Deliverable_3

Florek J. Bretton

12/10/2020

Introduction:

My choice in data has to do with flight delays. I have personally been on many flights that alwasy seem to have some sort of delay. One of the craziest delays pertained to a simple light that had gone out which caused us to completely de-plane and took hours to find a new plane. The data represented below is a collection of 2 sperate data sets. The first one contains flight delays for the entire year of 2008. This data includes why the plain was delayed along with specific data relating to the airport and carier information.
The second data set is a collection of all weather data for 2008 at the San Jose International Airport. This data contains info such the wind speeds and temperatures as well as the amount of rain fall.
The reason I did not do much with the first dataset is that I have switched my data science question listed below to a new datascience question with more potential. In the future I hope to link both my datasets by date and thus gather information about why flights were delayed.

Data Science Questions Trying to be answered:

My original Data Science question i set to answer was: Can we predict and shorten delay times by changing airlines at specific airports?
This turned out to be mich more complex than originally anticipated and due to the nature of airline companies working to keep there internal data secret I was unable to get ahold of the required data to compare Carrier Delays.
Thus My new data science questions im seeking to answer is can I predict the weather conditions at the San Jose Interantional Airport?

Flight Delays DATA

This dataset is on the flight delays in the year 2008. This dataset includes the exact departure times along with the schedule departure times. There is also listings of delays or cancellations allowing me to analyze the average departure time per airport.

```
df <- read.csv("DelayedFlights.csv")
```

Data Set Variables/Units

- **Year:** 2008 (Integer)
- **Month:** 1 - 12 (Integer)
- **DayofMonth:** 1 - x (x = number of days in month) (Integer)

- **DayofWeek:** 1 - 7 (1 = Monday, 7 = Sunday) (Integer)
- **DepTime:** Exact departure time in format HHMM.SS (H = hour, M = minutes, S = seconds) (Double)
- **CRSDepTime:** The scheduled Arrival time in format HHMM (H = hour, M = minutes) (Integer)
- **ArrTime:** The exact Arrival time in format HHMM.SS (H = hour, M = minutes, S = seconds) (Double)
- **CRSArrTime:** The scheduled Arrival time in format HHMM (H = hour, M = minutes) (Integer)
- **UniqueCarrier:** The Carrier code for flight company. (EX AA = American Airlines) (Factor)
- **FlightNum:** Flight Number (Integer)
- **TailNum:** Tail Number (Factor)
- **ActualElapsedTime:** Exact Flight time in minutes (Double)
- **CRSElapsedTime:** The scheduled or estimated flight time in minutes (Double)
- **AirTime:** Time in air in minutes (Double)
- **ArrDelay:** The arrival delay in minutes. (Double)
- **DepDelay:** The Departure delay in minutes. (Double)
- **Origin:** IATA code for origin airport (Factor)
- **Dest:** IATA code for destination airport (Factor)
- **Distance:** Flight distance in miles (Integer)
- **TaxiIn:** Time to taxi to gate. Represented in minutes (Double)
- **TaxiOut:** Time to taxin to runway. Represented in minutes (Double)
- **Cancelled:** 0 = NO 1 = YES (Integer)
- **CancellationCode:** Cancellation Reason A = Carrier, B = weather, C = NAS(National Airspace System), D = security (Factor)
- **Diverted:** 0 = NO, 1 = YES (Integer)
- **CarrierDelay:** Represented in minutes (Delay ex: bird strike, baggage loading) (Double)
- **WeatherDelay:** Represented in minutes (Delay ex: snow storm, de-icing) (Double)
- **NASDelay:** Represented in minutes (Delay ex: Airport operations, heavy traffic) (Double)
- **SecurityDelay:** Represented in minutes (Delay ex:Passenger screening) (Double)
- **LateAircraftDelay:** Represented in minutes (Delay ex: head wind) (Double)

Definitions:

- **NAS:** This stands for the National Airspace System. This system includes every airport, services, airspace rules, personell, procedures, and equipment. This sysem also includes the military. Reasons for this system causing a delay would most likely be due to a military procedure like training in an area casuing a delay to the flight path to avoid the training.

Sumary of Delays data

Looking at the summarry below some interesting things to note are the negative values shown. At first glance this seems incorrect but after looking more indepth these negative values in the delay columns are simply saying that the airplane is actually ahead of schedule either due to every passenger boarding early, or something simple like a tailwind during the flight.

```
summary(df$ArrDelay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -109.0     9.0    24.0    42.2    56.0  2461.0    8387
```

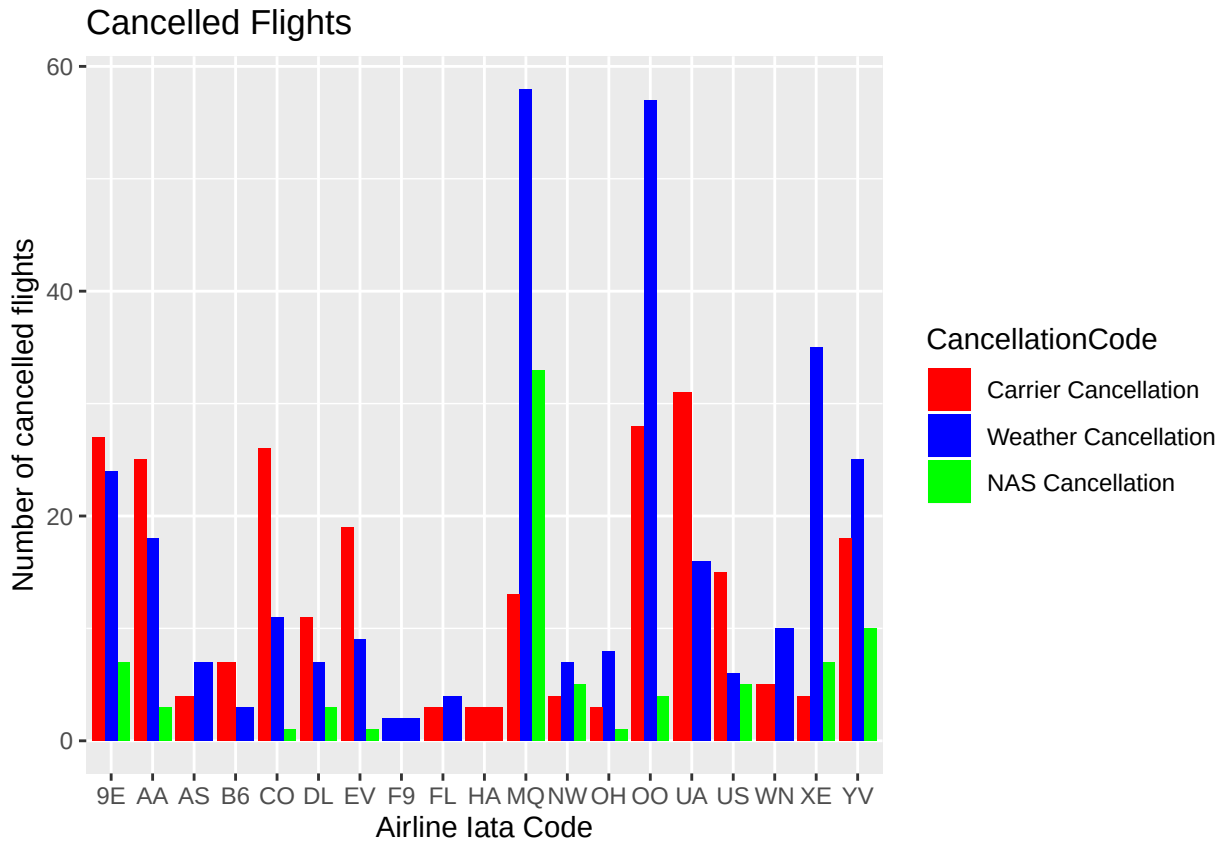
```
summary(df$DepDelay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   12.00   24.00   43.19   53.00 2467.00
```

Cancelled flights Per Origin Airport

This is showing the number of cancelled flights per carrier colord by the cancellation code. Notice there wasnt a single security cancellation this can most likely be due to the NAS cancellations being almost the same thing. Also notice that the Carrier cancellations were quite high for United Airlines(UA), American Airlines(AA), Skywest (OO), and Continental Airlines(CO). Notice also that there is a significant number of cancelled flight with the weather cancellation code for Skywest(OO) and Envoy Airlines(MQ).

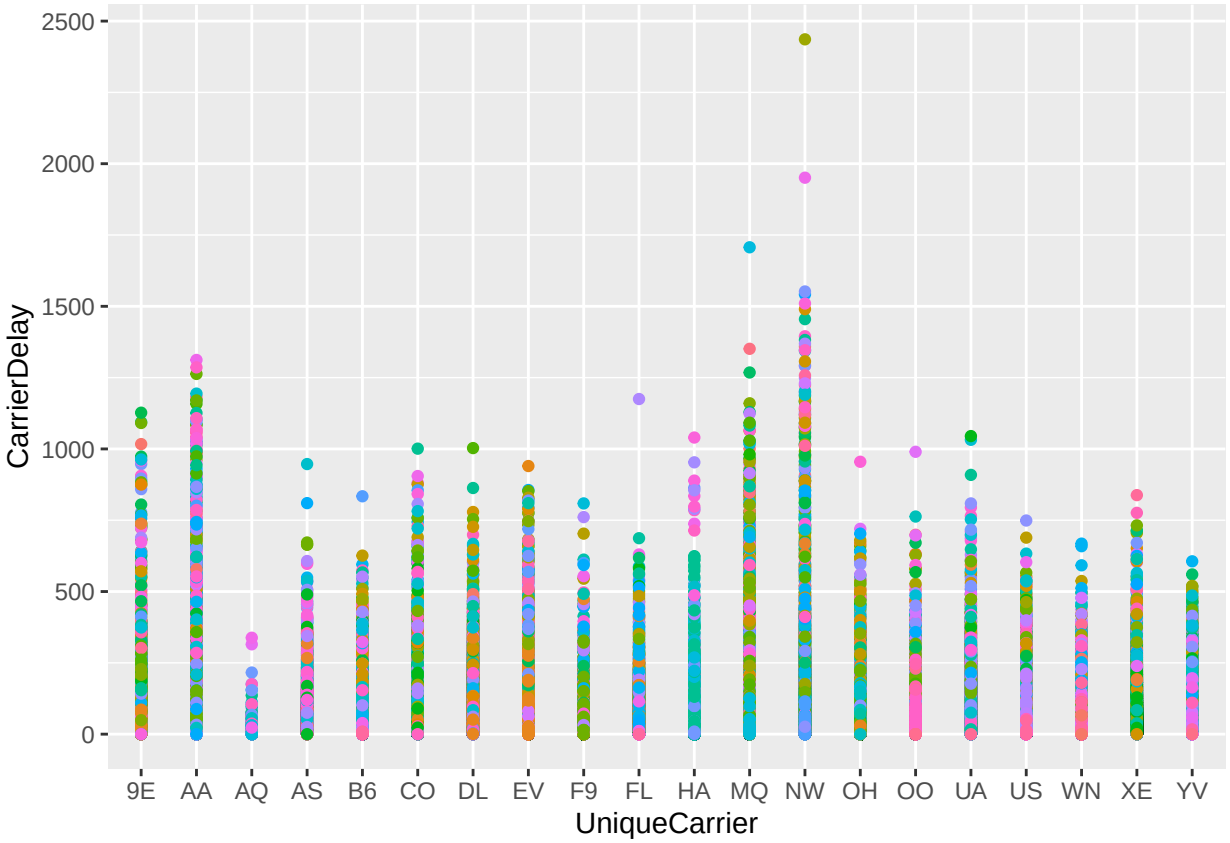
```
ggplot(data = filter(df, CancellationCode!= "N"), aes(x = UniqueCarrier )) +
 geom_bar(aes(fill = CancellationCode), show.legend = TRUE, position = "dodge") +
  labs(title = "Cancelled Flights", x = "Airline Iata Code", y = "Number of cancelled flights") +
 scale_fill_manual(labels = c("Carrier Cancellation", "Weather Cancellation", "NAS Cancellation"), valu
```



Carrier Delays Per Airline

As you can see below are the Carrier Cause delays. The Delays are all colored by Origin airport suggesting that most of these delays have to do with either some sort of flight clearance or gate clearance. One interesting thing to lookat is how Northwest airlines seems to have a rather high carrier delay with American Airlines and American Eagle airlines following closly behind.

```
ggplot(data = df, aes(x = UniqueCarrier, y = CarrierDelay)) +
  geom_point(aes(color = Origin), show.legend = FALSE, na.rm = TRUE)
```



Summary Departure Delays

This simply summarizes the Departure Delays data showing the minimum departure delay for the entire year was 6 minutes. This is interesting as not a single flight departed on time showing some sort of inefficiency. CRSDepTime is the estimated departure time.

```
selectedCol <- c(10, 7, 6, 17, 18)
summary(df[selectedCol])
```

```
##  UniqueCarrier      CRSDepTime      DepTime        DepDelay
##  WN     :377602   Min.   :   0   Min.   :   1   Min.   :   6.00
##  AA     :191865   1st Qu.:1135   1st Qu.:1203   1st Qu.:  12.00
##  MQ     :141920   Median :1510   Median :1545   Median :  24.00
##  UA     :141426   Mean   :1467   Mean   :1519   Mean   :  43.19
##  OO     :132433   3rd Qu.:1815   3rd Qu.:1900   3rd Qu.:  53.00
##  DL     :114238   Max.   :2359   Max.   :2400   Max.   :2467.00
##  (Other):837274
##      Origin
##  ATL    : 131613
##  ORD    : 125979
##  DFW    :  95414
##  DEN    :  74323
##  LAX    :  58772
##  IAH    :  56847
```

```
##  (Other):1393810
```

Summary Arrival Delays

This data summarization is for the arrival delays. Looking closly something to notice is that the arrival delays has a minimum of -109 minutes. This is saying that in the best conditions with a very minimal departure delay the flight actually arrived 109 minutes early. This can be due to a tail wind or the pilots pushing to plane faster.

```
selectedCol <- c(10, 8, 9, 16, 19)
summary(df[selectedCol])
```

```
##  UniqueCarrier       ArrTime       CRSArrTime      ArrDelay
##  WN     :377602   Min.   :   1   Min.   :   0   Min.   :-109.0
##  AA     :191865   1st Qu.:1316   1st Qu.:1325   1st Qu.:   9.0
##  MQ     :141920   Median :1715   Median :1705   Median :  24.0
##  UA     :141426   Mean   :1610   Mean   :1634   Mean   :  42.2
##  OO     :132433   3rd Qu.:2030   3rd Qu.:2014   3rd Qu.:  56.0
##  DL     :114238   Max.   :2400   Max.   :2400   Max.   :2461.0
##  (Other):837274   NA's   :7110                  NA's   :8387
##       Dest
##  ORD    : 108984
##  ATL    : 106898
##  DFW    :  70657
##  DEN    :  63003
##  LAX    :  59969
##  EWR    :  55861
##  (Other):1471386
```

Chaning of Data Science Question

When looking for my second data set is about the time I decided to switch to a new data science question. Though this has changed alot my goal is to try and bring both my datasets together in order to look at the weather on days there are cancellations or heavy delays at the San Jose International Airport. # San Jose International Airport Weather Data This data set came from almanac.com and was webscraped. The data represents the San Jose International Airport Weather conditions for the year 2008.

Webscrape

Web Scrape Web Data. Due to web server authentication settings the data will be loaded from csv file below as web server causes a time out during web scraping sometimes. This data is for the San Jose International Airport Weather.

```
data_date <- ymd(20080101)
weather_data <- tibble(Year = integer(), Month = integer(), Day = integer(), Minimum_Temp = character()
for(month in 1:5)
{
    for(i in 1:73)
```

```
  {
    df1 <- read_html(paste("https://www.almanac.com/weather/history/zipcode/95110/", data_date, sep = "
    html_table()

    df <- as_tibble(df1[[2]], .name_repair = "minimal")
    weather_data <- add_row(weather_data, Year = year(data_date), Month = month(data_date), Day = day(da
    data_date <- data_date + 1
    Sys.sleep(5)
    closeAllConnections()
    gc()
    }

  Sys.sleep(60)
  gc()
}
weather_data<-data.frame(lapply(weather_data, as.character), stringsAsFactors=FALSE)

write.csv(as_data_frame(weather_data),"/Users/brettflorek/Portfolio/San_Jose_2008_weather.csv", row.name
```

Load Pre-scraped Weather Data

This data was Pre-scraped from almanac.com for San Jose Internation Airport.

```
weather_data <- read.csv("San_Jose_2008_weather.csv")
```

Data Set Variables

List of all the variables of the San Jose International Airport Weather Data Set
- **Year:** 2008 (Integer)
- **Month:** Month 1-12 (Integer)
- **Day:** Day in month 1-31 respectively (Ineteger)
- **Minimum_Temp** Minimum temperature for that day in Farenheit (Factor)
- **Maximum_Temp** Maximum temperature for that day in Farenheit (Factor)
- **Total_Rain** Total amount of rain for that day in Inches (Factor)
- **Visibility** The visibility distance in miles (Factor)
- **Max_Wind_Speed** Maximum wind speed recorded that day in MPH (Factor)
- **Max_Wind_Gust** Maximum Wind Gust in MPH. Wind Speeds of 999MPH are electrical interference. (Factor)

Preparing data

Converted all factors to doubles and excluded all 999 MPH wind gusts as these are mostly caused by interference. Converted interference values to NA.

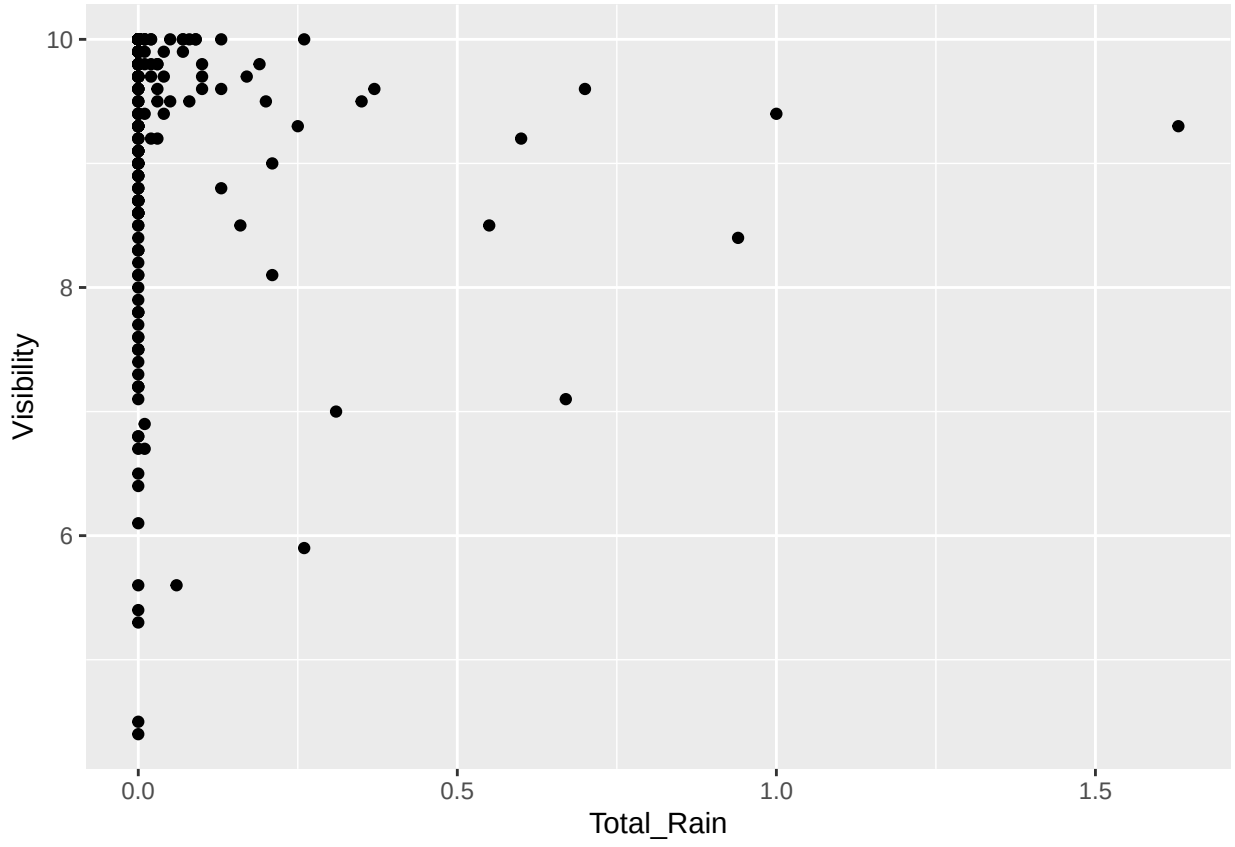
```
index <- c(1:365)
mutating <- weather_data
weather_data_doubles <- tibble(Year = integer(), Month = integer(), Day = integer(), Minimum_Temp = doul
for(i in index)
```

```
{
  min_temp_double <- as.double(substr(as.character(mutating$Minimum_Temp[i]), 1, nchar(as.character(muta
  max_temp_double <- as.double(substr(as.character(mutating$Maximum_Temp[i]), 1, nchar(as.character(muta
  total_rain_double <- as.double(substr(as.character(mutating$Total_Rain[i]), 1, nchar(as.character(muta
  visibility_double <- as.double(substr(as.character(mutating$Visibility[i]), 1, nchar(as.character(muta
  max_wind_speed_double <- as.double(substr(as.character(mutating$Max_Wind_Speed[i]), 1, nchar(as.chara
  max_wind_gust_double <- as.double(substr(as.character(mutating$Max_Wind_Gust[i]), 1, nchar(as.charact

  if(max_wind_gust_double == 999)
  {
    max_wind_gust_double <- NA
  }

  weather_data_doubles <- add_row(weather_data_doubles, Year = mutating$Year[i], Month = mutating$Month
}
```

Plotting Variables

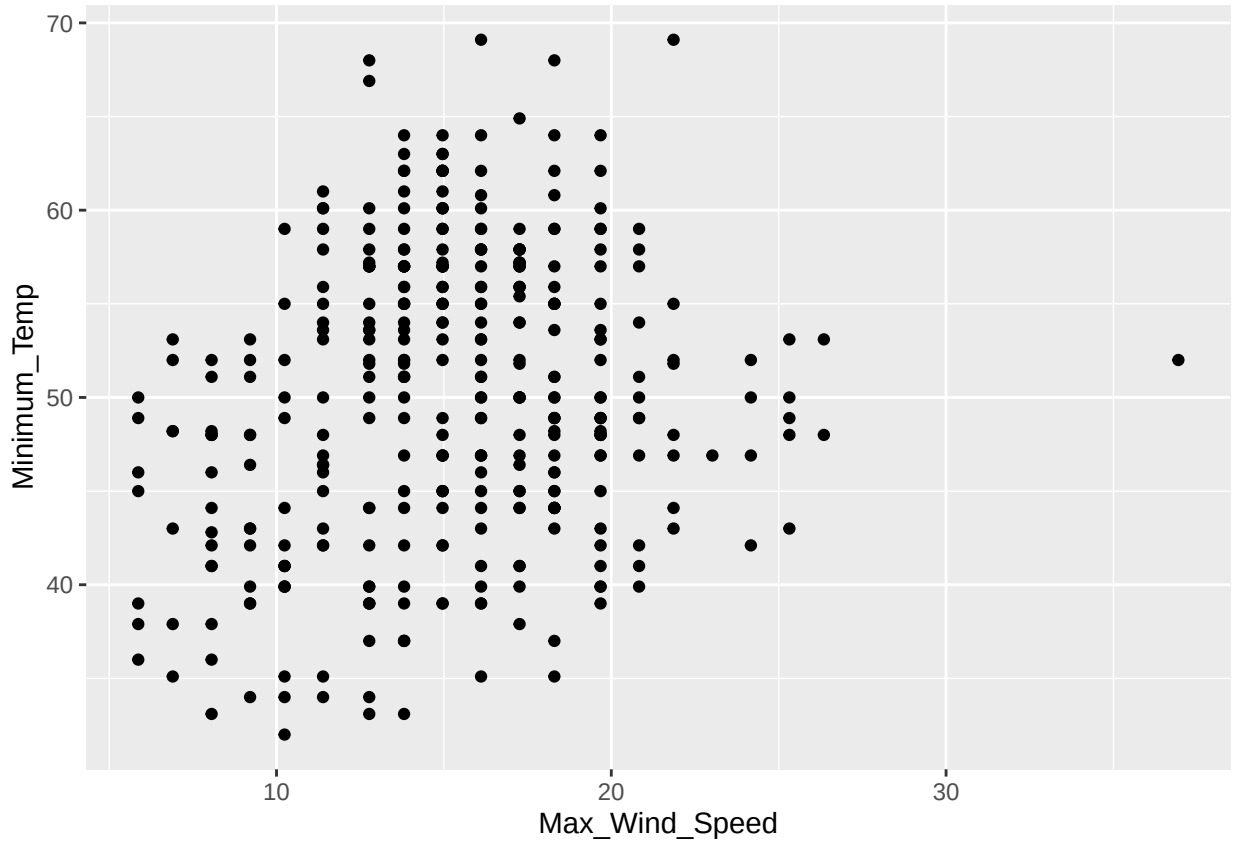
The plots below were used to try and determine the best variables for our models generated.
This first plot looks at the total rain vs the visibility for each day. This helped me realize these variables would be too clumped together for a decent model.

```
ggplot(data = weather_data_doubles, aes(x = Total_Rain, y = Visibility)) +
  geom_point()
```



This second plot looks at the Max wind speed vs the minimum temp for each day.
This plot looked much better in terms of spread of variables. Thus these are the variables I decided to use for my Model 2.

```
ggplot(data = weather_data_doubles, aes(x = Max_Wind_Speed, y = Minimum_Temp)) +
  geom_point()
```



Model

This is the initial model showing Visibility agains the Total_Rain and Minimum_Temp The reason I chose total Rain and Minimum temp for my model is that they both would have the best relation with the visibility factor.
Unfortunately this model did not work out as well as I wanted due to visibility having to many values that are the same. This caused my predictions to appear far from their actual values.

```
model <- lm(Visibility ~ Total_Rain + Minimum_Temp, data = weather_data_doubles)
summary(model)
```

```
##
## Call:
## lm(formula = Visibility ~ Total_Rain + Minimum_Temp, data = weather_data_doubles)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -5.1221 -0.1308  0.4656  0.5343  1.1069
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.773345   0.344748  28.349   <2e-16 ***
## Total_Rain   -0.800322   0.385594  -2.076   0.0386 *
## Minimum_Temp -0.006127   0.006767  -0.905   0.3658
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9917 on 362 degrees of freedom
## Multiple R-squared:  0.01284,    Adjusted R-squared:  0.007387
## F-statistic: 2.354 on 2 and 362 DF,  p-value: 0.0964
```

Data split for model testing and training

I split the data using the average split of 80% for training and 20% for testing.
Once the Data was split the model was recreated using the training data.
This new model allowed for predictions to be made using the testing dataset.
I then added my predictions back to my dataset.

```
#split data
train_rows <- as.vector(createDataPartition(weather_data_doubles$Visibility, p = 0.8, list = FALSE))

#get test rows and rest rows
rest <- weather_data_doubles[train_rows, ]
test <- weather_data_doubles[-train_rows, ]

#split data again to get validation rows/test rows/train rows (20/20/60)
train_rows <- as.vector(createDataPartition(rest$Visibility, p = 0.75, list = FALSE))

#split rest rows to validate and train
validate <- rest[-train_rows, ]
train <- rest[train_rows, ]

#recreate model with training rows
model <- lm(Visibility ~ Total_Rain + Minimum_Temp , data = train)
```

Add in Predictions

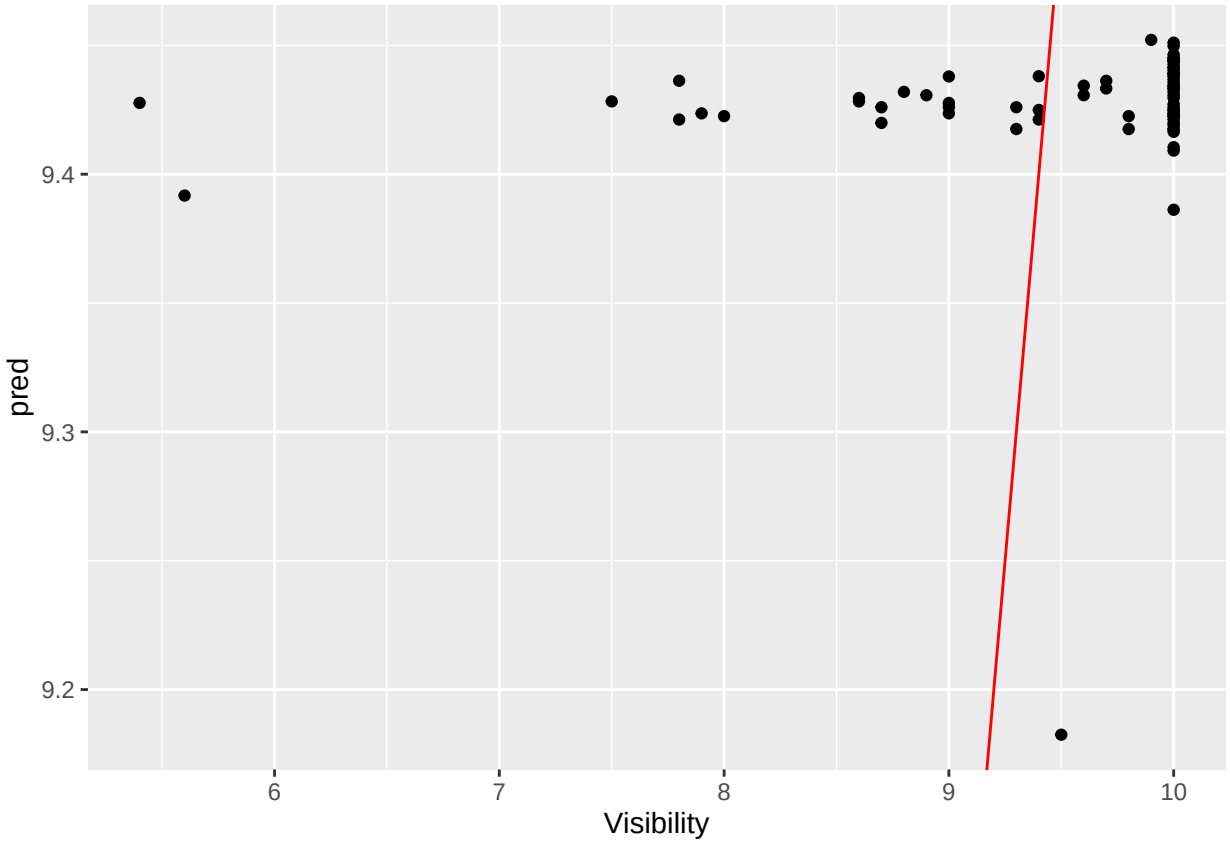
This adds the predictions to the model using the validation set

```
predictions <- add_predictions(validate, model)
```

Predictions Graph

Below is a graph showing my predictions generated using the testing set with my model.

```
ggplot(data = predictions, mapping = aes(x = Visibility, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red")
```



Goodness of fit measures

These are the goodness of fit measures

```
r2 <- R2(predictions$pred, predictions$Visibility)
mae <- MAE(predictions$pred, predictions$Visibility)
rmse <- RMSE(predictions$pred, predictions$Visibility)

print(r2)
```

```
## [1] 0.0118205
```

```
print(mae)
```

```
## [1] 0.6742821
```

```
print(rmse)
```

```
## [1] 0.9296593
```

Add in Residuals

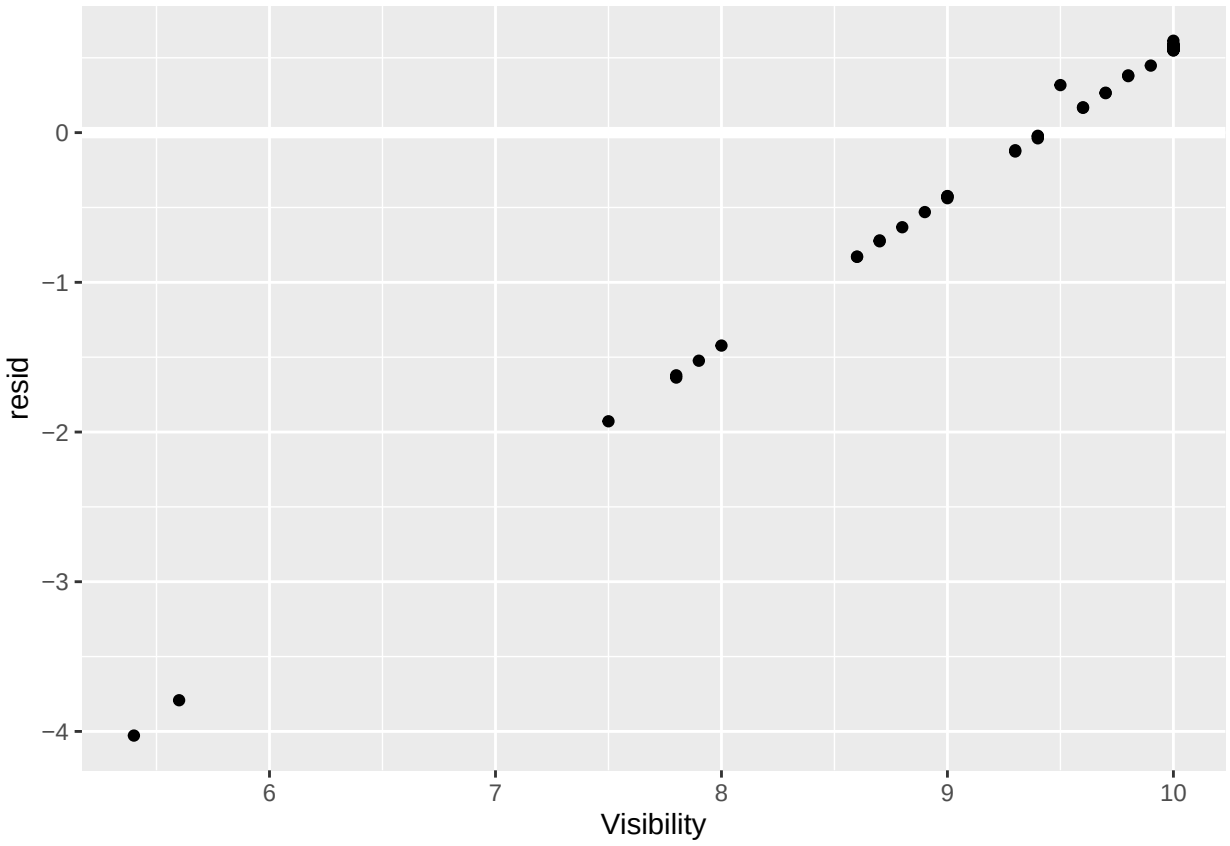
This adds in the residuals to the model using the validation set

```
resids <- add_residuals(validate, model)
```

Residuals Graph

Below is a graph of the residuals where anything below 0 means its being under predicted and anything above 0 means its being over predicted.
As you can see in this graph for my initial model most of my values were being under predicted and some of them were very far off.

```
ggplot(data = resids, mapping = aes(x = Visibility, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point()
```



Model2

This model is used to predict the Minimum Temperature. The way I came to picking my variables is simply I looked for variables that haver values in a decent range being that they dont have a bunch of values with the same numbers. Thus I also looked for what factors I believe would contribute to the temperature like the wind speed and the rain.
This model so far looks much better than the first model with the Pr(>|t|) tab showing multiple * values for no only the intercept but also my two variables. This tells me that there is a high level of confidence with 3* values being the best.

```
model2 <- lm(Minimum_Temp ~ Max_Wind_Speed + Total_Rain, data = weather_data_doubles)
summary(model2)
```

```
##
## Call:
## lm(formula = Minimum_Temp ~ Max_Wind_Speed + Total_Rain, data = weather_data_doubles)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.7498  -5.5390   0.3134   5.9379  18.5415
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    44.65443    1.44430  30.918  < 2e-16 ***
## Max_Wind_Speed  0.37620    0.09183   4.097 5.18e-05 ***
## Total_Rain     -8.81718    2.91577  -3.024  0.00267 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.529 on 362 degrees of freedom
## Multiple R-squared:  0.06145,    Adjusted R-squared:  0.05626
## F-statistic: 11.85 on 2 and 362 DF,  p-value: 1.035e-05
```

Data split for model testing and training

I split the data using the average split of 80% for training and 20% for testing.
Once the Data was split the model was recreated using the training data.
This new model allowed for predictions to be made using the testing dataset.
I then added my predictions back to my dataset.

```
#split data
train_rows <- as.vector(createDataPartition(weather_data_doubles$Minimum_Temp, p = 0.8, list = FALSE))

#get test rows and rest rows
rest <- weather_data_doubles[train_rows, ]
test <- weather_data_doubles[-train_rows, ]

#split data again to get validation rows/test rows/train rows (20/20/60)
train_rows <- as.vector(createDataPartition(rest$Minimum_Temp, p = 0.75, list = FALSE))

#split rest rows to validate and train
```

```
validate <- rest[-train_rows, ]
train <- rest[train_rows, ]

#recreate model with training rows
model2 <- lm(Minimum_Temp ~ Max_Wind_Speed + Total_Rain , data = train)
```

Add in Predictions

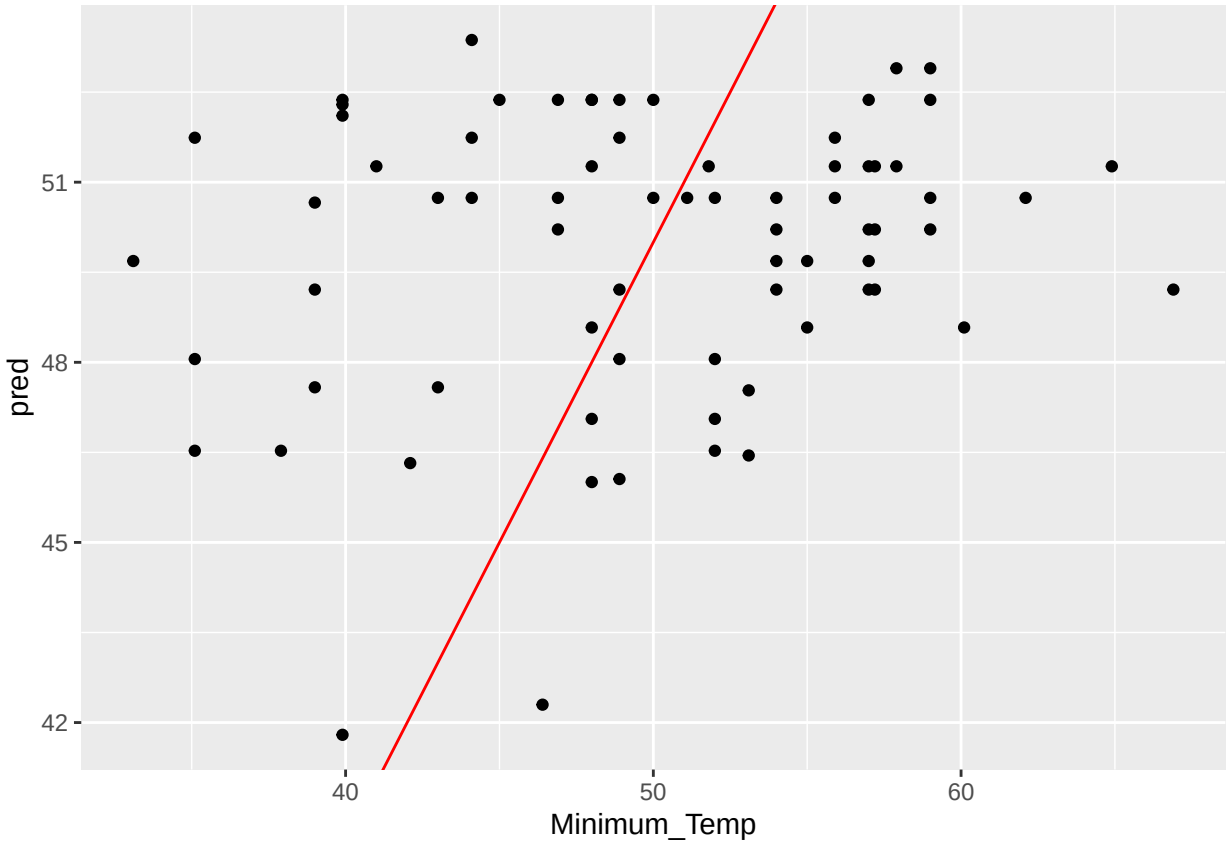
This adds the predictions to the model using the validation set

```
predictions <- add_predictions(validate, model2)
```

Predictions Graph

Below is a graph showing my predictions generated using the testing set with my model.
This model also isnt that great of a model but it seems to be doing better being that my values are more spread out across the graph.

```
ggplot(data = predictions, mapping = aes(x = Minimum_Temp, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red")
```



Goodness of fit measures

These are the goodness of fit measures

```
r2 <- R2(predictions$pred, predictions$Minimum_Temp)
mae <- MAE(predictions$pred, predictions$Minimum_Temp)
rmse <- RMSE(predictions$pred, predictions$Minimum_Temp)

print(r2)
```

```
## [1] 0.03689106
```

```
print(mae)
```

```
## [1] 6.366966
```

```
print(rmse)
```

```
## [1] 7.524261
```

Add in Residuals

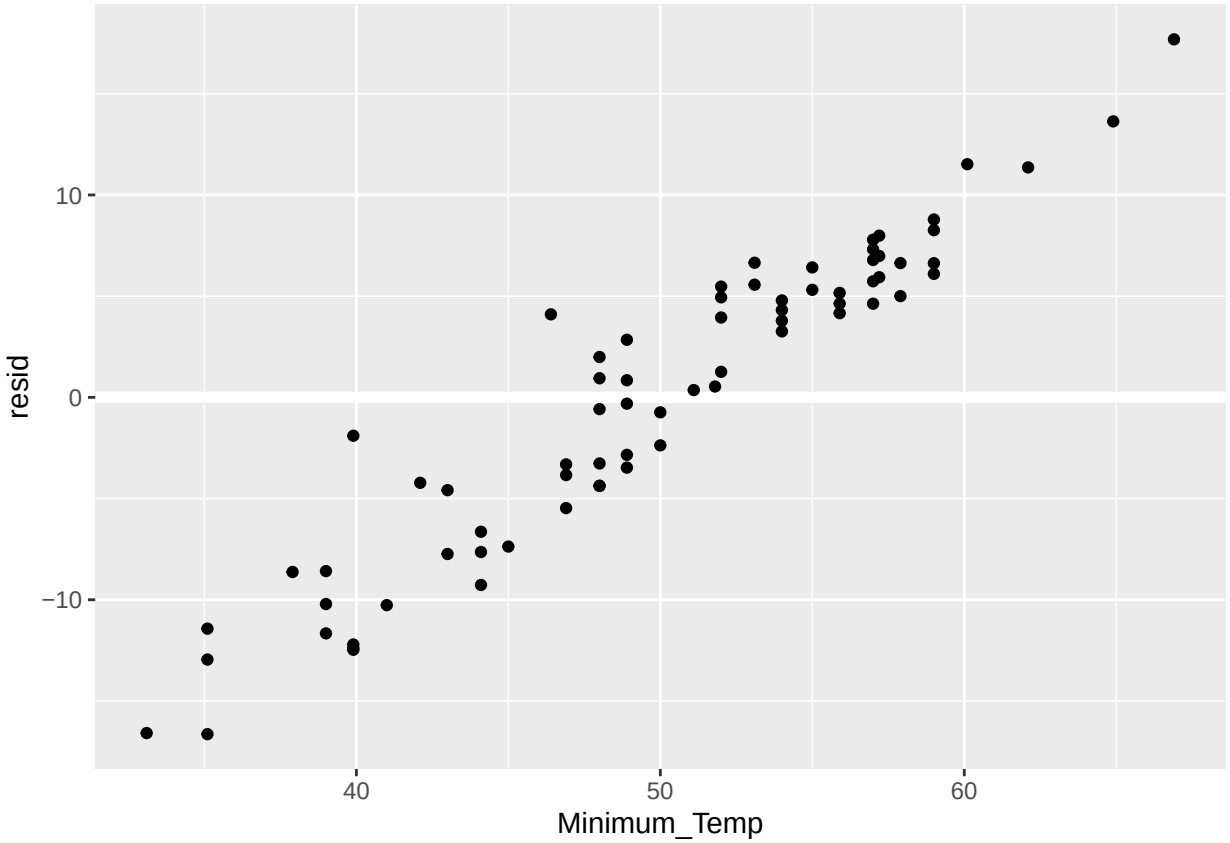
This adds in the residuals to the model using the validation set

```
resids <- add_residuals(validate, model2)
```

Residuals Graph

Below is a graph of the residuals where anything below 0 means its being under predicted and anything above 0 means its being over predicted.
As you can see in this graph for my current model most of my values are spread out event across the graph. Being that some of these are hitting a value of +- 10 on the y axis tells me there is something not completely correct but for the most part this model seems to be a decent one.

```
ggplot(data = resids, mapping = aes(x = Minimum_Temp, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point()
```


10
0
−10
resid
40
50
60
Minimum_Temp