

# STA234 Midterm

Derin Gezgin

2025-04-08

92/100 + 5 bonus=97 😊

## Importing the Required Libraries

```
library(tidyverse)
library(ggalt)
library(UsingR)
library(nycflights13)
```

## Problem 1 - [50 points]

Using the flights data from nycflights13 package, answer the following:

1.(a) [2 points]

*Extract flights from carrier Alaska (AS) leaving NYC in 2013. Use this subset for parts (b) through (f).*

```
alaskaFlights = flights %>%
  filter(carrier == "AS")
```

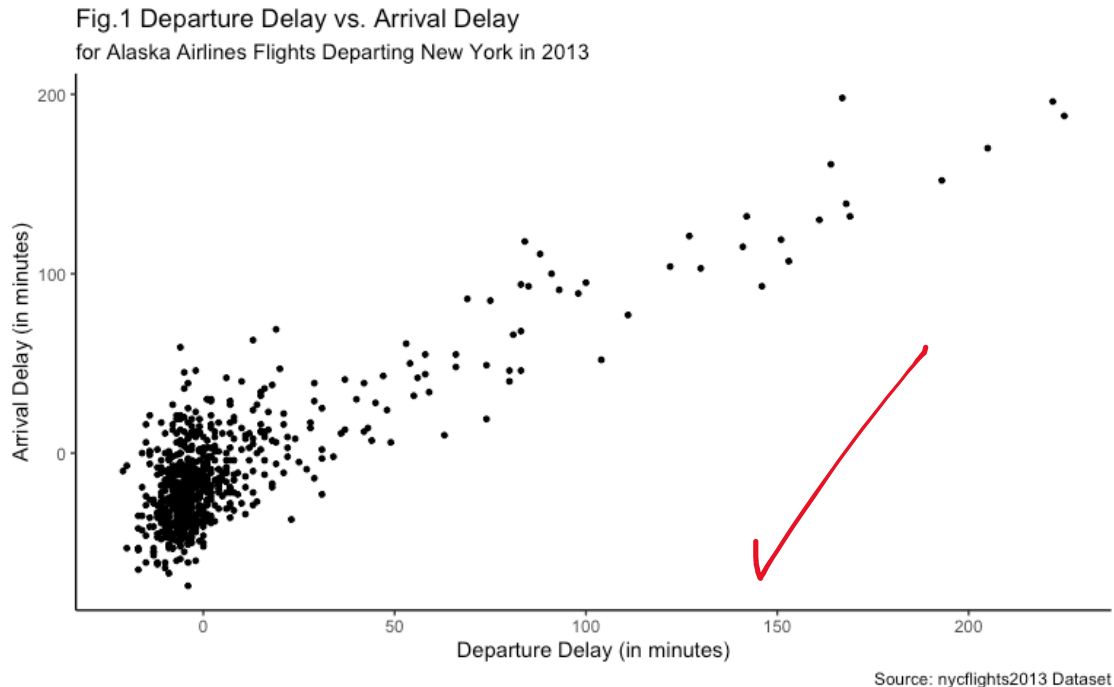
We can use the filter function to extract the Alaska Airlines flights from the NYC flights in 2013.

1.(b) [5 points]

*Show scatterplot to explain the relation between dep\_delay (x-axis) and arr\_delay (y-axis).*

```
depArrDelaySplot = ggplot(data = alaskaFlights,
                           aes(x = dep_delay,
                               y = arr_delay)) +
  geom_point(size = 1) +
  labs(title = "Fig.1 Departure Delay vs. Arrival Delay",
       subtitle = "for Alaska Airlines Flights Departing New York in 2013",
       x = "Departure Delay (in minutes)",
       y = "Arrival Delay (in minutes)",
       caption = "Source: nycflights2013 Dataset") +
  theme_classic()
```

depArrDelaySplot



### *Comment on the nature of the relation.*

We can definitely see an upward linear trend in the positive direction between the departure delay and the arrival delay. This makes sense as if a flight departs the airport late, it is expected and usual that, it will arrive the destination airport also late with a similar delay. This also explains the strong linear (nearly 45 degrees) relationship as it is nearly impossible for a flight to depart late and arrive with a significantly lower delay than the departure delay.

### *Why do you think there is a cluster of points near (0, 0)? What does (0, 0) correspond to in terms of the Alaska Air flights?*

(0,0) represents the flights which departed and arrived on time. There is a cluster of points around (0,0) as having a large departure/arrival delay is not common in general. We can say that, the Alaska Airlines flights were mostly on time considering the cluster of points around (0,0).

### *How would you try to spread out the points in the cluster?*

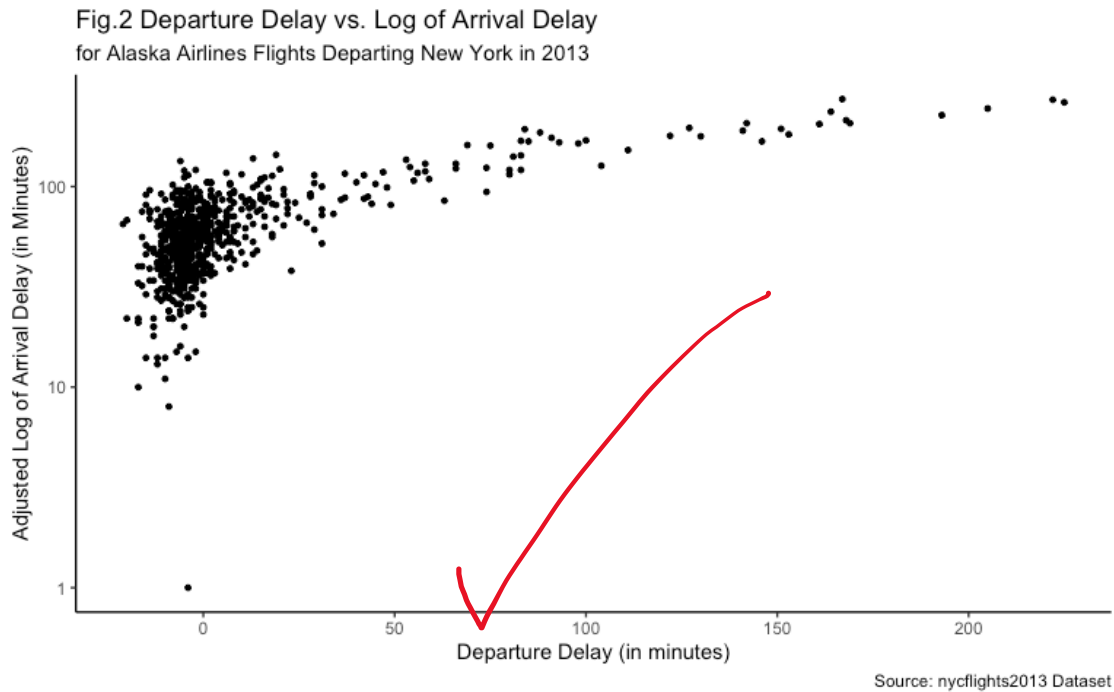
```
alaskaFlightsShifted = alaskaFlights %>%
  mutate(arr_delay = arr_delay - min(arr_delay, na.rm = TRUE) + 1)
```

We can spread out the points in the cluster using a transformation to the data. In this case, I used the log transformation on the Y-Axis variable: arrival delay. As the log transformation can be applied to positive values (negative values got removed) I shifted all the arrival delay values by the minimum value.

Show the plot.

```
depLogArrDelaySplot = depArrDelaySplot %>% alaskaFlightsShifted +  
  scale_y_continuous(trans = "log10") +  
  labs(title = "Fig.2 Departure Delay vs. Log of Arrival Delay",  
    y = "Adjusted Log of Arrival Delay (in Minutes)")
```

depLogArrDelaySplot



It is also possible to remove the obvious outlier in this case, but I did not want to go this further in this question.

**The transformation has not fixed the obvious problem of cluster? It's not spread out yet! -1**

1.(c) [2 points]

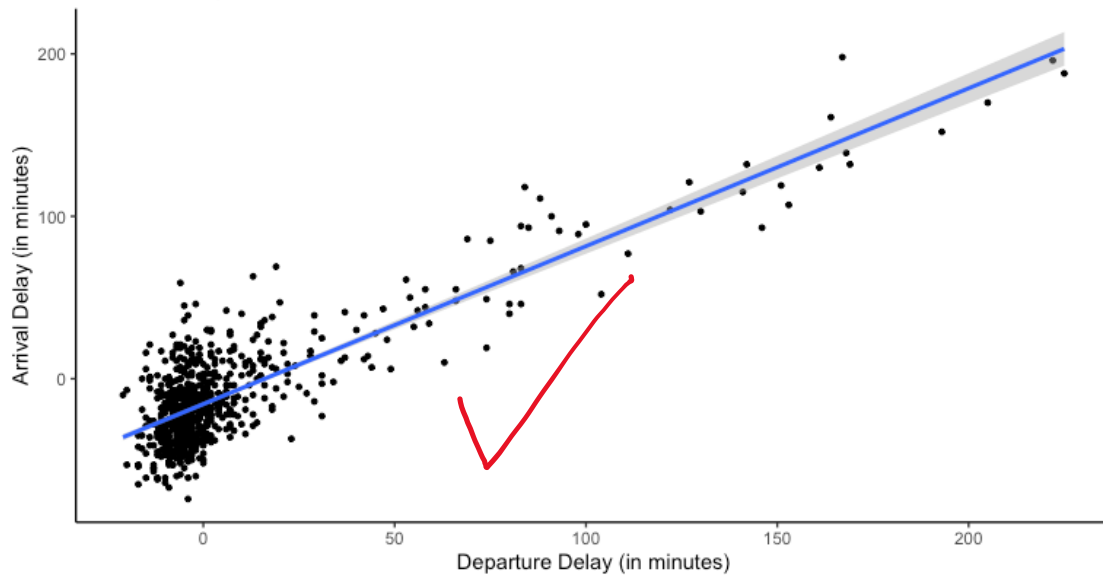
Fit a regression and a non-linear line to the above scatterplots.

Linear regression on the Original Data

```
depArrDelaySplotLinReg = depArrDelaySplot +  
  geom_smooth(method = "lm") +  
  labs(title = "Fig.3 Departure Delay vs. of Arrival Delay",  
    subtitle = "for Alaska Airlines Flights Departing New York in  
    2013\nwith Linear Regression Line")
```

depArrDelaySplotLinReg

Fig.3 Departure Delay vs. of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Linear Regression Line



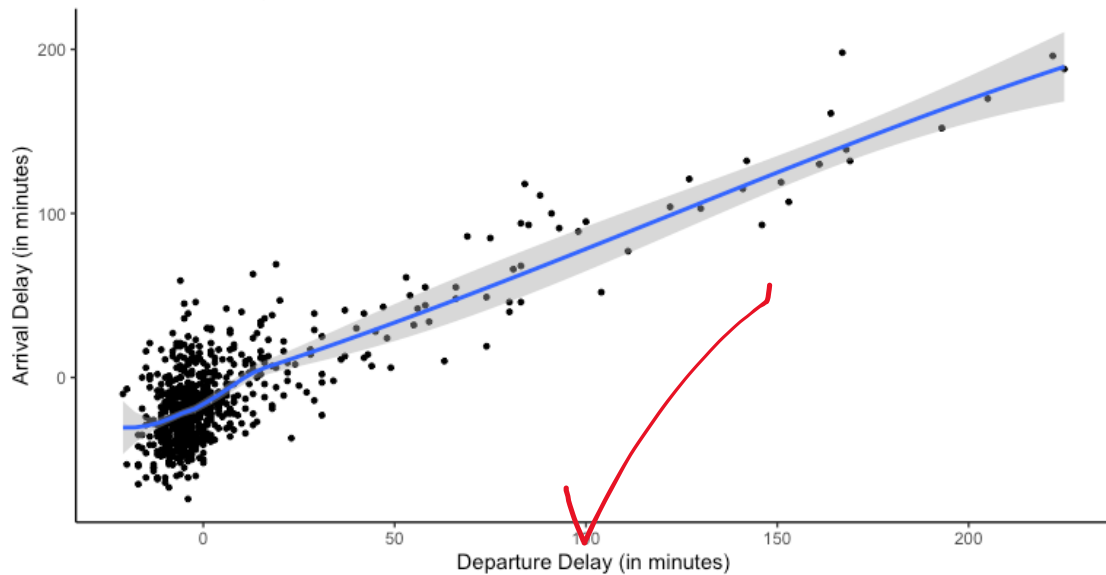
Source: nycflights2013 Dataset

#### Non-Linear Regression on the Original Data

```
depArrDelaySplotNonLinReg = depArrDelaySplot +  
  geom_smooth() +  
  labs(title = "Fig.4 Departure Delay vs. of Arrival Delay",  
        subtitle = "for Alaska Airlines Flights Departing New York in  
2013\nwith Non-Linear Regression Line")
```

```
depArrDelaySplotNonLinReg
```

Fig.4 Departure Delay vs. of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Non-Linear Regression Line



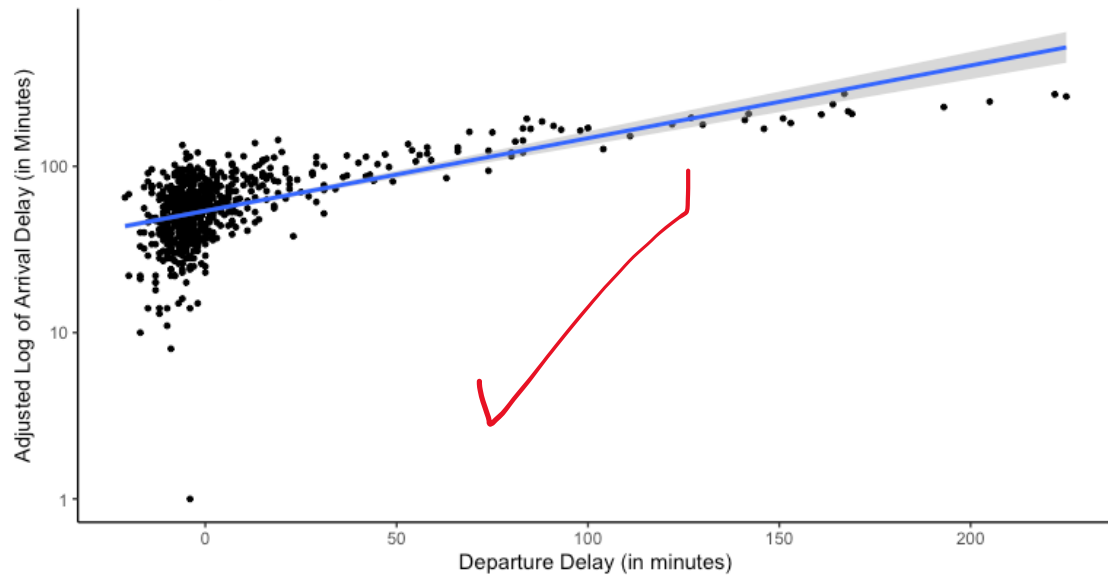
Source: nycflights2013 Dataset

#### Linear Regression on Adjusted Data

```
depLogArrDelaySplotLinReg = depLogArrDelaySplot +
  geom_smooth(method = "lm") +
  labs(title = "Fig.5 Departure Delay vs. Log of Arrival Delay",
        subtitle = "for Alaska Airlines Flights Departing New York in
2013\nwith Linear Regression Line")
```

```
depLogArrDelaySplotLinReg
```

Fig.5 Departure Delay vs. Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Linear Regression Line



Source: nycflights2013 Dataset

**Since the transformation was unable to solve the clustering the model shows that too!**

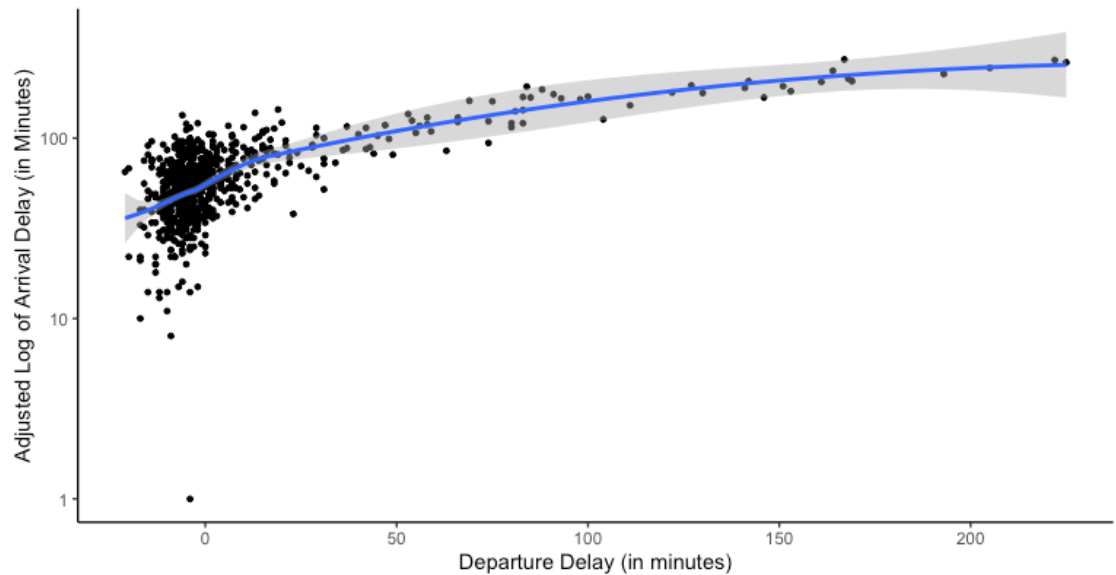
Non-Linear Regression on Adjusted Data

```
depLogArrDelaySplotNonLinReg = depLogArrDelaySplot +  
  geom_smooth() +  
  labs(title = "Fig.6 Relationship Between the Departure Delay and the Log  
of Arrival Delay",  
        subtitle = "for Alaska Airlines Flights Departing New York in  
2013\nwith Non-Linear Regression Line")
```

```
depLogArrDelaySplotNonLinReg
```

**Since the transformation was unable to solve the clustering the model shows that too!**

Fig.6 Relationship Between the Departure Delay and the Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Non-Linear Regression Line



Source: nycflights2013 Dataset

**Since the transformation was unable to solve the clustering the model shows that too!**

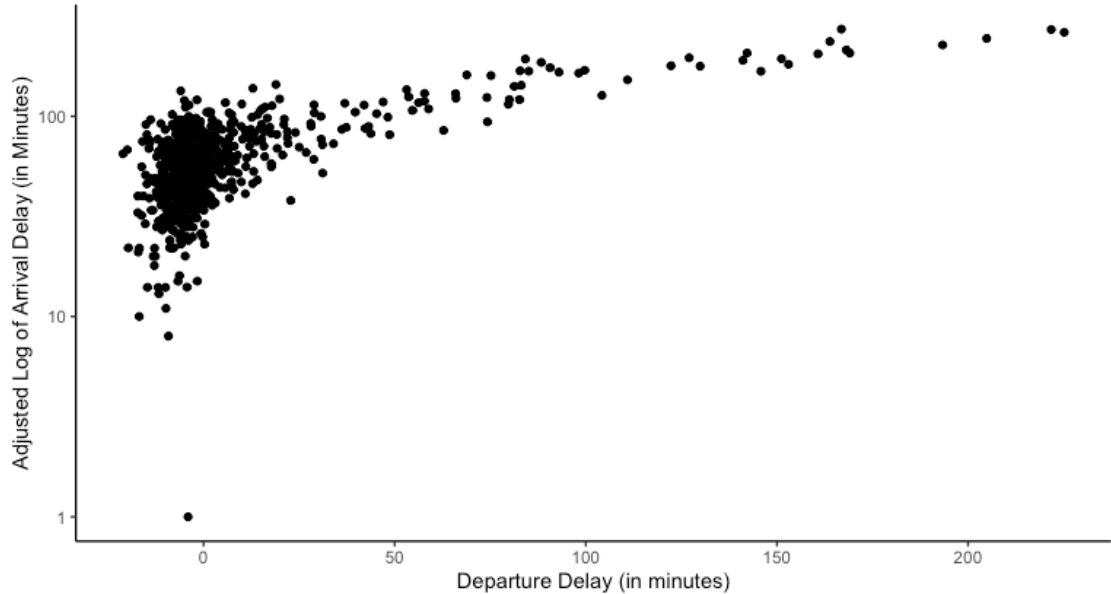
1.(d) [2 points]

*Make a jittered plot to show the points scattered in the scatterplot from part(b).*

```
depLogArrDelayJittered = depLogArrDelaySplot +
  geom_jitter() +
  labs(title = "Fig.7 Relationship Between the Departure Delay and Log of
Arrival Delay",
       subtitle = "for Alaska Airlines Flights Departing New York in 2013")

depLogArrDelayJittered
```

Fig.7 Relationship Between the Departure Delay and Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013



Source: nycflights2013 Dataset

**Adjust width to see more points, here the jitter hasn't helped in any way! -0.5**

1.(e) [2 points]

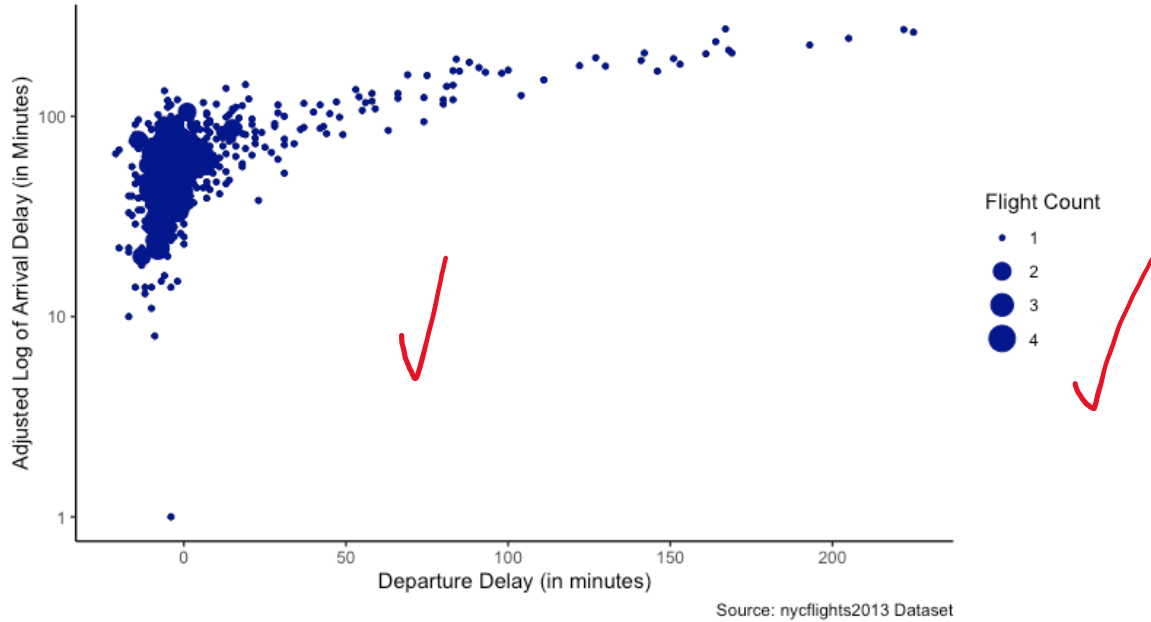
*Make a counts plot for the scatterplot in part (b).*

```
depLogArrDelayCounts = depLogArrDelaySplot +  
  geom_count(col = "darkblue") +  
  labs(title = "Fig.8 Relationship Between the Departure Delay and Log of  
Arrival Delay",  
        subtitle = "for Alaska Airlines Flights Departing New York in 2013",  
        size = "Flight Count")
```

depLogArrDelayCounts



Fig.8 Relationship Between the Departure Delay and Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013



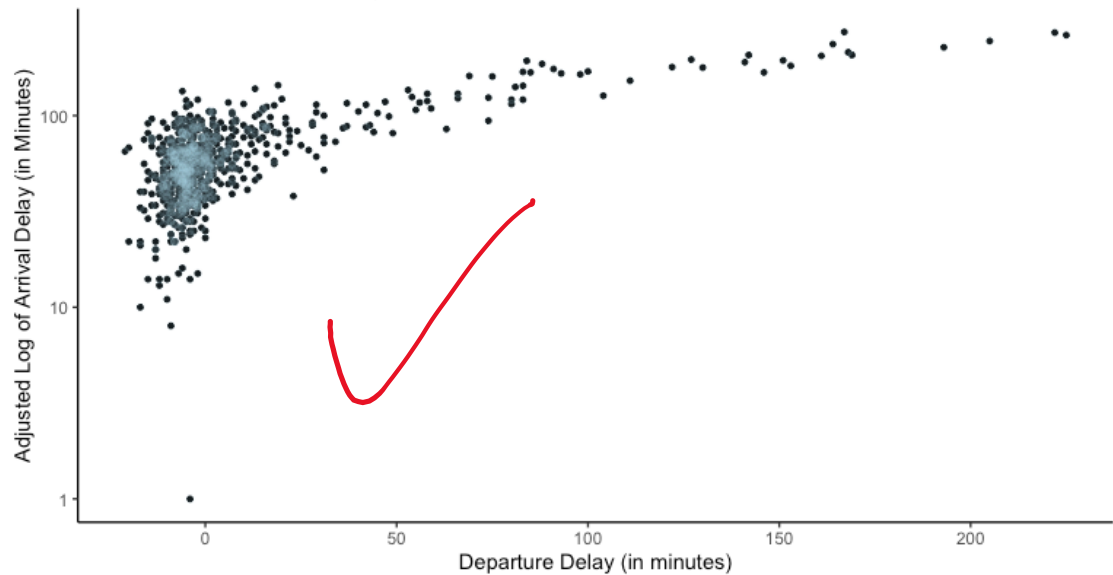
1.(f) [2 points]

*Change the transparency of dots in the scatterplot from part (b).*

```
depLogArrDelaySplotTransparent = depLogArrDelaySplot +
  geom_point(alpha = 0.1,
             color = "lightblue") +
  labs(title = "Fig.9 Relationship Between the Departure Delay and the Log
of Arrival Delay",
       subtitle = "for Alaska Airlines Flights Departing New York in
2013\nwith Reduced Dot Transparency")
```

depLogArrDelaySplotTransparent

Fig.9 Relationship Between the Departure Delay and the Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Reduced Dot Transparency



Source: nycflights2013 Dataset

1.(g) [2 points]

*Change the theme and background color of plot original plot in part (b).*

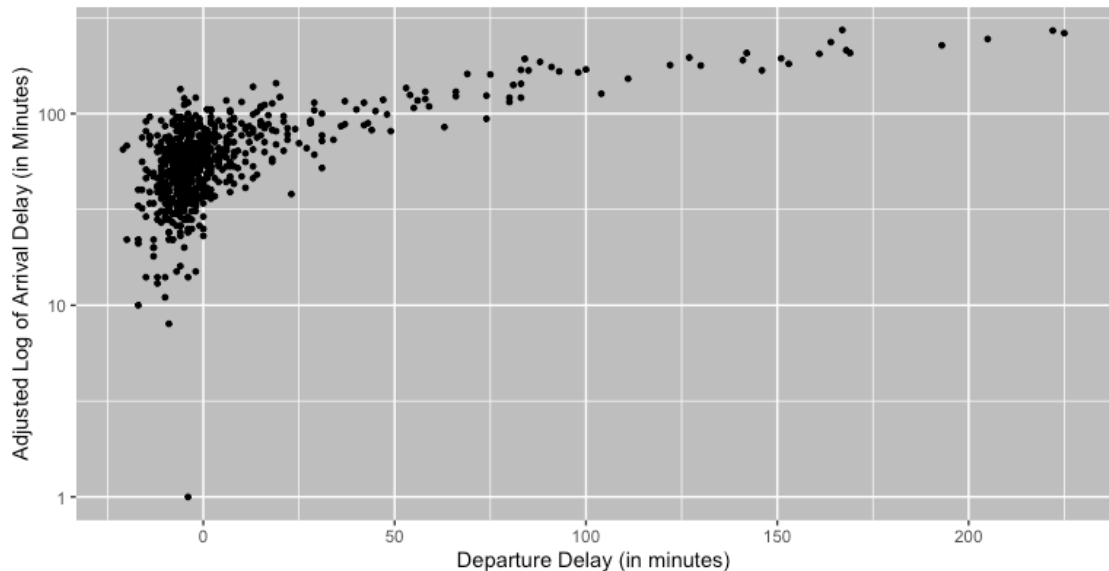
```
depLogArrDelaySplotCustom = depLogArrDelaySplot +
  theme_set(theme_bw()) +
  theme(panel.background = element_rect(fill = "gray")) +
  labs(title = "Fig.10 Relationship Between the Departure Delay and the Log
of Arrival Delay",
       subtitle = "for Alaska Airlines Flights Departing New York in
2013\nwith Custom Theme")
```

depLogArrDelaySplotCustom

**This is not the original plot, it's the transformed one!**

**-1**

Fig.10 Relationship Between the Departure Delay and the Log of Arrival Delay  
for Alaska Airlines Flights Departing New York in 2013  
with Custom Theme



Source: nycflights2013 Dataset

### 1.(h) [8 points]

From original flights data, extract data for UA and AA airlines.

```
flightsUAandAA = flights %>%  
  filter(carrier %in% c("UA", "AA"))
```

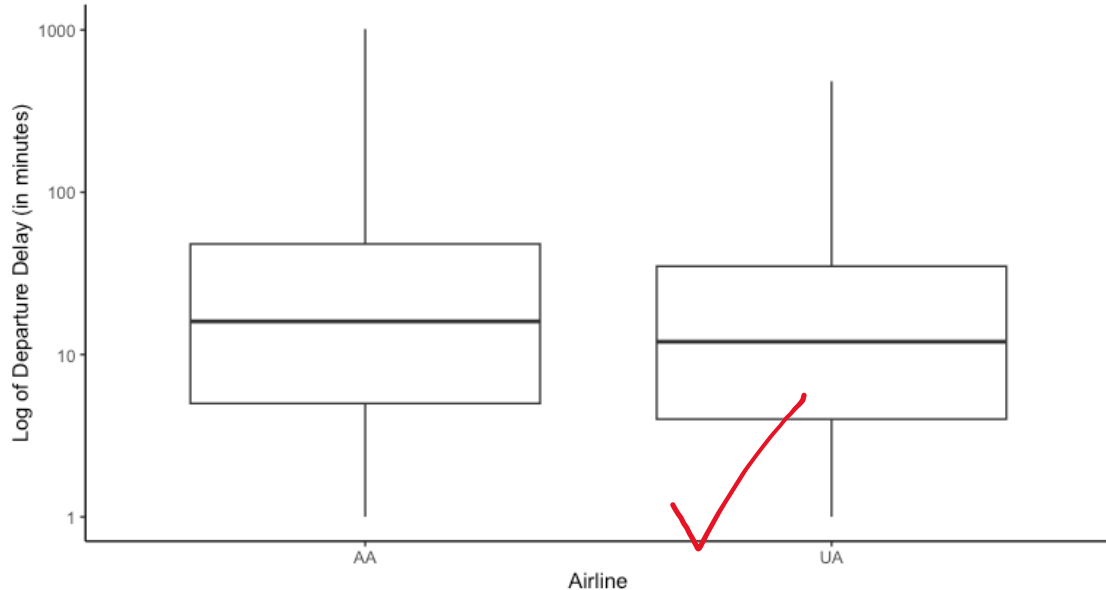
To select the UA and AA airlines flights from the flights data, we can use the filter function.

Make a boxplot of average departure delays.

```
departureDelayBoxplot = ggplot(data = flightsUAandAA,  
                               aes(x = carrier,  
                                   y = dep_delay)) +  
  geom_boxplot() +  
  scale_y_continuous(trans = "log10") +  
  labs(title = "Fig.11 Departure Delays for United Airlines (UA) and  
American Airlines (AA)",  
       subtitle = "for Flights Departing New York in 2013",  
       x = "Airline",  
       y = "Log of Departure Delay (in minutes)",  
       caption = "Source: nycflights2013 Dataset") +  
  theme_classic()
```

departureDelayBoxplot

Fig.11 Departure Delays for United Airlines (UA) and American Airlines (AA)  
for Flights Departing New York in 2013



Source: nycflights2013 Dataset

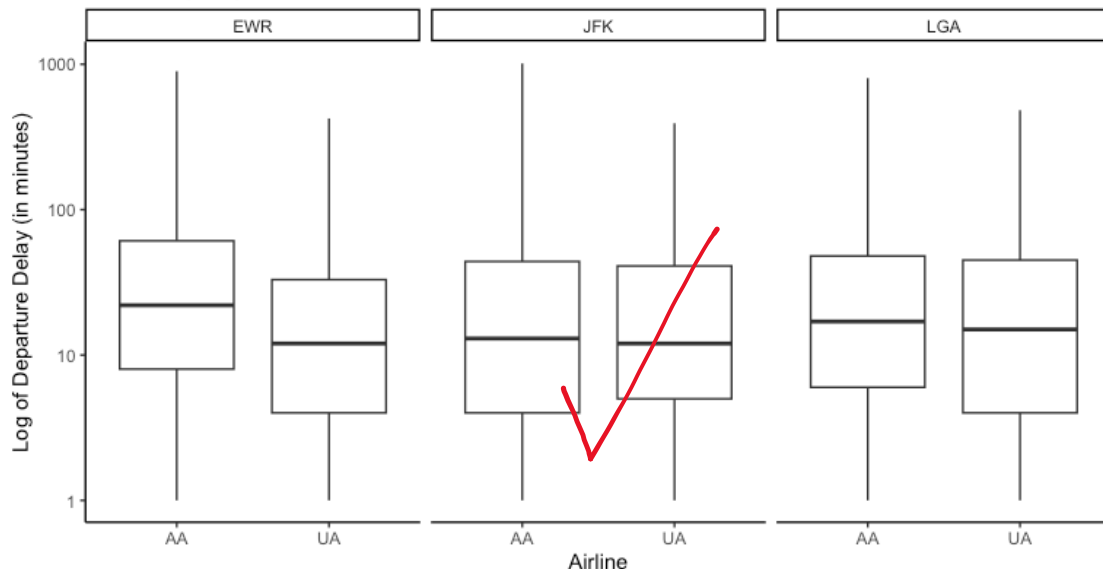
When I looked at the original boxplot of the variables, I saw that they were not distributed evenly, so I applied to log transformation on this part, to better interpret the graphs.

*Then, make boxplots comparing two airlines departure delay by three different origin airports.*

```
departureDelayBoxplotAirport = departureDelayBoxplot +
  facet_wrap(~ origin) +
  labs(title = "Fig.12 Departure Delays for United Airlines (UA) and
American Airlines (AA)",
       subtitle = "for Flights Departing Different Airports in New York in
2013\nFacetted by Origin Airport")

departureDelayBoxplotAirport
```

Fig.12 Departure Delays for United Airlines (UA) and American Airlines (AA)  
for Flights Departing Different Airports in New York in 2013  
Facetted by Origin Airport



Source: nycflights2013 Dataset

Prepare summary of departure delay for two airlines using the function summary.

```
summaryUA = summary(flightsUAandAA$dep_delay[flightsUAandAA$carrier == "UA"])
summaryUA
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -20.00  -4.00    0.00   12.11  11.00   483.00    686
```

```
summaryAA = summary(flightsUAandAA$dep_delay[flightsUAandAA$carrier == "AA"])
summaryAA
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -24.000  -6.000   -3.000    8.586  4.000  1014.000    636
```

Using all this information, which carrier (UA or AA) should you take if you want to have the shortest on average delay from the NYC flights information at 2013?

Considering figures 11 and 12, and the summary tables for United Airlines and American Airlines, I would take a flight with the United Airlines as it has a lower median departure delay for all 3 airports. At the same time, for all 3 airports, it has a lower log of departure delay in minutes. Overall, it can be seen that United Airlines has a lower average departure delay.

. Without Looking at any measure of variability is your response accurate?  
What about outliers?

## 1.(i) [5 points]

In the entire flights data, make a barplot comparing the number of flights from different carriers?

```
countData = flights %>%  
  group_by(carrier, origin) %>%  
  summarise(flight_count = n()) %>%  
  mutate(flightPercentage = flight_count / sum(flight_count))
```

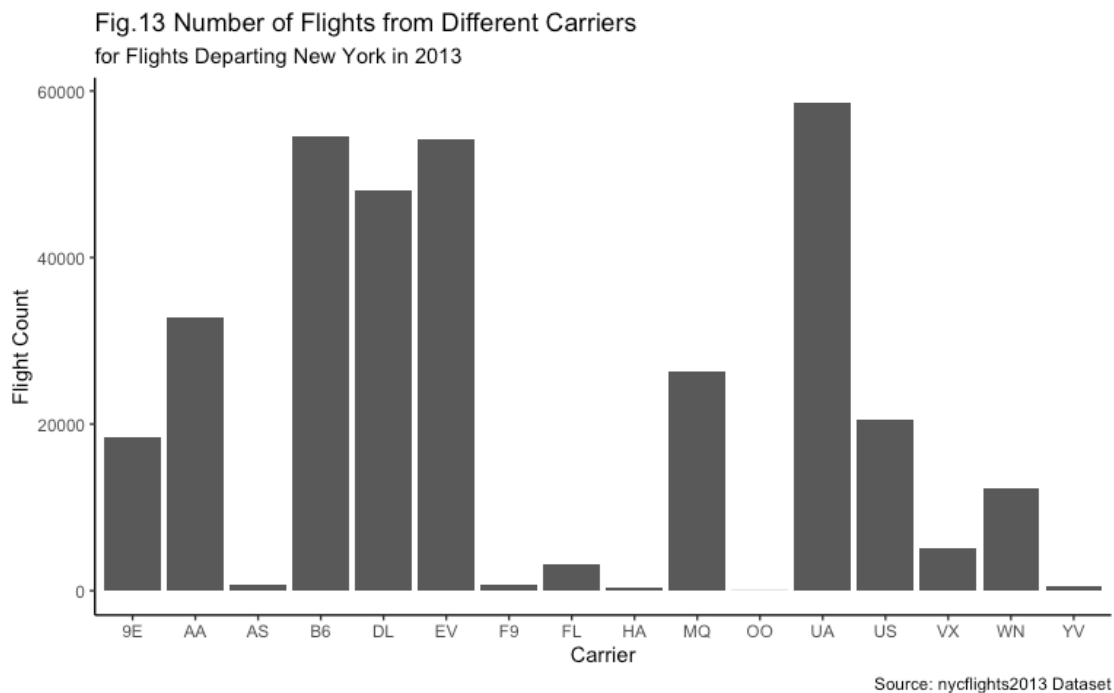
I started by preparing a summary dataset of flight count by different carrier and origins. After this, I calculated the flight percentages.

```
basePlot = ggplot(aes(x = carrier), data = flights) +  
  labs(subtitle = "for Flights Departing New York in 2013",  
       x = "Carrier",  
       y = "Flight Count",  
       caption = "Source: nycflights2013 Dataset") +  
  theme_classic()
```

I use this base plot to build on in the upcoming parts.

```
carrierBarplot = basePlot +  
  geom_bar(data = countData,  
          aes(y = flight_count),  
          stat = "identity") +  
  labs(title = "Fig.13 Number of Flights from Different Carriers")
```

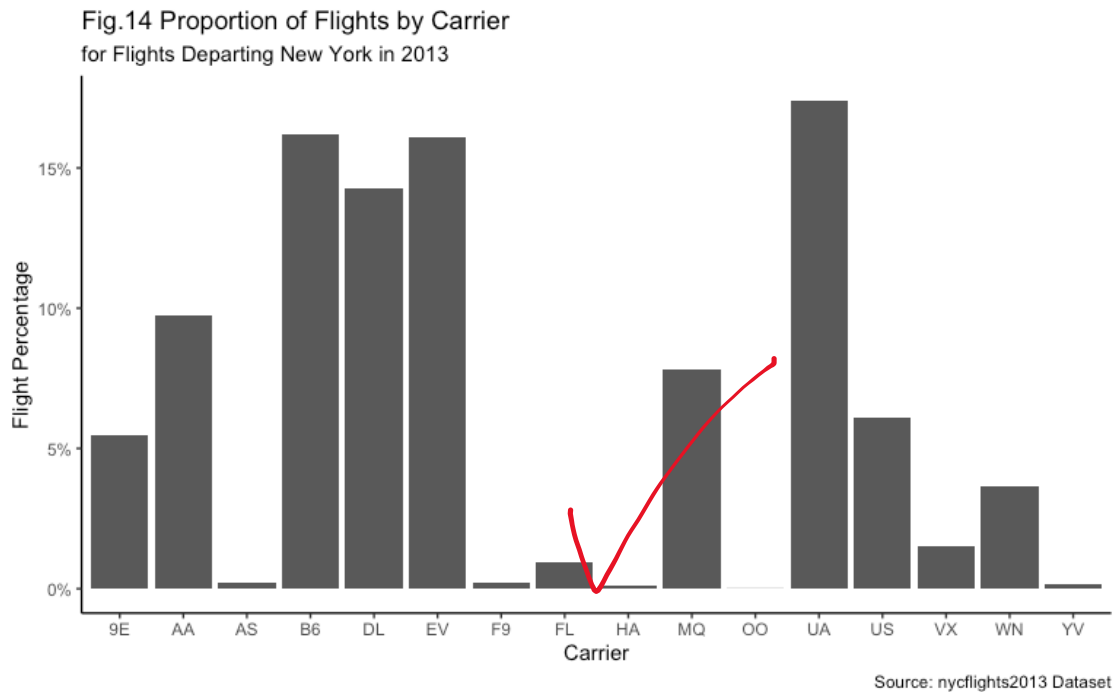
carrierBarplot



Make y-axis a percentage?

```
percentPlot = basePlot +  
  geom_bar(data = countData,  
    aes(y = flight_count / sum(flight_count),  
      stat = "identity") +  
  scale_y_continuous(labels = scales::percent) +  
  labs(title = "Fig.14 Proportion of Flights by Carrier",  
    y = "Flight Percentage")
```

percentPlot



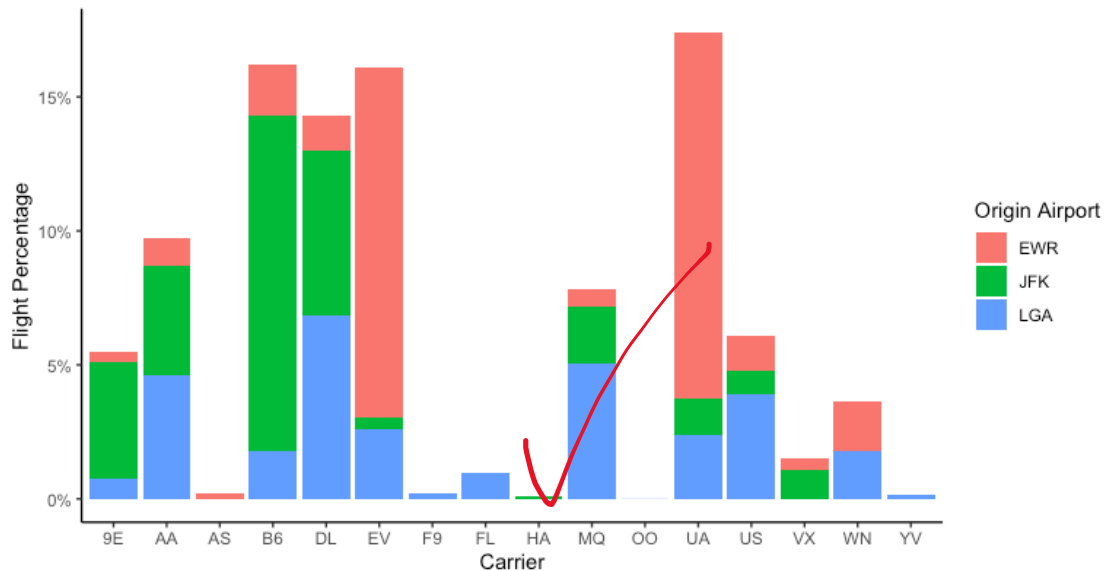
1.(j) [5 points]

Within the above plot (part i) add categorical variable origin to show stacked bar chart for the three airports (EWR, JFK, and LGA).

```
originPlot = basePlot +  
  geom_bar(data = countData,  
    aes(y = flight_count / sum(flight_count),  
      fill = origin),  
    stat = "identity") +  
  scale_y_continuous(labels = scales::percent) +  
  labs(title = "Fig.15 Proportion of Flights by Carrier",  
    subtitle = "for Flights Departing New York in 2013\nColored by  
Origin Airport",  
    y = "Flight Percentage",  
    fill = "Origin Airport")
```

originPlot

Fig.15 Proportion of Flights by Carrier  
for Flights Departing New York in 2013  
Colored by Origin Airport



Source: nycflights2013 Dataset

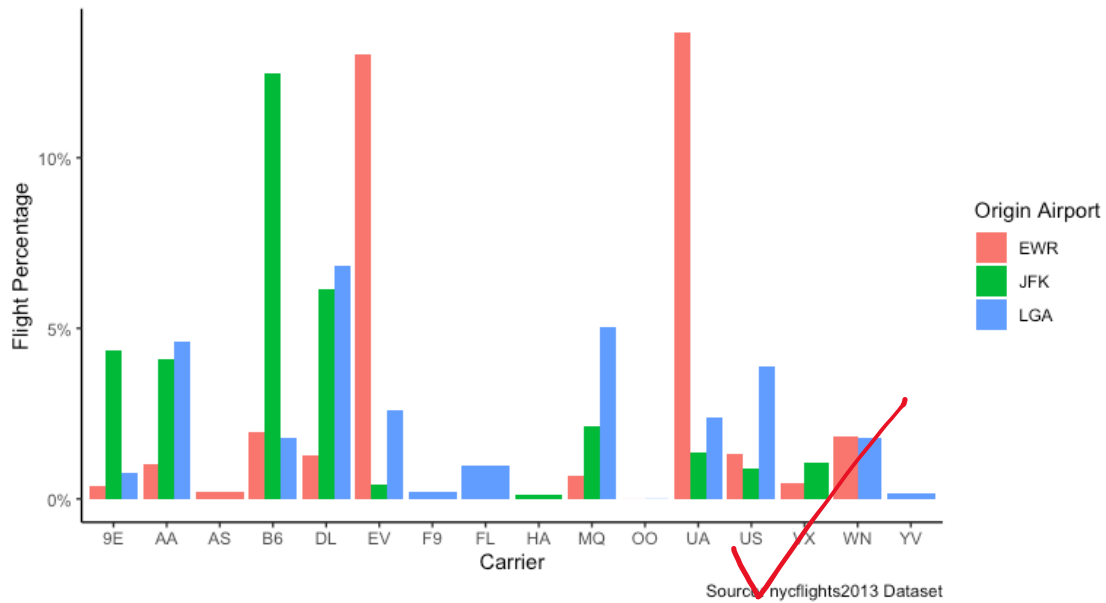
Then, show a plot with position dodge.

```
dodgePlot = basePlot +
  geom_bar(data = countData,
    aes(y = flight_count / sum(flight_count),
      fill = origin),
    stat = "identity", position = "dodge") +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Fig.16 Proportion of Flights by Carrier",
    subtitle = "for Flights Departing New York in 2013\nColored by
Origin Airport",
    y = "Flight Percentage",
    fill = "Origin Airport")
```

dodgePlot



Fig.16 Proportion of Flights by Carrier  
for Flights Departing New York in 2013  
Colored by Origin Airport



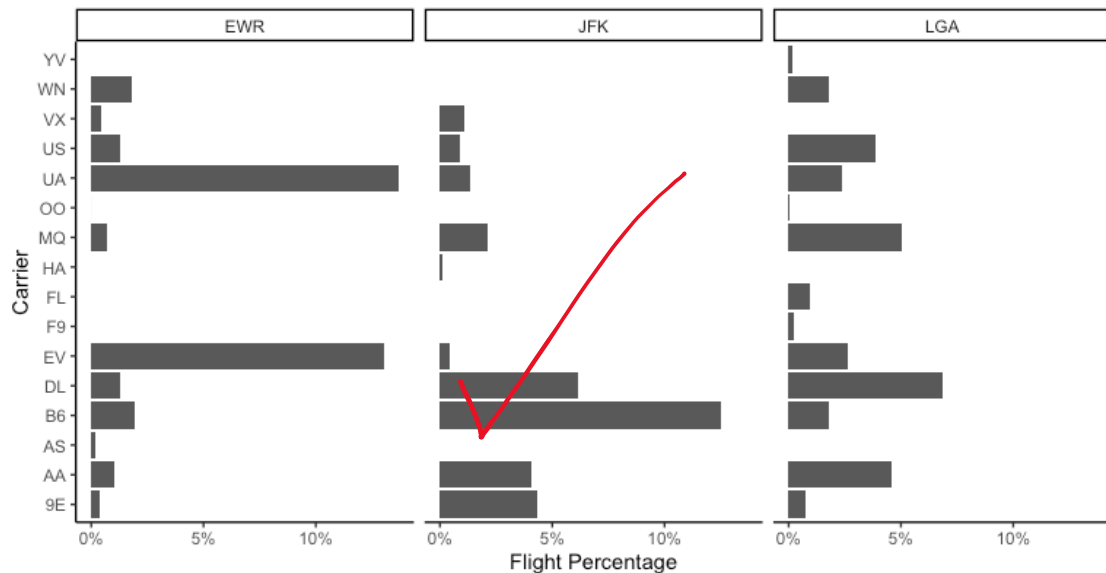
1.(k) [4 points]

Make bar plots in part (i) for each origin using facets.

```
facetCarrierPlot = percentPlot +
  facet_wrap(~origin) +
  coord_flip() +
  labs(title = "Fig.17 Proportion of Flights by Carrier",
        subtitle = "for Flights Departing New York in 2013\nFacetted by
Origin Airport")
```

facetCarrierPlot

Fig.17 Proportion of Flights by Carrier  
for Flights Departing New York in 2013  
Facetted by Origin Airport



Source: nycflights2013 Dataset

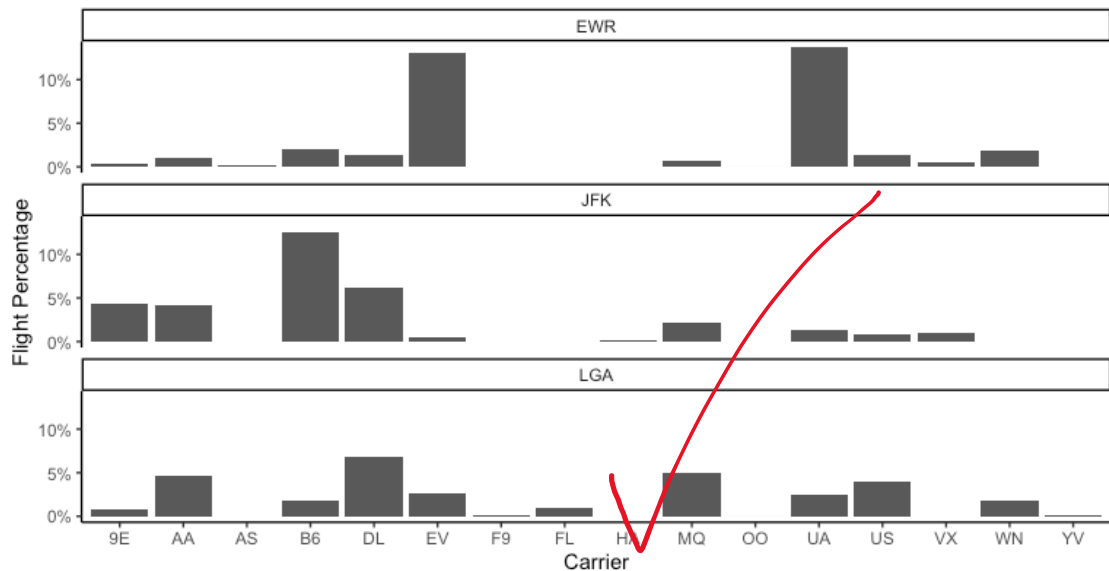
I flipped the axis as the x-axis variable labels were not fitting in my plot with a reasonable font size.

*Then, put them all in one column as facets.*

```
facetCarrierPlot = percentPlot +
  facet_wrap(~origin,
    nrow = 3, ncol = 1) +
  labs(title = "Fig.18 Proportion of Flights by Carrier",
    subtitle = "for Flights Departing New York in 2013\nFacetted by
Origin Airport")

facetCarrierPlot
```

Fig.18 Proportion of Flights by Carrier  
for Flights Departing New York in 2013  
Facetted by Origin Airport



Source: nycflights2013 Dataset

### 1.(l) [6 points]

Which carrier should you take if you want to have the shortest on average distance from the NYC flights information at 2013?

```
minAvgDistance = flights %>%
  group_by(carrier) %>%
  summarise(avg_distance = mean(distance, na.rm = TRUE)) %>%
  arrange(avg_distance) %>%
  slice(1)
```

minAvgDistance

```
## # A tibble: 1 × 2
##   carrier avg_distance
##   <chr>      <dbl>
## 1 YV          375.
```

To have the shortest on average distance from NYC, we can pick the Mesa Airlines. (how do you know the name?)

Which carrier should you take if you want to have the shortest on average arrival delay from the NYC flights information at 2013?

```
minAvgArrDelay = flights %>%
  group_by(carrier) %>%
  summarise(avg_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(avg_arr_delay) %>%
  slice(1)
```

```
minAvgArrDelay
```

```
## # A tibble: 1 × 2
##   carrier avg_arr_delay
##   <chr>      <dbl>
## 1 AS        -9.93
```

To have the shortest average arrival delay, I would pick the Alaska Airlines which has nearly a -10 minutes of an average arrival delay.

### 1.(m) [5 points]

*Suppose you want to go to St. Louis (STL) in July, and you don't care about the date, which airport should you fly from in terms of shortest airtime? And which carrier would you choose?*

```
flightsJulySTL = flights %>%
  filter(month == 7, dest == "STL")

flightsJulySTLAirportCarrier = flightsJulySTL %>%
  group_by(origin, carrier) %>%
  summarise(median_airtime = median(air_time, na.rm = TRUE)) %>%
  arrange(median_airtime)
```

```
flightsJulySTLAirportCarrier
```

**why median?-1**

```
## # A tibble: 4 × 3
## # Groups:   origin [2]
##   origin carrier median_airtime
##   <chr>   <chr>      <dbl>
## 1 EWR    WN          123
## 2 EWR    EV          125
## 3 LGA    AA          125
## 4 LGA    WN          126
```

To have the shortest air time possible in my flight to St. Louis in July, I would take a plane from the Newark airport via the Southwest Airlines. It has a slightly lower median air-time compared to other flights via different carriers and origin airports.

Note: I was unsure if I should group carrier and origin together or separately (choosing the best airport and the best carrier separately) but I ended up doing them together as it made more sense for me.

## Problem 2 - [15 points]

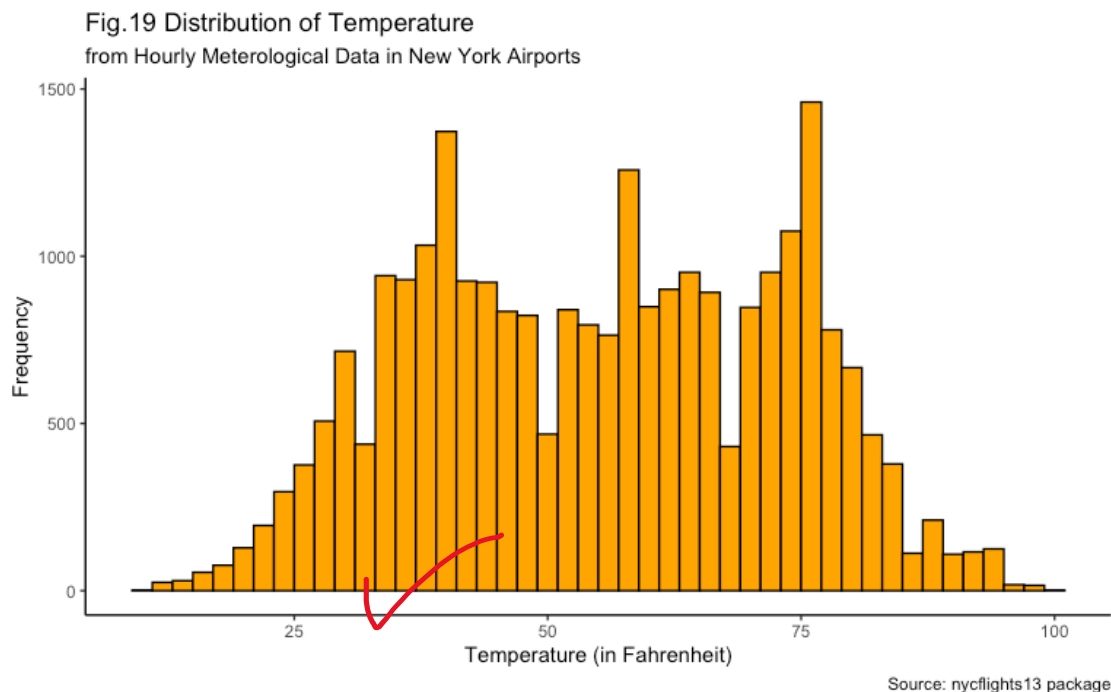
Using the weather data from the package `nycflights13` package do the following:

2.(a) [4 points]

Make a histogram for variable `temp`. Try options for different arguments within `geom_histogram()` to make the graph better.

```
temperatureHist = ggplot(data = weather,
                          aes(x = temp)) +
  geom_histogram(binwidth = 2,
                color = "black",
                fill = "orange") +
  labs(title = "Fig.19 Distribution of Temperature",
       subtitle = "from Hourly Meterological Data in New York Airports",
       x = "Temperature (in Fahrenheit)",
       y = "Frequency",
       caption = "Source: nycflights13 package") +
  theme_classic()
```

temperatureHist



*What do you observe?*

We can see that, the data shows a multi-modal distribution in a binwidth of 2. The distribution of Temperature in New York Airports has 3 peaks around 40, 55 and 75

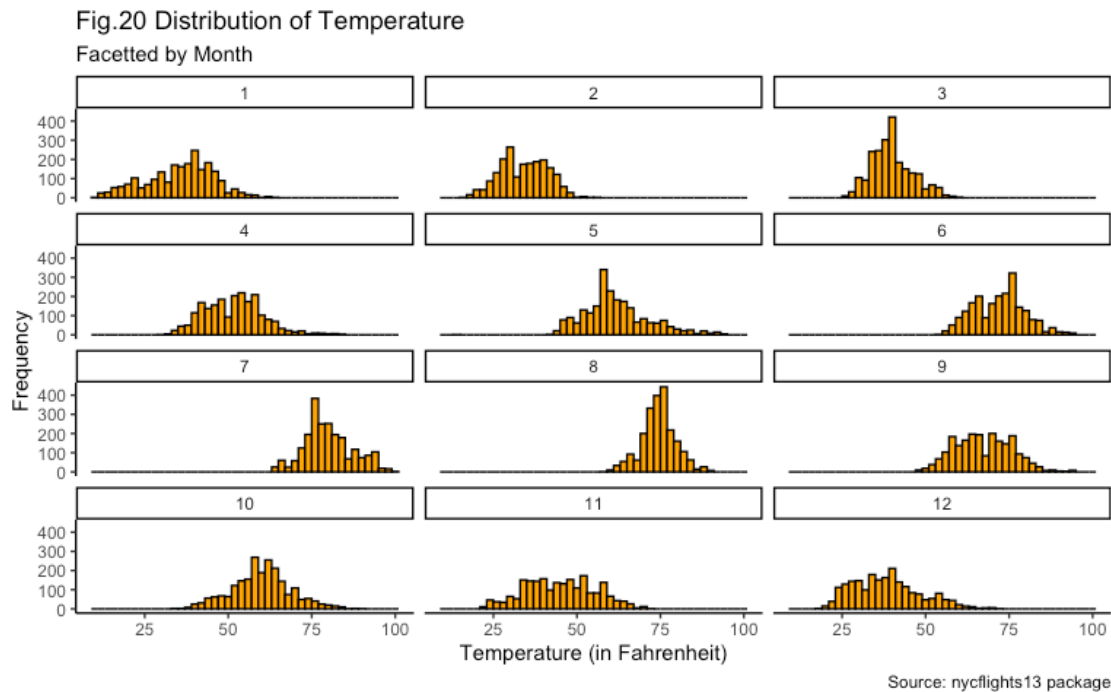
Fahrenheit. We can see that, as we go to extreme temperature values (in both directions), the temperature frequencies decrease. We can also see that the data is well-spread around the range (without obvious one-sided skewness) which points out that New York City experiences all seasons. The obvious peaks can point out the most common temperatures in Winter (left-most), Spring (middle), and Summer (right-most).

## 2.(b) [4 points]

*Suppose we were interested in looking at how the histogram of hourly temperature recordings at the three NYC airports we saw in (a) differed in each month. To do that make a facet plot which splits the histogram in (a) by the 12 possible months in a given year. In other words, make histograms of temp for each month separately using facets for month. Make the facets with four rows and three columns. Add both range fixed and range free plots in the facets.*

```
temperatureHistFixedFacet = temperatureHist +
  facet_wrap(~ month,
    nrow = 4, ncol = 3) +
  labs(title = "Fig.20 Distribution of Temperature",
    subtitle = "Facetted by Month")
```

temperatureHistFixedFacet



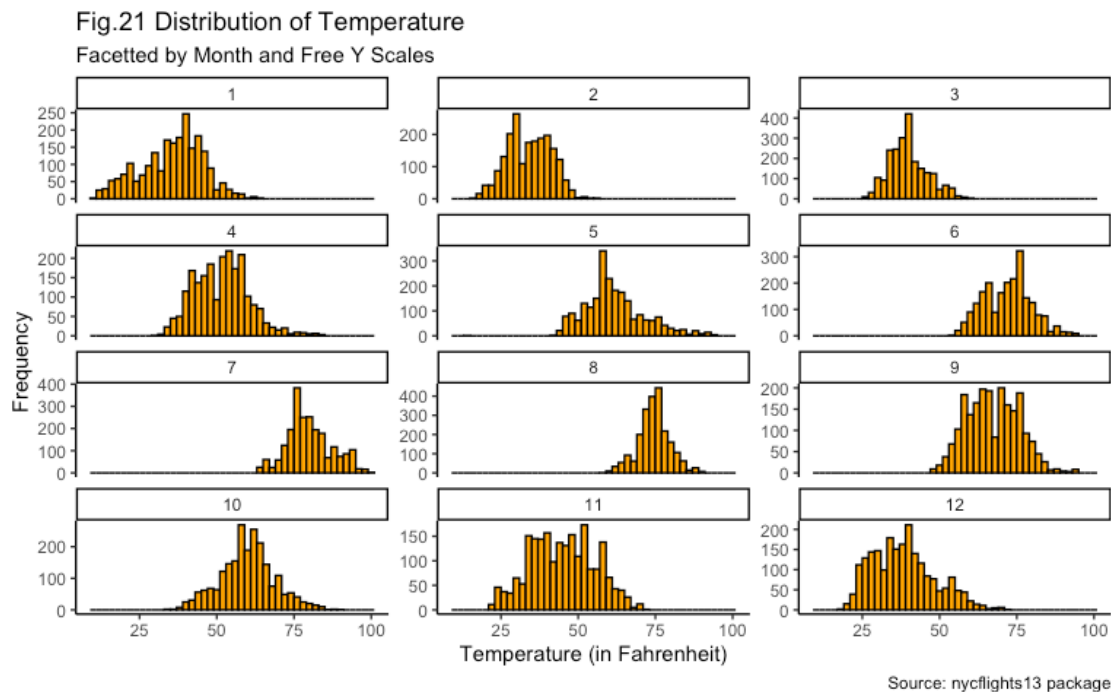
*It would be informative to have 1, 2, etc replaced by Jan., Feb, and make y-axis a % -1*

```

temperatureHistFreeFacet = temperatureHist +
  facet_wrap(~ month,
             nrow = 4, ncol = 3,
             scales = "free_y") +
  labs(title = "Fig.21 Distribution of Temperature",
       subtitle = "Facetted by Month and Free Y Scales")

```

temperatureHistFreeFacet



*It would be informative to have 1, 2, etc replaced by Jan., Feb, and make y-axis a %*

## 2.(c) [7 points]

Convert the numerical variable month into a factor categorical variable to make a comparative boxplot for the temperatures in 12 months.

```

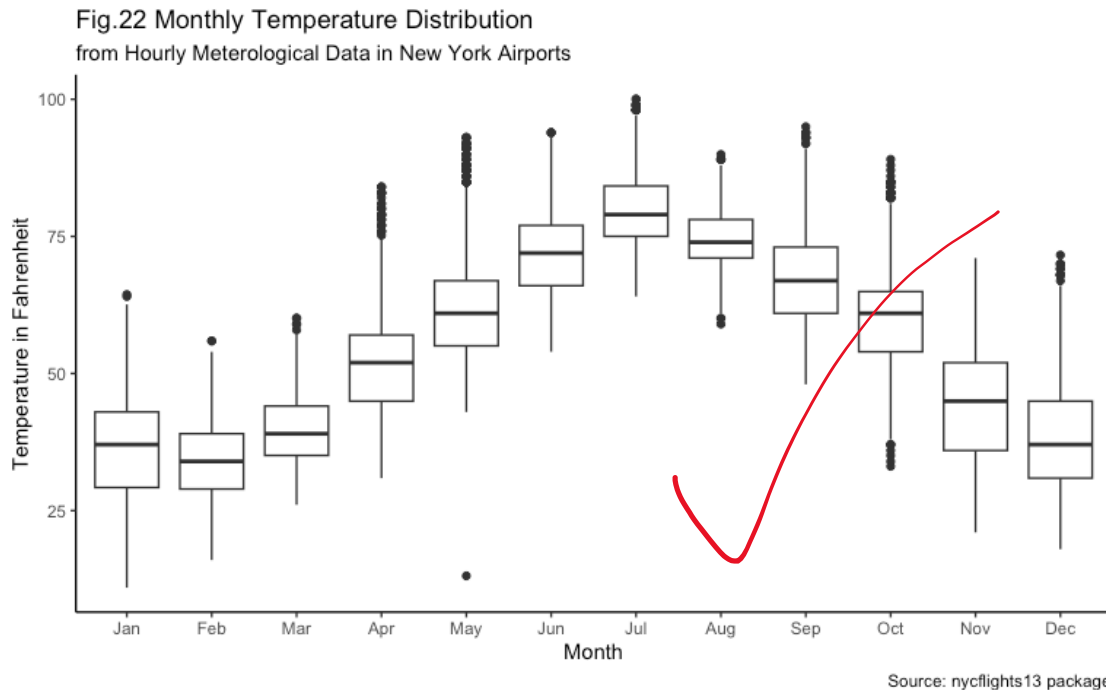
weather$month = factor(weather$month,
                       levels = c(1:12),
                       labels = c("Jan", "Feb", "Mar",
                                  "Apr", "May", "Jun",
                                  "Jul", "Aug", "Sep",
                                  "Oct", "Nov", "Dec"))

temperatureBoxplot = ggplot(data = weather,
                           aes(x = month,
                               y = temp)) +
  geom_boxplot() +
  labs(title = "Fig.22 Monthly Temperature Distribution",
       subtitle = "from Hourly Meteorological Data in New York Airports",

```

```
x = "Month",
y = "Temperature in Fahrenheit",
caption = "Source: nycflights13 package") +
theme_classic()
```

temperatureBoxplot



*What does the dot at the bottom of the plot for May correspond to? Explain what might have occurred in May to produce this point.*

The dot at the bottom of the plot for May is obviously an *outlier* point in our data which represents an extremely cold temperature. This could happen because of an unusual geological event on 2013, a late-spring cold burst, or simply a sensor error in the measuring system (which is more likely in my opinion).

*Which months have the highest variability in temperature? What reasons can you give for this?*

From the boxplot, we can see that *April, October, and November* has the highest variability in temperature. We can see this from the the size of whiskers and the interquartile ranges. I think that, the reason for this increased variability is the seasonal changes where the day-by-day temperature difference can be high, with significantly cold and hot days in the same month.



## Problem 3 - [15 points]

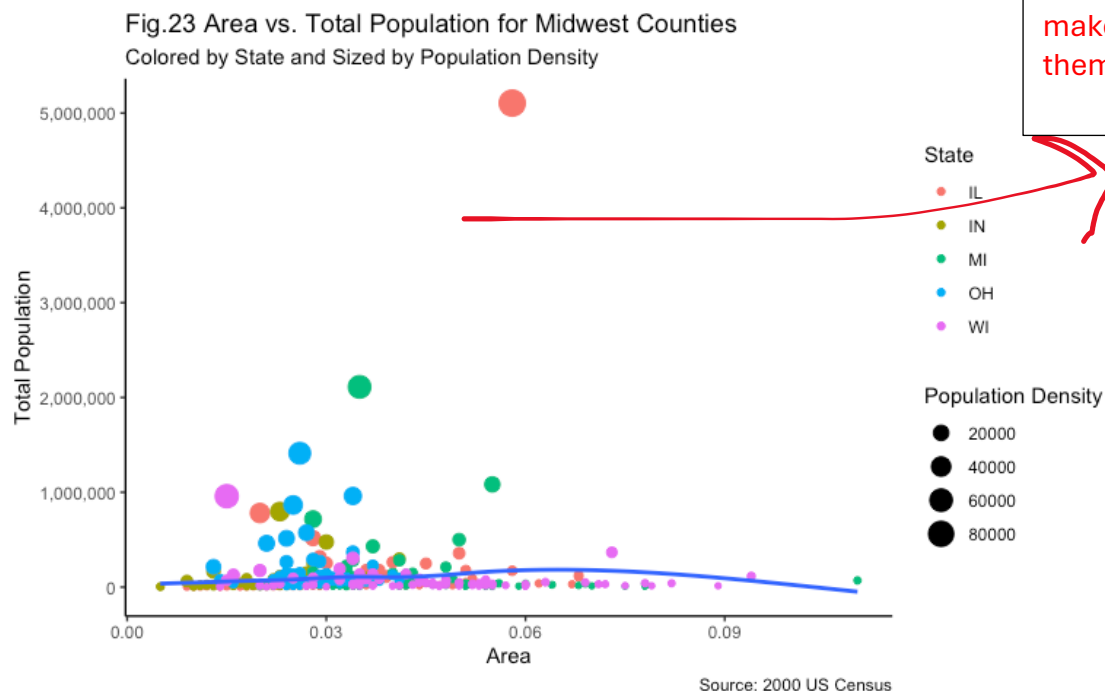
For the midwest data in ggplot2 package, answer the following:

### 3.(a) [5 points]

Make a scatter plot between area (x-axis) and population total (y-axis) with dots colored by states and dot sizes varied by popdensity. Include the fitted loess (R's default) line without the 95% confidence band.

```
areaPopulationPlot = ggplot(data = midwest,
                             aes(x = area,
                                 y = poptotal)) +
  geom_point(aes(color = state,
                 size = popdensity)) +
  scale_y_continuous(labels = scales::comma) +
  geom_smooth(se = FALSE) +
  labs(title = "Fig.23 Area vs. Total Population for Midwest Counties",
       subtitle = "Colored by State and Sized by Population Density",
       x = "Area",
       y = "Total Population",
       color = "State",
       size = "Population Density",
       caption = "Source: 2000 US Census") +
  theme_classic()
```

areaPopulationPlot



Anytime your graph shows outliers, we got to make a graph without them -1

### 3.(b) [10 points]

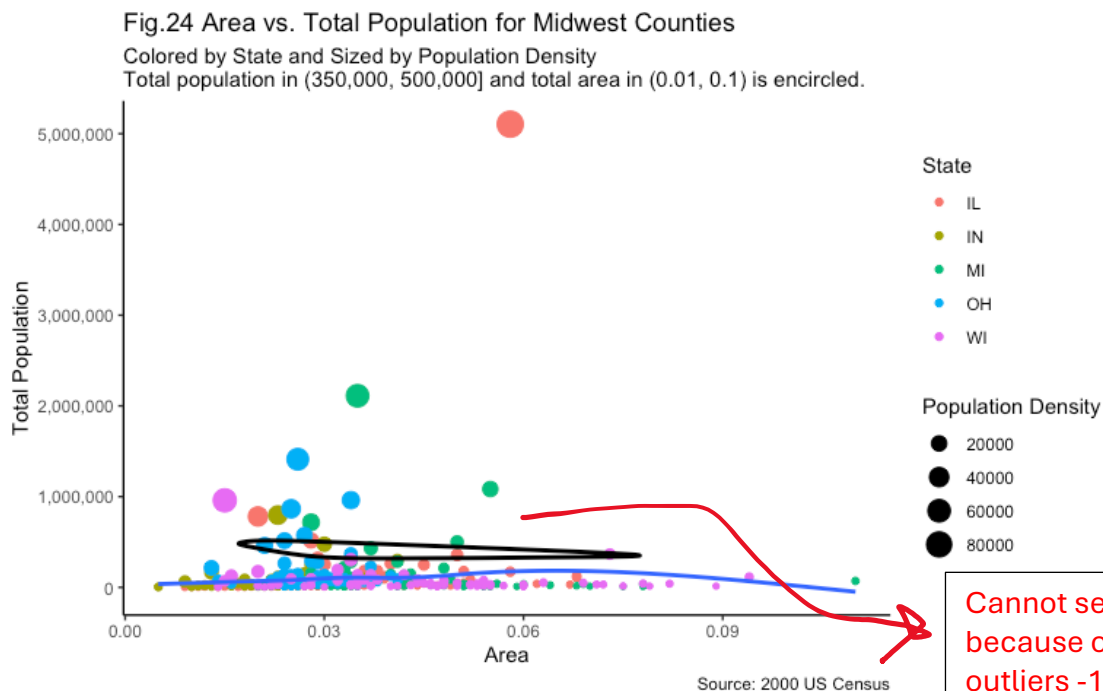
Make a subset with total population between (350,000, 500,000] and area between (0.01 and 0.1).

```
midwestSubset = midwest %>%  
  filter(350000 < poptotal,  
         poptotal <= 500000,  
         0.01 < area,  
         area < 0.1)
```

Using this subset and function `geom_encircle()` (it is in the package `ggalt`), encircle the points in scatter plot from (a) who are in this subset. In other words, using the new dataframe with only the points (rows) of interest, you are highlighting (encircling) the points in scatterplot made from the midwest data in part (a).

```
areaPopulationPlotEncircle = areaPopulationPlot +  
  geom_encircle(data = midwestSubset,  
               aes(x = area,  
                   y = poptotal),  
               size = 3) +  
  labs(title = "Fig.24 Area vs. Total Population for Midwest Counties",  
        subtitle = "Colored by State and Sized by Population Density\nTotal  
population in (350,000, 500,000] and total area in (0.01, 0.1) is  
encircled.")
```

areaPopulationPlotEncircle



## Problem 4 - [20 points]

Using the dataset `movie_data_2011` from `UsingR` package (see help file for more details) answer the following:

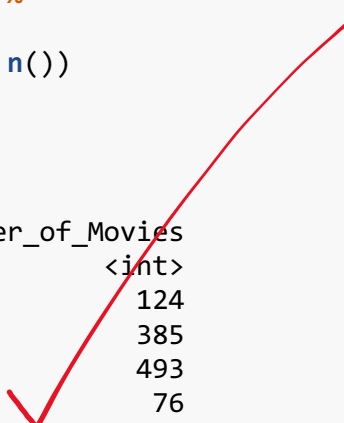
### 4.(a) [5 points]

*Report the genre types with the number of movies in each genre.*

```
genreSummary = movie_data_2011 %>%  
  group_by(Genre) %>%  
  summarise(Number_of_Movies = n())
```

genreSummary

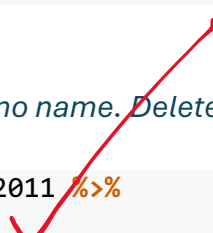
```
## # A tibble: 14 × 2  
##   Genre                Number_of_Movies  
##   <fct>                <int>  
## 1 ""                    124  
## 2 "Action"              385  
## 3 "Adventure"           493  
## 4 "Black Comedy"        76  
## 5 "Comedy"              674  
## 6 "Documentary"         1347  
## 7 "Drama"               1894  
## 8 "Horror"              171  
## 9 "Musical"             21  
## 10 "Romantic Comedy"    247  
## 11 "Thriller/Suspense"  478  
## 12 "Western"            20  
## 13 "Multiple Genres"    40  
## 14 "Concert/Performance" 45
```



### 4.(b) [5 points]

*In part(a) there is one Genre with no name. Delete the movies which belong to this no name genre.*

```
filteredMovies = movie_data_2011 %>%  
  filter(Genre != "")
```



**Note: For rest of the parts use the cleaned data from 4.(b)**

### 4.(c) [5 points]

*Prepare a summary of the gross per current weekend for each genre including mean, median and sd of gross for each genre.*

```
grossSummary = filteredMovies %>%  
  group_by(Genre) %>%  
  summarise(Mean_Gross = mean(Gross, na.rm = TRUE),  
            Median_Gross = median(Gross, na.rm = TRUE),
```

```
SD_Gross = sd(Gross, na.rm = TRUE))
```

```
grossSummary
```

```
## # A tibble: 13 × 4
```

##	Genre	Mean_Gross	Median_Gross	SD_Gross
##	<fct>	<dbl>	<dbl>	<dbl>
##	1 Action	3742803.	110292	11052610.
##	2 Adventure	3487573.	253066	11372828.
##	3 Black Comedy	334394.	3032.	1304534.
##	4 Comedy	1892501.	78223	5751765.
##	5 Documentary	33369.	4997	188702.
##	6 Drama	550946.	13142.	3813355.
##	7 Horror	1813467.	72569	5266815.
##	8 Musical	1915922.	70026	4065828.
##	9 Romantic Comedy	1603544.	93155	3976037.
##	10 Thriller/Suspense	1377117.	51045	3780869.
##	11 Western	2789862.	359614	4204241.
##	12 Multiple Genres	24709.	2016.	71023.
##	13 Concert/Performance	1618074.	36472	4965764.

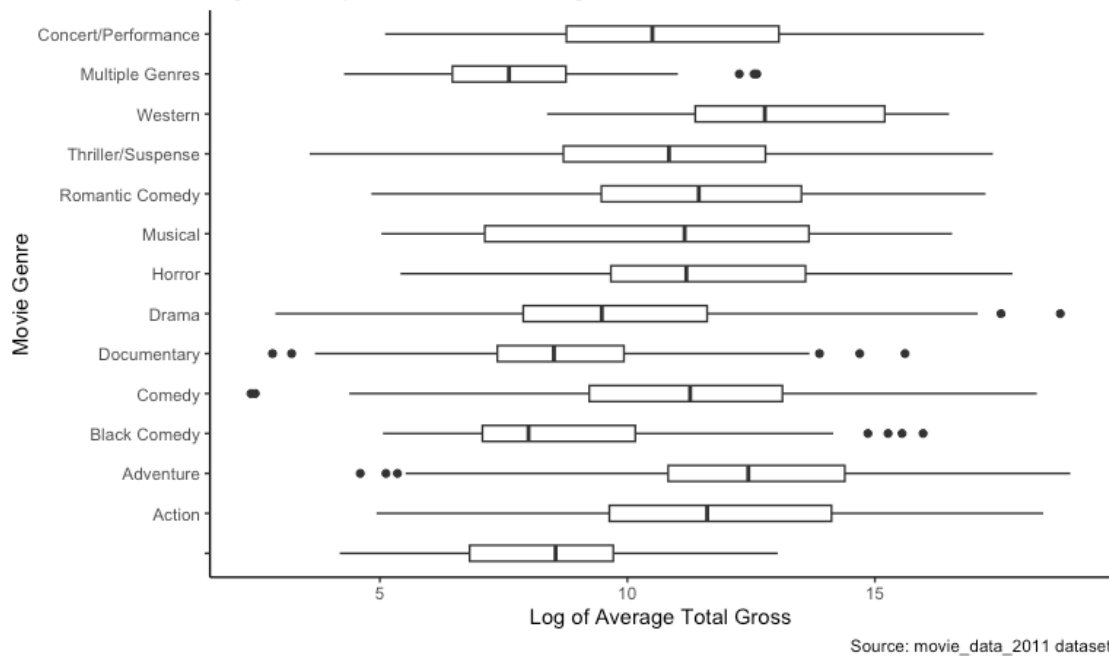
#### 4.(d) [5 points]

*Make comparative boxplots for total gross to the date for the movie's genres.*

```
grossBoxplot = ggplot(data = movie_data_2011,  
  aes(x = Genre,  
    y = log(Gross))) +  
  geom_boxplot(width = 0.4) +  
  scale_y_continuous(labels = scales::comma) +  
  labs(title = "Fig.25 Comparison of the Average Total Gross for Different  
Movie Genres",  
    x = "Movie Genre",  
    y = "Log of Average Total Gross",  
    caption = "Source: movie_data_2011 dataset") +  
  coord_flip() +  
  theme_classic()
```

```
grossBoxplot
```

Fig.25 Comparison of the Average Total Gross for Different Movie Genres



Not asked log of total gross so doing that without any reasoning is not good! -1.5

I flipped the axis as the x-axis variable labels were not fitting in my plot with a reasonable font size. ✓

Name the top three genre in terms of high total gross.

```
top3AverageGross = movie_data_2011 %>%
  group_by(Genre) %>%
  summarise(MaxGross = max(TotalGross, na.rm = TRUE)) %>%
  arrange(desc(MaxGross)) %>%
  slice(1:3)
```

top3AverageGross

```
## # A tibble: 3 × 2
##   Genre      MaxGross
##   <fct>      <dbl>
## 1 Adventure 422772856
## 2 Action   352358779
## 3 Drama    275530738
```

From this tibble and the boxplot above, we can see that the top-3 genres with the high total gross is **Adventure**, **Action**, and **Drama**.

Do you agree with these based on the boxplot (corrected with total gross)?

## Bonus Problem - [5 points]

### Bonus.(i) [2.5 points]

A dataset named `df` has been read in R and missing values in this data have been read as `NA`. Which of the following codes will not give the number of missing values in each column?

i.(A) `colSums(is.na(df))`

i.(B) `apply(is.na(df),2,sum)`

i.(C) `sapply(df,function(x) sum(is.na(x)))`

i.(D) `table(is.na(df))`

```
df = data.frame(
  name = c("Derin", "Johnny", NA, "Muhammed"),
  age = c(20, 19, 30, NA),
  score = c(NA, NA, 100, 100)
)
```

```
colSums(is.na(df))
```

```
## name age score
##    1    1     2
```

```
apply(is.na(df), 2, sum)
```

```
## name age score
##    1    1     2
```

```
sapply(df, function(x) sum(is.na(x)))
```

```
## name age score
##    1    1     2
```

```
table(is.na(df))
```

```
##
## FALSE TRUE
##     8   4
```

In my example dataframe, there was one missing value in the name column, one missing value in the age column and two missing values in the score column. The first three options were able to give me these values as expected.

**i.(A)** sums the True values in each column (using `colSums`). These values are obtained using the `is.na` function which returns TRUE for NA values. The result is as expected.

**i.(B)** uses the `apply` function to sum the values for each column. The function sums the values by column using the second parameter in the function call. The result is as expected.

i.(C) uses the `sapply` function to each column of the dataframe. `sapply` applied the same function (in this case `sum(is.na(x))`) to each column of `df`. The result is as expected.

i.(D) would be correct if the question asked for the count of NA values in the data which would be 4. `is.na` returns TRUE for NA values, and the `table` function returns the count of TRUE and FALSE in the result of `is.na`. The result is not correct.

In this case, the answer is i.(D)

### Bonus.(ii) [2.5 points]

*Which of the following command will help us remove the duplicate rows in the dataframe named `df`?*

ii.(A) `df[!duplicated(df),]`

ii.(B) `unique(df)`

ii.(C) `dplyr::distinct(df)`

ii.(D) All of the above

ii.(E) None of the above

```
df = data.frame(
  name = c("Derin", "Johnny", "Derin", "Muhammed"),
  age = c(20, 19, 20, 18),
  score = c(97, 98, 97, 100)
)
```

```
df[!duplicated(df),]
```

```
##      name age score
## 1  Derin  20    97
## 2  Johnny 19    98
## 4 Muhammed 18   100
```

```
unique(df)
```

```
##      name age score
## 1  Derin  20    97
## 2  Johnny 19    98
## 4 Muhammed 18   100
```

```
dplyr::distinct(df)
```

```
##      name age score
## 1  Derin  20    97
## 2  Johnny 19    98
## 3 Muhammed 18   100
```

In my example dataframe, the first and the third rows were duplicated. All the given options from ii.(A) to ii.(C) was able to remove the duplicate rows from the dataframe `df`.

**ii.(A)** basically checks the duplicated rows using the `duplicated` function which would return `FALSE` for a unique row and `TRUE` for duplicates. As we are aiming for filtering the duplicates, we reverse it using `!` which makes unique rows `TRUE`. After this, we can just use this array of logical values to select specific rows.

For **ii.(B)**, the `unique` function directly extracts the unique values in a data frame. I confirmed this functionality with the built-in R help using `?unique`.

For **ii.(C)**, the `distinct` function in `dplyr` works very similar to the `unique` function from the previous part (ii.(B)) and it returns the unique rows in the data frame.

In this case, the answer would be **ii.(D)** as all the examples successfully return the unique rows.

TS