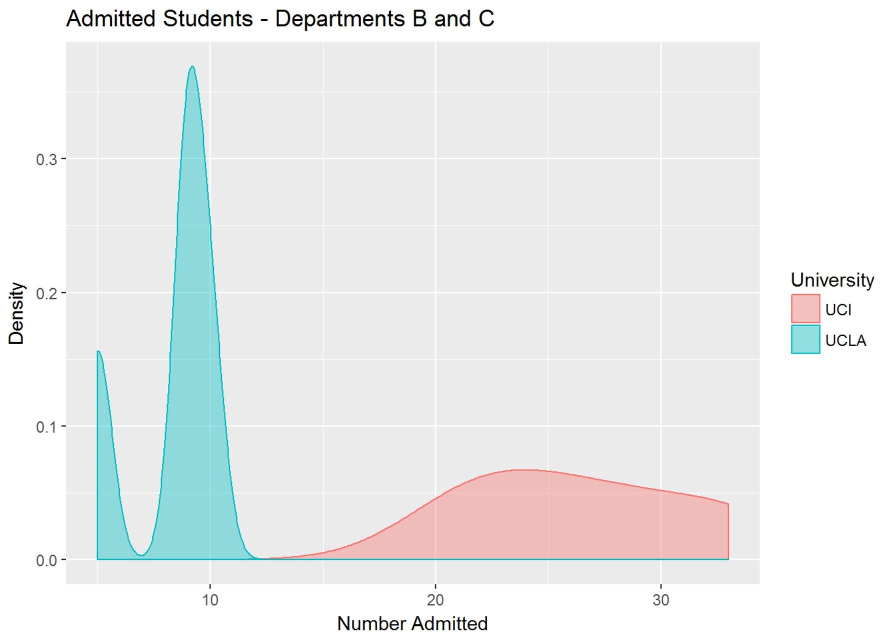


VOila! We got the graph we wanted.

From the density plot, we can see that only a very small number of females are admitted to UCLA, while UCSD admits the greatest number of females. One important note to keep in mind: It is tempting to interpret this density plot as demonstrating that UCLA has a lower acceptance rate that UCSD; however, we cannot technically say this by just looking at this graph since the graph does *not* provide information regarding how many people *in total* applied to each university.

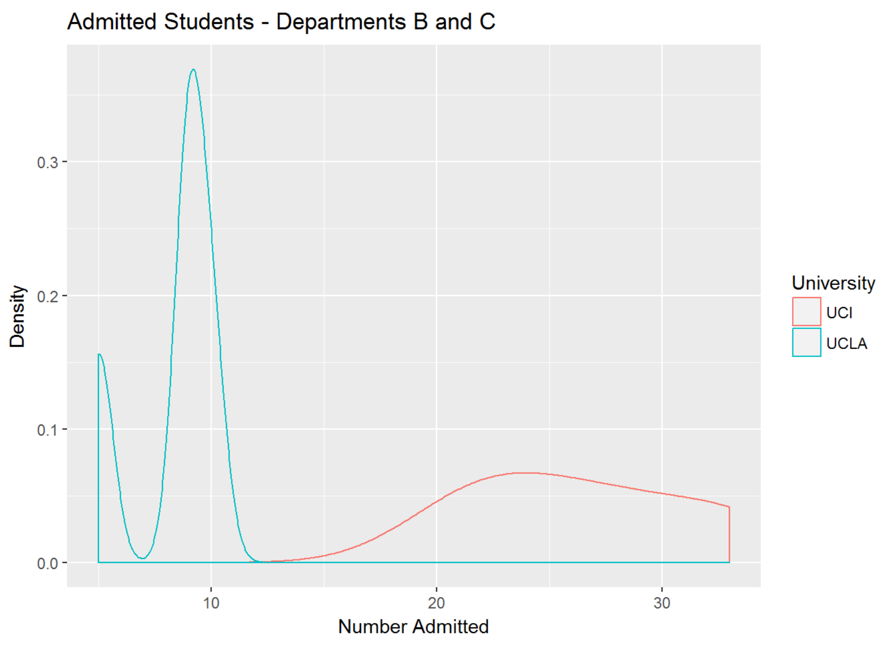
Let’s take a look at the density plots for admitted students in two specific departments at UCI and UCLA.

```
admitted_students_dep <- filter(universities, Admit == "Admitted", Dept == "B"| Dept == "C", University == "UCLA"|
University == "UCI")

ggplot(admitted_students_dep, aes(Freq, color = University, fill = University)) + geom_density(alpha = 0.40) + ggt
itle("Admitted Students - Departments B and C") + labs(x= "Number Admitted", y = "Density")
```



Note that the “fill” parameter is what adds color to the inside of each density plot. If we hadn’t provided that parameter, our graph would have looked like this:

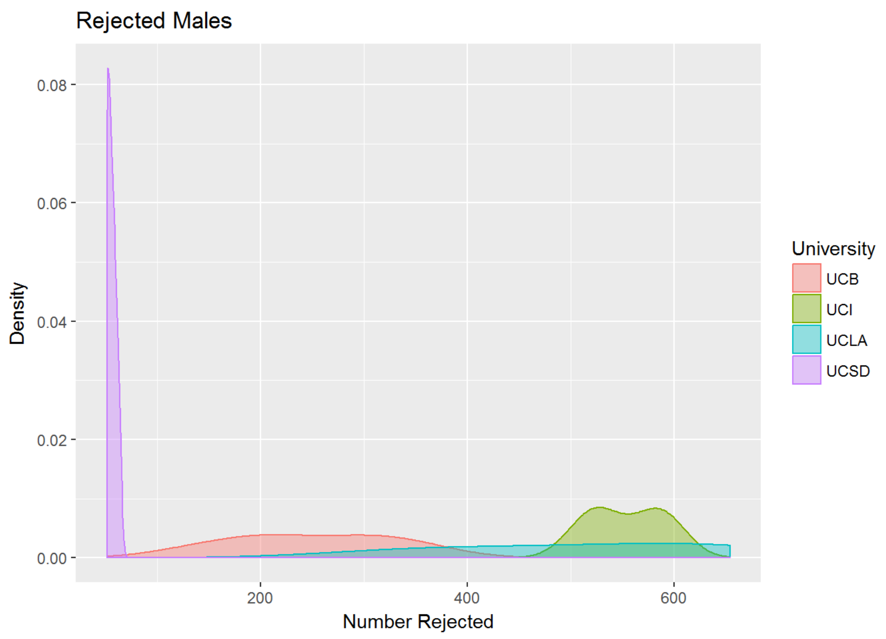
```
# No fill

ggplot(admitted_students_dep, aes(Freq, color = University)) + geom_density(alpha = 0.40) + ggtitle("Admitted Stud
ents - Departments B and C") + labs(x= "Number Admitted", y = "Density")
```



Doesn’t look quite as nice!

Now let’s take a look at the number of rejected males at UCLA, UCI, UCB, and UCSD.

```
rejected_males <- filter(universities, Gender == "Male", Admit == "Rejected")

ggplot(rejected_males, aes(Freq, color = University, fill = University)) + geom_density(alpha = 0.40) + ggtitle("R
ejected Males") + labs(x= "Number Rejected", y = "Density")
```

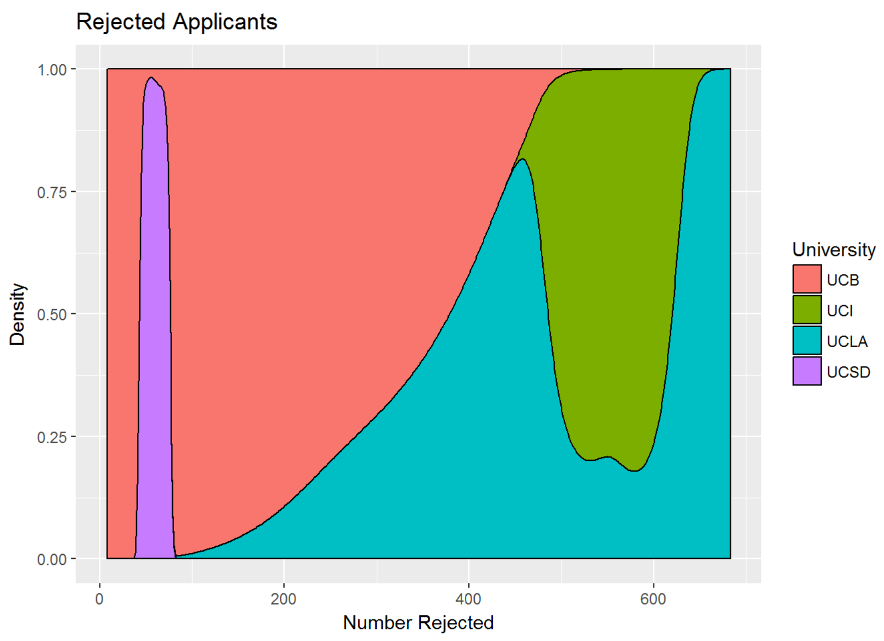


From these overlaid density plots, we can see that UCSD rejected the smallest number of males while UCI appears to have rejected the most.

Before moving on from density plots, I would like to walk through an example of a *stacked* density plot! Not only do these density plots help one visualize potential trends in one’s data, but they also look really cool!

For this example, we’ll be looking at the number of rejected males and females for all four of the universities.

```
rejected <- filter(universities, Admit == "Rejected")

# Stacked density plot for rejected male and female applicants

ggplot(rejected, aes(Freq, group = University, fill = University)) + geom_density(position = "fill") + ggtitle("Re
jected Applicants") + labs(x = "Number Rejected", y = "Density")
```



The part of the above code that is responsible for creating a *stacked* density plot is the “position” argument in geom_density(). If we were to exclude this argument and leave geom_density() emtpy, the output would consist of 4 regular density plots (one for each university).

Area Graphs

We’ll now move on to talking about area graphs!

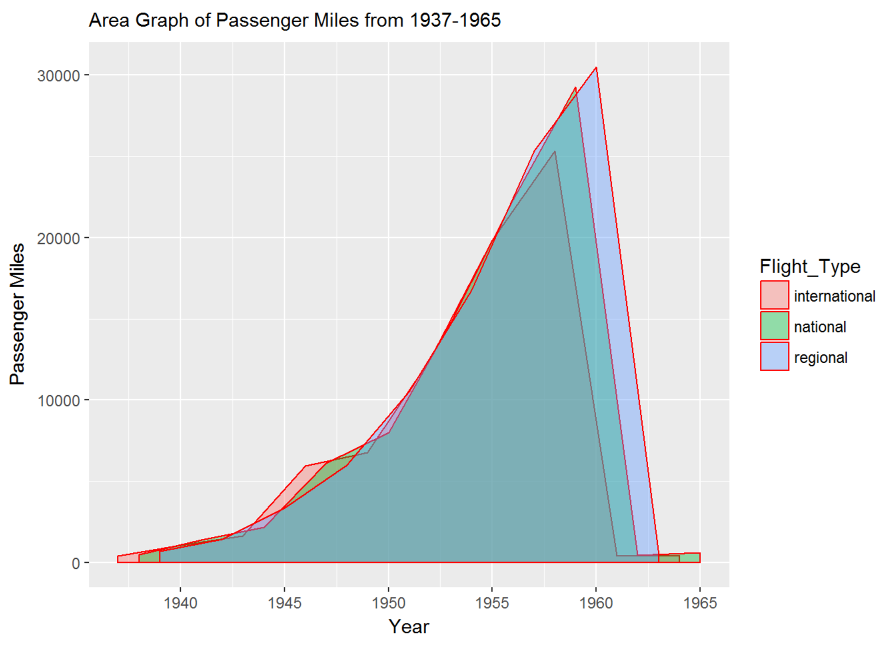
But *first*, we’ll need to make use of another data set. This time we’ll be using the R data set “airmiles.” This data set has been described by RStudio as “Passenger Miles on Commercial US Airlines, 1937-1960.” As was required for the data set “UCBAdmissions”, we’ll first need to do a bit of data manipulation/preparation. It is important to keep in mind that the original “airmiles” data is a **time series** object.

```
# Create a data frame "air_miles" from the R data set/time series "airmiles"

air_miles <- as.data.frame(airmiles)

# Rename the data frame's only column "Passenger_mi"

names(air_miles) <- "Passenger_mi"

# Create a new time series object called "passenger_miles"

passenger_miles <- ts(c(as.vector(air_miles$Passenger_mi), 400, 450, 460, 470, 600), start = 1937, end = 1965)

# Create a new data frame called "air_miles2"

air_miles2 <- data.frame("Pass_mi" = passenger_miles)

# Add a new column "Year" to the data frame "air_miles2" and assign it a vector containg the numbers 1 through 29

air_miles2$Year <- rep(1:29)

# Replace the numbers in the column "Year" with the years 1937 through 1965

holder <- 0
for(i in 1:length(air_miles2$Year)){
  air_miles2$Year[i] <- 1937 + holder
  holder = holder + 1

}

# Add a "type" of flight column to the data frame "air_miles2"

type_of_flight <- c()
first <- 1
second <- 2
third <- 3

for(i in 1:nrow(air_miles2)){
  if(i == first){
    type_of_flight <- c(type_of_flight, "international")
    first = first + 3
  } else if(i == second){
    type_of_flight <- c(type_of_flight, "national")
    second = second + 3
  }else if(i == third){
    type_of_flight <- c(type_of_flight, "regional")
    third = third + 3
  }
}

air_miles2$Flight_Type <- type_of_flight
```

Great! Now we'll use this data to create an area graph. Note that in order to create an area graph you must use **geom_area()**.

```
# Area graph for passenger miles from 1937 to 1965

ggplot(air_miles2, aes(x = Year, y = Pass_mi, fill = Flight_Type)) + geom_area(alpha = 0.4, color = "red") + ggtit
le("Area Graph of Passenger Miles from 1937-1965") + theme(plot.title = element_text(size = 11)) + labs(y = "Passe
nger Miles")
```

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



Ok, so we've now seen what an area graph looks like! Let's take a look at a few more. We'll be doing a bit more data manipulation in order to produce different area graphs. The first two graphs shown below can be considered semi-stacked area graphs (I, myself, have labeled them this way simply because they do not look like standard stacked area graphs). The third graph we will create is an example of a standard **stacked** area graph.

The purpose of this next code chunk is to create a new data frame, "air_miles3", whose only "Flight_Type" is "international".

```r
# Create a numeric vector called "passenger_miles2"

passenger_miles2 <- as.vector(air_miles$Passenger_mi)

# Create a new data frame called "air_miles3"

air_miles3 <- data.frame("Pass_mi" = passenger_miles2)

# Add a new column "Year" to the data frame "air_miles3" and assign it a vector containg the numbers 1 through 24

air_miles3$Year <- rep(1:24)

# Replace the numbers in the column "Year" with the years 1937 through 1960

holder <- 0
for(i in 1:length(air_miles3$Year)){
  air_miles3$Year[i] <- 1937 + holder
  holder = holder + 1

}

# Add a "type" of flight column to the data frame "air_miles2" (only "international")

type_of_flight <- c()

for(i in 1:nrow(air_miles3)){
    type_of_flight <- c(type_of_flight, "international")
}

air_miles3$Flight_Type <- type_of_flight
```

In the code chunk below, we will create another data frame called "air_miles4" from "air_miles3". The code for creating this data frame is a bit longer and more complicated since my intention was to create a data frame with 5 instances of each year as opposed to 1, meaning that "air_miles4" has 120 rows instead of 24.

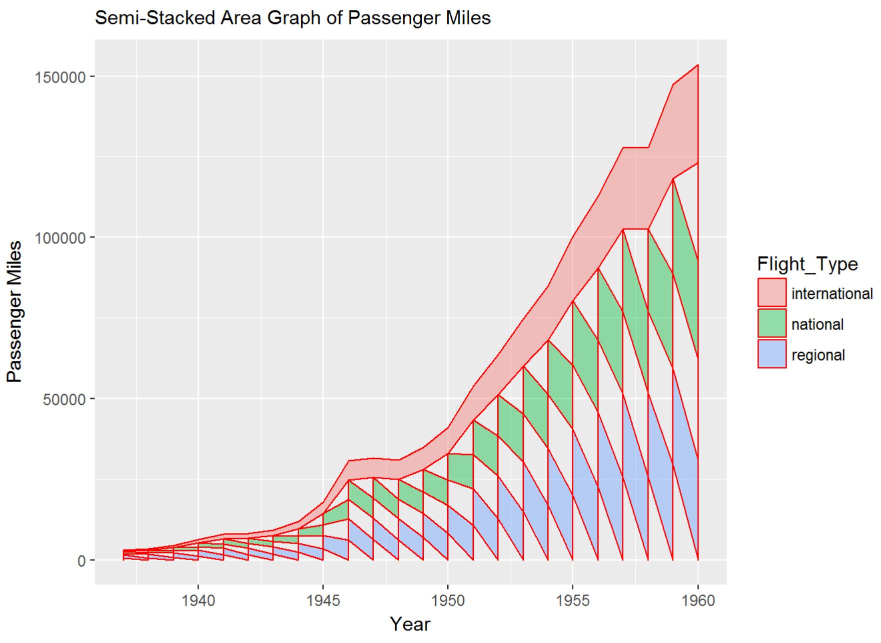
Additionally, for every 5 rows there is 1 international flight, 2 national flights, and 2 regional flights. From the output of the code chunk below, one can see that (for every five rows) the passenger miles for the international flight and the first national flight are the same. The same goes for the second national flight and the first regional flight.

```r
loop1_index <- 1
loop2_index <- 1
holder <- 2
flight_type <- c("international", "national", "regional")
f_index <- 2
year_holder <- 1
row_index <- 1

# Initially, we set air_miles4 equal to air_miles3

air_miles4 <- air_miles3

# The outer while loop specifies "<=24" because air_miles4 initially contains 24 rows. The inner while loop specif
ies "<=4" because for each original row, we are adding 4 new rows, namely the 2 "national" rows and the 2 "regiona
l" rows.

while(loop1_index <= 24){
while(loop2_index <= 4){

# Working in groups of 5 rows, this is how we add the first "national" flight
  air_miles4[holder, ] <- c(air_miles3$Pass_mi[row_index], air_miles3$Year[year_holder], flight_type[f_index])
holder = holder + 1
traverse = holder
loop2_index = loop2_index + 1

# This is how we add the second "national flight"
  if(traverse == holder){
  air_miles4[holder, ] <- c(air_miles3$Pass_mi[row_index] + 200, air_miles3$Year[year_holder], flight_type[f_index
])
  f_index <- f_index + 1
  holder = holder + 1
  traverse = holder
  loop2_index = loop2_index + 1
  }

# This is how we add the first "regional" flight
  if(f_index == 3){
  air_miles4[holder, ] <- c(air_miles3$Pass_mi[row_index] + 200, air_miles3$Year[year_holder], flight_type[f_index
])
  holder <- holder + 1
  traverse = holder
  loop2_index = loop2_index + 1
  }

# This is how we add the second "regional" flight
  if(traverse == holder){
  air_miles4[holder, ] <- c(air_miles3$Pass_mi[row_index] + 700, air_miles3$Year[year_holder], flight_type[f_index
])
  holder = holder + 1
  traverse = holder
  loop2_index = loop2_index + 1

  }

# This is how we keep the original rows from "air_miles3"
  if(traverse == holder){
    air_miles4[holder, ] <- air_miles3[row_index + 1, ]
    holder = holder + 1
    }

}

# We reset in order to be able to repeat the process coded within the inner while loop
  row_index = row_index + 1
  year_holder = year_holder + 1
  f_index = 2
  loop1_index <- loop1_index + 1
  loop2_index <- 1
}

# We need to make sure that "Year" and "Pass_mi" are both numeric in order to be able to output an area graph!
air_miles4$Year <- as.numeric(air_miles4$Year)

air_miles4$Pass_mi <- as.numeric(air_miles4$Pass_mi)
air_miles4 <- air_miles4[1:120, ]

# Let's take a look at what air_miles4 looks like
head(air_miles4, 20)
```

```
##    Pass_mi Year   Flight_Type
## 1      412 1937 international
## 2      412 1937      national
## 3      612 1937      national
## 4      612 1937      regional
## 5     1112 1937      regional
## 6      480 1938 international
## 7      480 1938      national
## 8      680 1938      national
## 9      680 1938      regional
## 10    1180 1938      regional
## 11     683 1939 international
## 12     683 1939      national
## 13     883 1939      national
## 14     883 1939      regional
## 15    1383 1939      regional
## 16    1052 1940 international
## 17    1052 1940      national
## 18    1252 1940      national
## 19    1252 1940      regional
## 20    1752 1940      regional
```

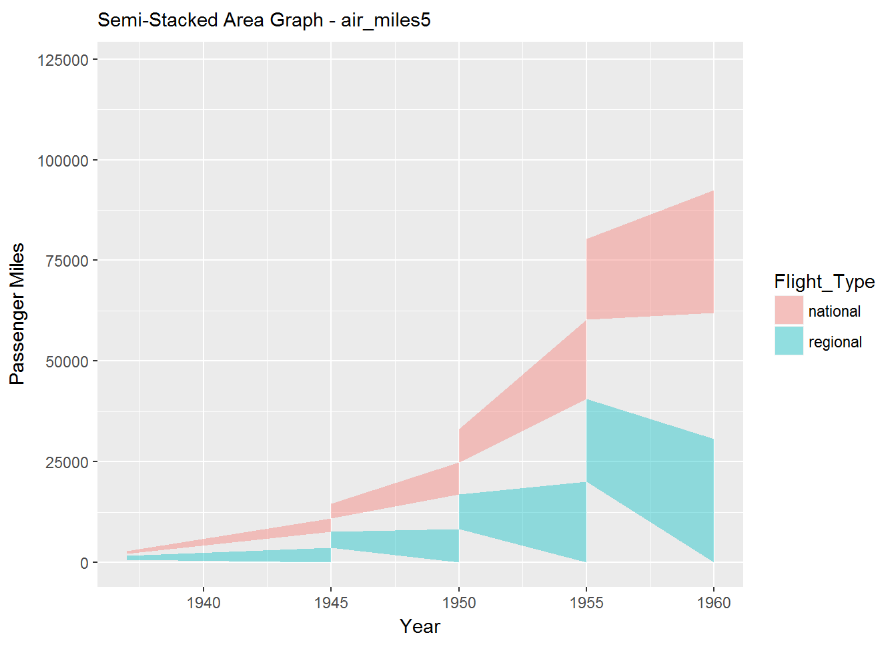
Now let's create a semi-stacked area graph for "air_miles4"!

```
# Semi-stacked area graph for "air_miles4"

ggplot(air_miles4, aes(x = Year, y = Pass_mi, fill = Flight_Type)) + geom_area(alpha = 0.4, color = "red") + ggtit
le("Semi-Stacked Area Graph of Passenger Miles") + theme(plot.title = element_text(size = 11)) + labs(y = "Passeng
er Miles")
```



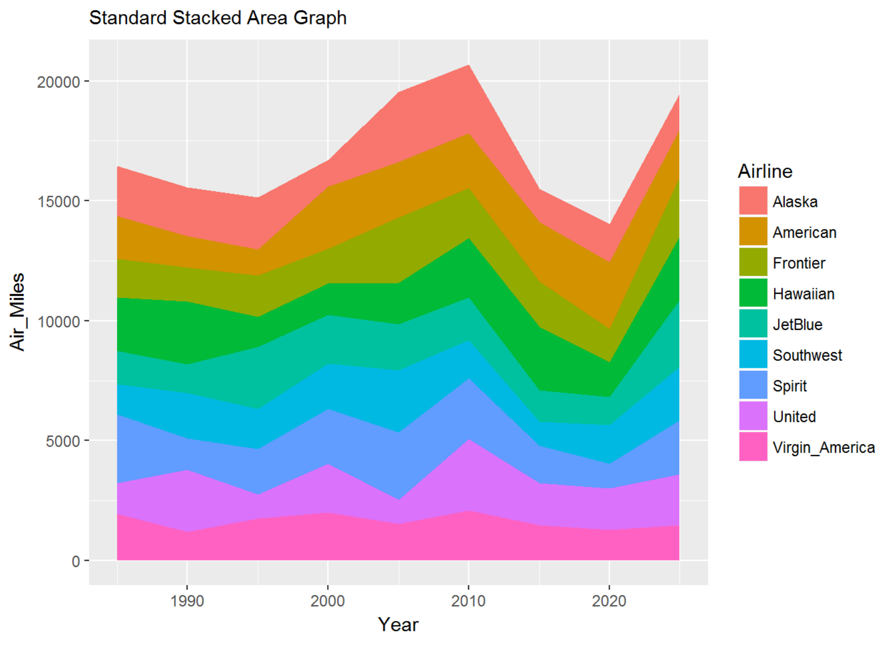
Below is the code for our second semi-stacked area graph example. We'll be creating another data frame called "air_miles5" from "air_miles4". The data frame "air_miles5" differs from "air_miles4" in that it only lists "national" and "regional" flights and includes only those flights which occurred in the years 1937, 1945, 1950, 1955, and 1960.

```
air_miles5 <- air_miles4[air_miles4$Flight_Type == "national"| air_miles4$Flight_Type == "regional", ]

air_miles5 <- air_miles5[air_miles5$Year == 1937 | air_miles5$Year == 1945| air_miles5$Year == 1950| air_miles5$Ye
ar == 1955| air_miles5$Year == 1960, ]

# Semi-stacked area graph for "air_miles5"

ggplot(air_miles5, aes(x = Year, y = Pass_mi, fill = Flight_Type)) + geom_area(alpha = 0.4) + ggtitle("Semi-Stacke
d Area Graph - air_miles5") + theme(plot.title = element_text(size = 11)) + labs(y = "Passenger Miles")
```



Ok, now that we've seen a couple of "semi-stacked" area graphs, let's look at a more standard one!

```
# First, let's create a data frame called "airline_info"

Airline <- rep(c("Alaska", "JetBlue", "United", "Spirit", "American", "Virgin_America", "Southwest", "Frontier", "
Hawaiian"), times = 9)

Year <- rep(c(1985, 1990, 1995, 2000, 2005, 2010, 2015, 2020, 2025), each = 9)
set.seed(900)

Air_Miles <- runif(81, 1000, 3000)

airline_info <- data.frame(Airline, Year, Air_Miles)

# Standard area graph using the data frame "airline_info"

ggplot(airline_info, aes(x = Year, y = Air_Miles, fill = Airline)) + geom_area() + ggtitle("Standard Stacked Area
Graph") + theme(plot.title = element_text(size = 11))
```



Notice that because the areas are stacked, airlines such as Alaska Airlines and American Airlines appear near the 15,000 Air_Miles mark. Keep in mind, however, that these airlines also only have between 1,000 and 3,000 Air_Miles, as defined in the code chunk above.

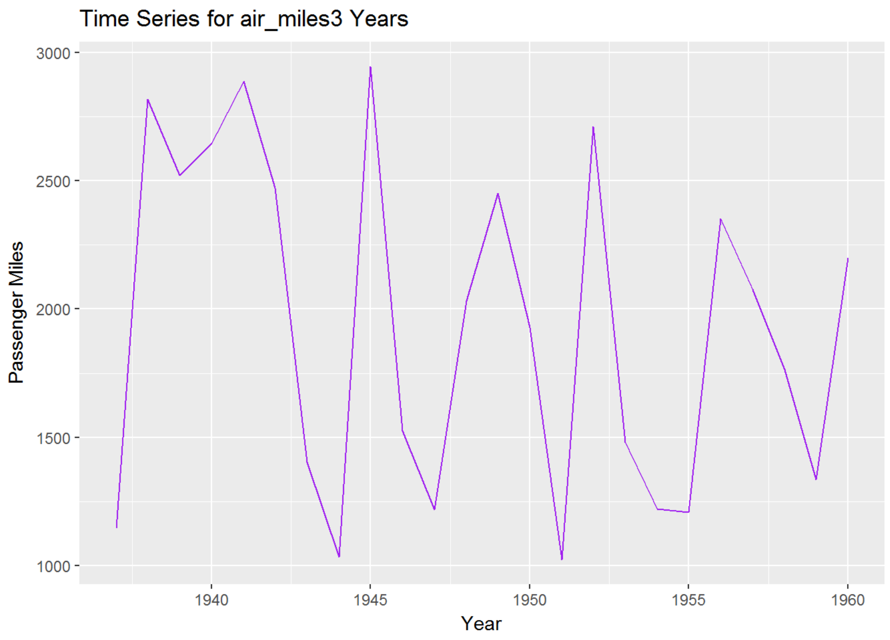
Time Series Plots

Ok, now let's move onto a third type of ggplot graph.

Let's create a time series plot!

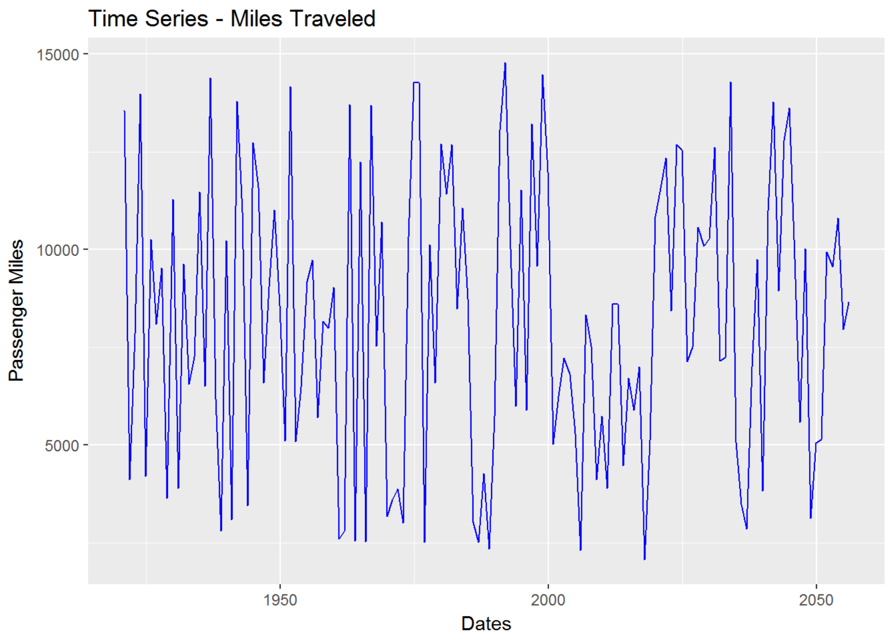
In the example code below, we'll be using part of "air_miles3", one of the data frames we created above to plot a time series. Note that in order to plot a time series, you need to use **geom_line()**.

```
set.seed(89)
air_miles3$Pass_mi <- runif(24, 1000, 3000)

ggplot(air_miles3, aes(x = Year, y = Pass_mi)) + geom_line(col = "purple") + ggtitle("Time Series for air_miles3 Y
ears") + labs(y = "Passenger Miles")
```



Let’s look at another time series that has more dates.

```
dates <- seq(as.Date("1920/12/11"), as.Date("2055/12/11"), "years")

set.seed(67)
miles_traveled <- runif(136, 2000, 15000)

times_df <- data.frame(dates, miles_traveled)

ggplot(times_df, aes(x = dates, y = miles_traveled)) + geom_line(col = "blue") + labs(x = "Dates", y = "Passenger
Miles") + ggtitle("Time Series - Miles Traveled")
```



Level Plots

I’ll now be moving on to discuss level plots!

Below is the code for our first example.

```
# Generate 30 reproducible, random values from 1 to 10

# Let's suppose that these values represent the sugar content of our first and second food items (perhaps for lunc
h)

set.seed(90)
sugar_content1 <- order(runif(30, 1, 10))

set.seed(95)
sugar_content2 <- order(runif(30, 1, 10))

# Using expand.grid() will output a data frame that contains every combination of sugar_content1 and sugar_content
2

sugar_combos <- expand.grid(sugar_content1 = sugar_content1, sugar_content2 = sugar_content2)
```

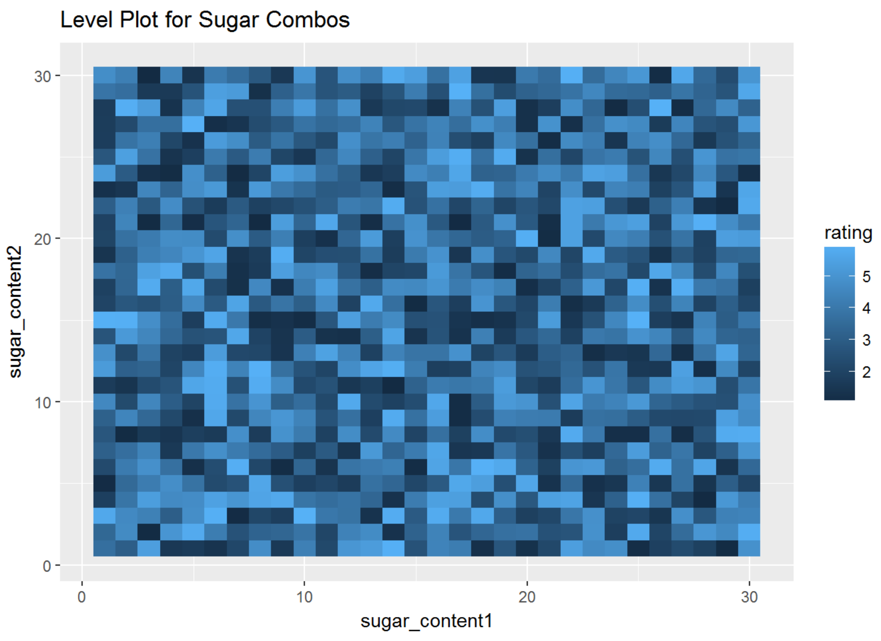
Note: If you decide to use runif() in order to generate random values for the numeric vectors which you will use to create your data frame, make sure to use order()! If the values of your numeric vector are not ordered, you will not be able to generate a level plot. Alternatively, you can use seq(), which will automatically output ordered numbers.

In order to give some context to our example, the values coded for below can be understood as the rating given by an individual after eating a meal that consisted of two food items with a particular sugar combination.

```
# We now want to add a "rating" column to our "sugar_combos" data frame

set.seed(45)
rating <- runif(900, 1, 6)

sugar_combos$rating <- rating
```

Now for the actual level plot! Note that the main component here is **geom_tile()**.

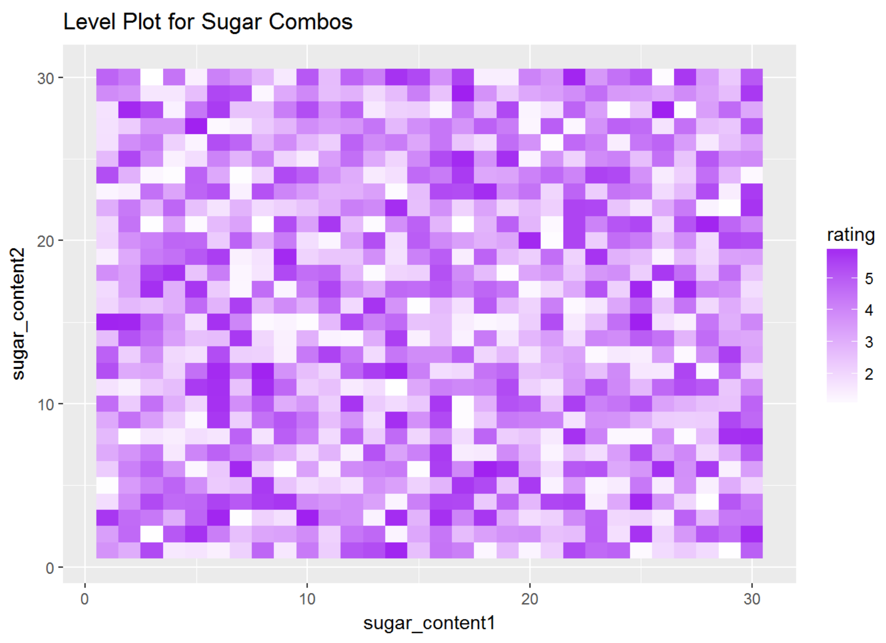
```
ggplot(sugar_combos, aes(x = sugar_content1, y = sugar_content2, z = rating)) + geom_tile(aes(fill = rating)) + gg
title("Level Plot for Sugar Combos")
```



How might we change the colors of the level plot?

We can use **scale_fill_gradient()**!

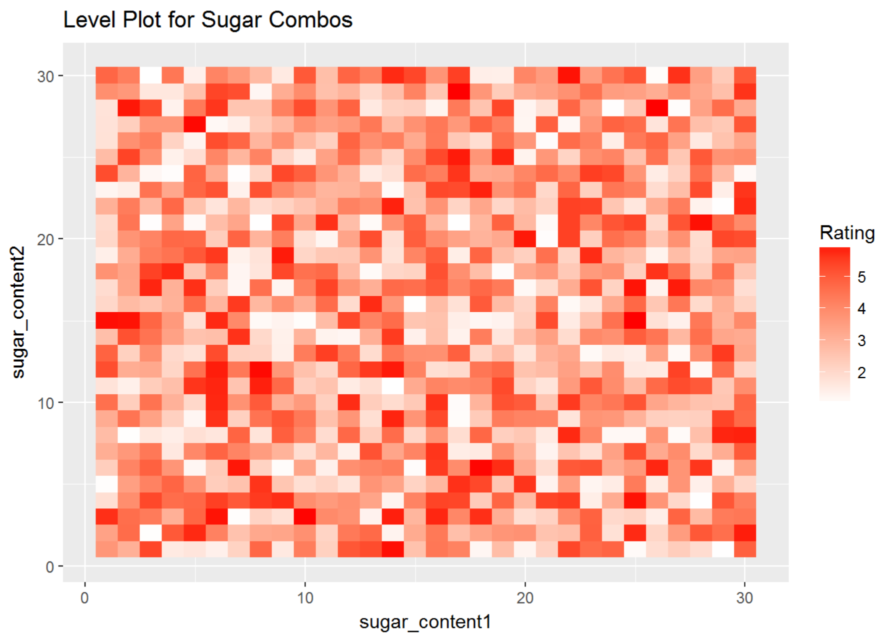
```
# Changing the colors used in the orignal level plot

ggplot(sugar_combos, aes(sugar_content1, sugar_content2, rating)) + geom_tile(aes(fill = rating)) + ggtitle("Level
Plot for Sugar Combos") + scale_fill_gradient(low = "white", high = "purple")
```



Great! Now we know how to create level plots! The level plot displayed above also provides a good opportunity to go over a more minor detail of ggplot aesthetics. Notice how the title of the legend ("rating") is not capitalized. How might we change the title to "Rating"?

You can do this by using **labs()** and specificying which parameter label you wish to modify. In this particular case, we modified the name of the "fill" argument.

```
# Changing the legend title from "rating" to "Rating"

ggplot(sugar_combos, aes(sugar_content1, sugar_content2, rating)) + geom_tile(aes(fill = rating)) + ggtitle("Level
Plot for Sugar Combos") + scale_fill_gradient(low = "white", high = "red") + labs(fill = "Rating")
```



Take-Home Message

From this post, we are able to see that ggplot2 has many different types of graphs and applications. With ggplot2, one has the ability to generate the type of graph that best suites one's data or that will best represent a certain trend one might want to highlight. As we have seen, a level plot is clearly very different from a time series plot; the input data each works with is also quite different. To summarize, in this post we have gone over examples of **density plots**, **area graphs**, **time series plots**, and **level plots**!

You got through the post!

References

I would like to specifically thank and recognize Yan Holtz for providing his site *The R graph Gallery*, which served as inspiration for the examples in this post.

Site 1

Site 2

Site 3

Site 4

Site 5

Site 6

Site 7

Site 10

Site 11

Site 12

Site 13

Site 14