CSGY – 6923

Machine Learning

Homework 2

Hasan Zubairi Masud: hzm212 : N15065030

Contents:

1. Introduction

This is an Exploratory Data Analysis (EDA) of the Motor Vehicle Collisions – Crashes data set provided by the city of New York. To begin let us first describe exactly what an EDA is to best explain what I wish to accomplish with this exercise.

An EDA is essentially a prescreening of a data set. This is where we analyze a data set to see if there are any anomalies, empty entries, underlying assumptions or structures, if there are any useless variables, understand links in the data, and more to gain the most possible insight from the data set before performing further and deeper analysis on it.

Why is the EDA necessary? We are specifically performing the EDA before performing Machine Learning techniques on our data set if we have many anomalies, outliers, empty values and more that can affect our models and create classification trends that may not be accurate. This way we make sure our data set is as useful as possible and maximizes our eventual output.

2. The Data

The data I have chosen to analyse is the Motor Vehicle Collisions – Crashes data set provided by the city of New York on the NYC Open Data platform. I wanted to choose a data set to analyse so that I could possibly discover trends to aid the general public in some way. By analysing this data set there are a few trends that may come out that may have positive consequences. For example, if I discover there is a specific area which happens to have a large number of accidents involving speeding, it could be worth exploring lowering the speed limit or creating more signs and awareness in the area of the speed limit. If there is a specific make of car that is getting into accidents more than others it may be worth exploring whether there is a feature of that care that is leading to accidents (of course this would have to be cross-referenced with data on the most common cars to come to an accurate conclusion, however). If there is an area where there are crashes constantly, maybe there is a defect in the way the road is built there. These are just a few examples of possible trends we may be able to discover.

Further background on the data set is that the data tables contain information from all police-reported motor vehicle collisions in NYC. The police report is required to be filled out for collisions where someone is injured or killed, or where there is at least $1000 worth of damage. This essentially means minor scrapes that do not total a lot of damage or have no one injured is not included. It has data on crashes from 2012 to the present with a current total of 1.9 million entries. There are 29 labels as well that are listed below.

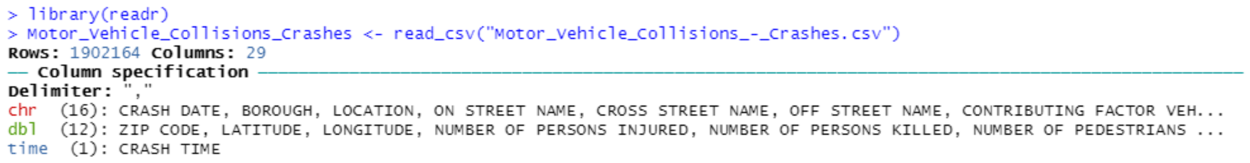
Label Name	Description	Variable type
CRASH DATE	Occurrence date of collision	Date & Time
CRASH TIME	Occurrence time of collision	Plain Text
BOROUGH	Borough where collision occurred	Plain Text
ZIP CODE	Postal code of incident occurrence	Plain Text
LATITUDE	Latitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)	Number
LONGITUDE	Longitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)	Number
LOCATION	Latitude , Longitude pair	Location
ON STREET NAME	Street on which the collision occurred	Plain Text
CROSS STREET NAME	Nearest cross street to the collision	Plain Text
OFF STREET NAME	Street address if known	Plain Text
NUMBER OF PERSONS INJURED	Number of persons injured	Number
NUMBER OF PERSONS KILLED	Number of persons killed	Number
NUMBER OF PEDESTRIANS	Number of pedestrians injured	Number

INJURED		
NUMBER OF PEDESTRIANS KILLED	Number of pedestrians killed	Number
NUMBER OF CYCLIST INJURED	Number of cyclists injured	Number
NUMBER OF CYCLIST KILLED	Number of cyclists killed	Number
NUMBER OF MOTORIST INJURED	Number of vehicle occupants injured	Number
NUMBER OF MOTORIST KILLED	Number of vehicle occupants killed	Number
CONTRIBUTING FACTOR VEHICLE 1	Factors contributing to the collision for designated vehicle	Plain Text
CONTRIBUTING FACTOR VEHICLE 2	Factors contributing to the collision for designated vehicle	Plain Text
CONTRIBUTING FACTOR VEHICLE 3	Factors contributing to the collision for designated vehicle	Plain Text
CONTRIBUTING FACTOR VEHICLE 4	Factors contributing to the collision for designated vehicle	Plain Text
CONTRIBUTING FACTOR VEHICLE 5	Factors contributing to the collision for designated vehicle	Plain Text
COLLISION_ID	Unique record code generated by system. Primary Key for Crash table.	Number
VEHICLE TYPE CODE 1	Type of vehicle based on the selected vehicle category (ATV, bicycle, car/suv, ebike, escooter, truck/bus, motorcycle, other)	Plain Text
VEHICLE TYPE CODE 2	Type of vehicle based on the selected vehicle category (ATV, bicycle, car/suv, ebike, escooter, truck/bus, motorcycle, other)	Plain Text

VEHICLE TYPE CODE 3	Type of vehicle based on the selected vehicle category (ATV, bicycle, car/suv, ebike, escooter, truck/bus, motorcycle, other)	Plain Text
VEHICLE TYPE CODE 4	Type of vehicle based on the selected vehicle category (ATV, bicycle, car/suv, ebike, escooter, truck/bus, motorcycle, other)	Plain Text
VEHICLE TYPE CODE 5	Type of vehicle based on the selected vehicle category (ATV, bicycle, car/suv, ebike, escooter, truck/bus, motorcycle, other)	Plain Text

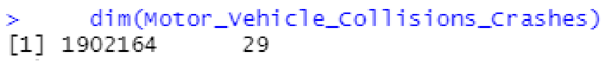
3. Loading the Data

The first step is to load the data. Here I load my data set into the variable Motor_Vehicle_Collisions_Crashes.



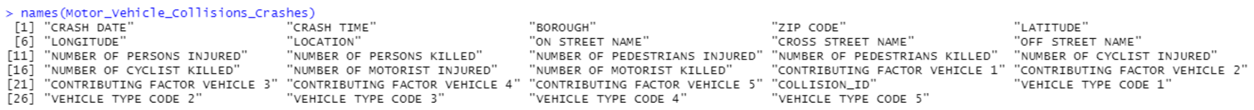
```
> library(readr)
> Motor_Vehicle_Collisions_Crashes <- read_csv("Motor_Vehicle_Collisions_-_Crashes.csv")
Rows: 1902164 Columns: 29
── Column specification ──────────────────────────────────────────────────────────────
Delimiter: ","
chr  (16): CRASH DATE, BOROUGH, LOCATION, ON STREET NAME, CROSS STREET NAME, OFF STREET NAME, CONTRIBUTING FACTOR VEH...
dbl  (12): ZIP CODE, LATITUDE, LONGITUDE, NUMBER OF PERSONS INJURED, NUMBER OF PERSONS KILLED, NUMBER OF PEDESTRIANS ...
time  (1): CRASH TIME
```

Next, we need to ensure that data was loaded correctly. To do this I first check the dimensions of my data.



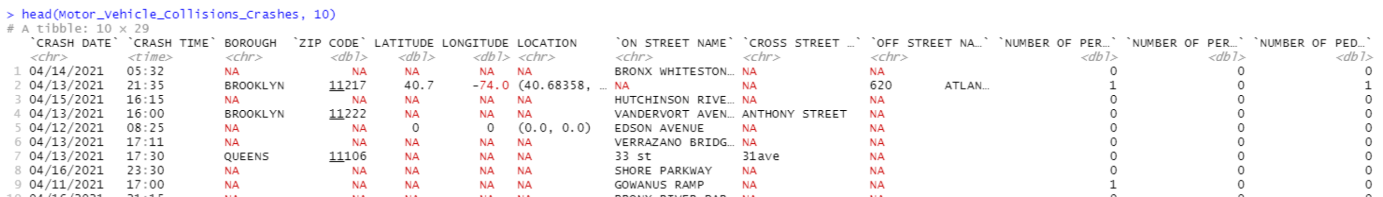
```
>     dim(Motor_Vehicle_Collisions_Crashes)
[1] 1902164      29
```

We see that the dimensions are correct, all 29 variables are present and there are 1.9 million entries as specified by the documentation on the dataset. We also need to see that all specified variables are there.



```
> names(Motor_Vehicle_Collisions_Crashes)
 [1] "CRASH DATE"                    "CRASH TIME"                    "BOROUGH"                       "ZIP CODE"                      "LATITUDE"
 [6] "LONGITUDE"                     "LOCATION"                      "ON STREET NAME"                "CROSS STREET NAME"             "OFF STREET NAME"
[11] "NUMBER OF PERSONS INJURED"     "NUMBER OF PERSONS KILLED"      "NUMBER OF PEDESTRIANS INJURED" "NUMBER OF PEDESTRIANS KILLED"  "NUMBER OF CYCLIST INJURED"
[16] "NUMBER OF CYCLIST KILLED"      "NUMBER OF MOTORIST INJURED"    "NUMBER OF MOTORIST KILLED"     "CONTRIBUTING FACTOR VEHICLE 1" "CONTRIBUTING FACTOR VEHICLE 2"
[21] "CONTRIBUTING FACTOR VEHICLE 3" "CONTRIBUTING FACTOR VEHICLE 4" "CONTRIBUTING FACTOR VEHICLE 5" "COLLISION_ID"                  "VEHICLE TYPE CODE 1"
[26] "VEHICLE TYPE CODE 2"           "VEHICLE TYPE CODE 3"           "VEHICLE TYPE CODE 4"           "VEHICLE TYPE CODE 5"
```

The 29 variables are all correctly named which is obviously important. As a final sanity check, we will look at a small subset of the data to ensure everything looks fine.

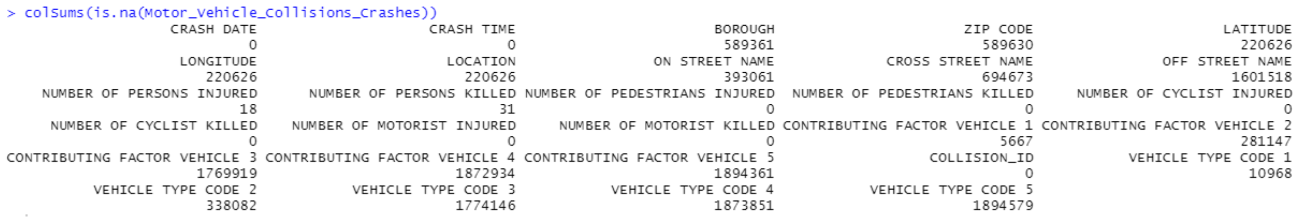


```
> head(Motor_Vehicle_Collisions_Crashes, 10)
# A tibble: 10 × 29
  `CRASH DATE` `CRASH TIME` BOROUGH  `ZIP CODE` LATITUDE LONGITUDE LOCATION      `ON STREET NAME` `CROSS STREET …` `OFF STREET NA…` `NUMBER OF PER…` `NUMBER OF PER…` `NUMBER OF PED…`
  <chr>        <time>       <chr>         <dbl>    <dbl>     <dbl> <chr>         <chr>            <chr>            <chr>                       <dbl>            <dbl>            <dbl>
1 04/14/2021   05:32        NA               NA       NA        NA NA            BRONX WHITESTON… NA               NA                              0                0                0
2 04/13/2021   21:35        BROOKLYN      11217     40.7     -74.0 (40.68358, … NA               NA               620       ATLAN…                1                0                1
3 04/15/2021   16:15        NA               NA       NA        NA NA            HUTCHINSON RIVE… NA               NA                              0                0                0
4 04/13/2021   16:00        BROOKLYN      11222       NA        NA NA            VANDERVORT AVEN… ANTHONY STREET   NA                              0                0                0
5 04/12/2021   08:25        NA               NA        0         0 (0.0, 0.0)    EDSON AVENUE     NA               NA                              0                0                0
6 04/13/2021   17:11        NA               NA       NA        NA NA            VERRAZANO BRIDG… NA               NA                              0                0                0
7 04/13/2021   17:30        QUEENS        11106       NA        NA NA            33 st            31ave            NA                              0                0                0
8 04/16/2021   23:30        NA               NA       NA        NA NA            SHORE PARKWAY    NA               NA                              0                0                0
9 04/11/2021   17:00        NA               NA       NA        NA NA            GOWANUS RAMP     NA               NA                              1                0                0
```

Everything looks correct so we can conclude the data has been adequately loaded into the project.

4. Checking for n/a values

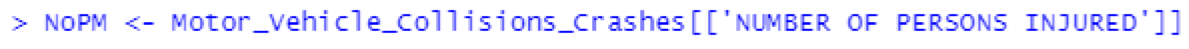
If there is one thing I have noticed about my data set is that there are a lot of n/a values in certain columns. This seems this is because Police Reports do not necessarily need to report all these 29 aspects 100% of the time. As such I want to do an analysis to see how many values in each of my 29 labels are n/a.



```
> colSums(is.na(Motor_Vehicle_Collisions_Crashes))
                   CRASH DATE                    CRASH TIME                       BOROUGH                      ZIP CODE                      LATITUDE
                            0                             0                        589361                        589630                        220626
                    LONGITUDE                      LOCATION                ON STREET NAME             CROSS STREET NAME               OFF STREET NAME
                       220626                        220626                        393061                        694673                       1601518
      NUMBER OF PERSONS INJURED      NUMBER OF PERSONS KILLED NUMBER OF PEDESTRIANS INJURED  NUMBER OF PEDESTRIANS KILLED    NUMBER OF CYCLIST INJURED
                           18                            31                             0                             0                             0
       NUMBER OF CYCLIST KILLED     NUMBER OF MOTORIST INJURED      NUMBER OF MOTORIST KILLED CONTRIBUTING FACTOR VEHICLE 1 CONTRIBUTING FACTOR VEHICLE 2
                            0                             0                             0                          5667                        281147
CONTRIBUTING FACTOR VEHICLE 3 CONTRIBUTING FACTOR VEHICLE 4 CONTRIBUTING FACTOR VEHICLE 5                  COLLISION_ID           VEHICLE TYPE CODE 1
                      1769919                       1872934                       1894361                             0                         10968
          VEHICLE TYPE CODE 2           VEHICLE TYPE CODE 3           VEHICLE TYPE CODE 4           VEHICLE TYPE CODE 5
                       338082                       1774146                       1873851                       1894579
```

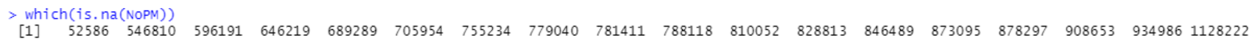
Instantly we see a few clear things. The first is the labels that have data for all entries. These are fields with a 0, indicating that there were no n/a values in the entire data set for these labels. The labels are: Crash Date, Crash Time, Number of Pedestrians Injured, Number of Pedestrians Killed, Number of Cyclists Injured, Number of Cyclists Killed, Number of Motorists Injured, Number of Motorists Killed and Collision ID. Nine labels in total are those you would most likely expect to have no n/a values. Crash Date and Crash Time are always applicable. There are 6 labels here about categories of people who have either been injured or killed, would rarely be n/a over simply putting 0, it is pretty clear when someone is injured or killed and police priority to document the involved parties. The last is the Collision ID which must be generated for each documented collision so once again not surprising to see no n/a values.

Second, there are the labels with such a small number of n/a values that it is negligible when compared to the fact the data set has 1,902,164 total entries. These values are the number of Persons Injured (18) and the Number of persons killed (31). I am lost for meaning on how exactly these labels could be n/a values, as it would seem pretty clear cut for any police reports. So let us look at some of these such entries. Let us analyse the Person's Injured category first. We begin by loading the column into a variable.



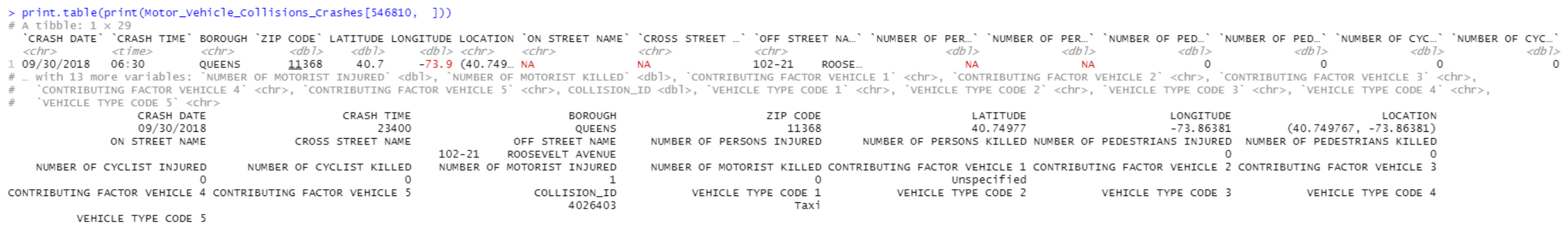
```
> NoPM <- Motor_Vehicle_Collisions_Crashes[['NUMBER OF PERSONS INJURED']]
```

Next, I check to find the specific entries that are n/a values.

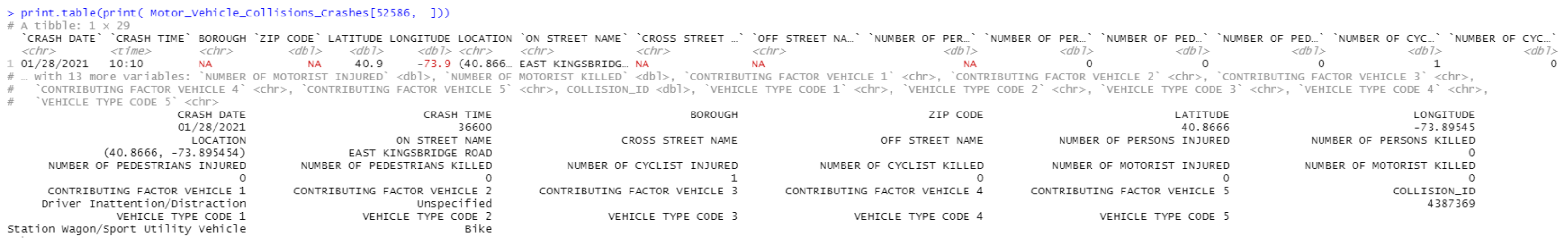


```
> which(is.na(NoPM))
 [1]   52586  546810  596191  646219  689289  705954  755234  779040  781411  788118  810052  828813  846489  873095  878297  908653  934986 1128222
```

Then I print out a few of these entries to see if there is anything I may be able to glean.



```
> print.table(print(Motor_Vehicle_Collisions_Crashes[546810,  ]))
# A tibble: 1 × 29
  `CRASH DATE` `CRASH TIME` BOROUGH `ZIP CODE` LATITUDE LONGITUDE LOCATION `ON STREET NAME` `CROSS STREET …` `OFF STREET NA…` `NUMBER OF PER…` `NUMBER OF PER…` `NUMBER OF PED…` `NUMBER OF PED…` `NUMBER OF CYC…` `NUMBER OF CYC…`
  <chr>        <time>       <chr>        <dbl>    <dbl>     <dbl> <chr>    <chr>            <chr>            <chr>                       <dbl>            <dbl>            <dbl>            <dbl>            <dbl>            <dbl>
1 09/30/2018   06:30        QUEENS       11368     40.7     -73.9 (40.749… NA               NA               102-21     ROOSE…             NA               NA                0                0                0                0
# … with 13 more variables: `NUMBER OF MOTORIST INJURED` <dbl>, `NUMBER OF MOTORIST KILLED` <dbl>, `CONTRIBUTING FACTOR VEHICLE 1` <chr>, `CONTRIBUTING FACTOR VEHICLE 2` <chr>, `CONTRIBUTING FACTOR VEHICLE 3` <chr>,
#   `CONTRIBUTING FACTOR VEHICLE 4` <chr>, `CONTRIBUTING FACTOR VEHICLE 5` <chr>, COLLISION_ID <dbl>, `VEHICLE TYPE CODE 1` <chr>, `VEHICLE TYPE CODE 2` <chr>, `VEHICLE TYPE CODE 3` <chr>, `VEHICLE TYPE CODE 4` <chr>,
#   `VEHICLE TYPE CODE 5` <chr>
                   CRASH DATE                    CRASH TIME                       BOROUGH                      ZIP CODE                      LATITUDE                     LONGITUDE                      LOCATION 
                   09/30/2018                         23400                        QUEENS                         11368                      40.74977                     -73.86381          (40.749767, -73.86381) 
               ON STREET NAME             CROSS STREET NAME               OFF STREET NAME     NUMBER OF PERSONS INJURED      NUMBER OF PERSONS KILLED NUMBER OF PEDESTRIANS INJURED  NUMBER OF PEDESTRIANS KILLED 
                                                          102-21    ROOSEVELT AVENUE                                                                                          0                             0 
    NUMBER OF CYCLIST INJURED      NUMBER OF CYCLIST KILLED    NUMBER OF MOTORIST INJURED     NUMBER OF MOTORIST KILLED CONTRIBUTING FACTOR VEHICLE 1 CONTRIBUTING FACTOR VEHICLE 2 CONTRIBUTING FACTOR VEHICLE 3 
                            0                             0                             1                             0                   Unspecified                                                             
CONTRIBUTING FACTOR VEHICLE 4 CONTRIBUTING FACTOR VEHICLE 5                  COLLISION_ID           VEHICLE TYPE CODE 1           VEHICLE TYPE CODE 2           VEHICLE TYPE CODE 3           VEHICLE TYPE CODE 4 
                                                                              4026403                          Taxi                                                                                           

          VEHICLE TYPE CODE 5 
```



```
> print.table(print( Motor_Vehicle_Collisions_Crashes[52586,  ]))
# A tibble: 1 × 29
  `CRASH DATE` `CRASH TIME` BOROUGH `ZIP CODE` LATITUDE LONGITUDE LOCATION `ON STREET NAME` `CROSS STREET …` `OFF STREET NA…` `NUMBER OF PER…` `NUMBER OF PER…` `NUMBER OF PED…` `NUMBER OF PED…` `NUMBER OF CYC…` `NUMBER OF CYC…`
  <chr>        <time>       <chr>        <dbl>    <dbl>     <dbl> <chr>    <chr>            <chr>            <chr>                       <dbl>            <dbl>            <dbl>            <dbl>            <dbl>            <dbl>
1 01/28/2021   10:10        NA              NA     40.9     -73.9 (40.866… EAST KINGSBRIDG… NA               NA                             NA                0                0                0                1                0
# … with 13 more variables: `NUMBER OF MOTORIST INJURED` <dbl>, `NUMBER OF MOTORIST KILLED` <dbl>, `CONTRIBUTING FACTOR VEHICLE 1` <chr>, `CONTRIBUTING FACTOR VEHICLE 2` <chr>, `CONTRIBUTING FACTOR VEHICLE 3` <chr>,
#   `CONTRIBUTING FACTOR VEHICLE 4` <chr>, `CONTRIBUTING FACTOR VEHICLE 5` <chr>, COLLISION_ID <dbl>, `VEHICLE TYPE CODE 1` <chr>, `VEHICLE TYPE CODE 2` <chr>, `VEHICLE TYPE CODE 3` <chr>, `VEHICLE TYPE CODE 4` <chr>,
#   `VEHICLE TYPE CODE 5` <chr>
                        CRASH DATE                         CRASH TIME                            BOROUGH                           ZIP CODE                           LATITUDE                          LONGITUDE 
                        01/28/2021                              36600                                                                                                     40.8666                          -73.89545 
                          LOCATION                     ON STREET NAME                  CROSS STREET NAME                    OFF STREET NAME          NUMBER OF PERSONS INJURED           NUMBER OF PERSONS KILLED 
              (40.8666, -73.895454)               EAST KINGSBRIDGE ROAD                                                                                                                                         0 
      NUMBER OF PEDESTRIANS INJURED       NUMBER OF PEDESTRIANS KILLED           NUMBER OF CYCLIST INJURED           NUMBER OF CYCLIST KILLED          NUMBER OF MOTORIST INJURED          NUMBER OF MOTORIST KILLED 
                                  0                                  0                                  1                                  0                                  0                                  0 
      CONTRIBUTING FACTOR VEHICLE 1      CONTRIBUTING FACTOR VEHICLE 2      CONTRIBUTING FACTOR VEHICLE 3      CONTRIBUTING FACTOR VEHICLE 4      CONTRIBUTING FACTOR VEHICLE 5                       COLLISION_ID 
     Driver Inattention/Distraction                        Unspecified                                                                                                                            4387369 
                VEHICLE TYPE CODE 1                VEHICLE TYPE CODE 2                VEHICLE TYPE CODE 3                VEHICLE TYPE CODE 4                VEHICLE TYPE CODE 5 
Station Wagon/Sport Utility Vehicle                               Bike                                                                                                          
```

What is interesting about these is in both cases there is someone listed as injured in the categories of Motorist, Pedestrian or Cyclist, yet the number of people injured is n/a. I am unsure why this is but this happens such a negligible amount of times in my data set I feel comfortable removing these entries without sacrificing any of the dataset's integrity.

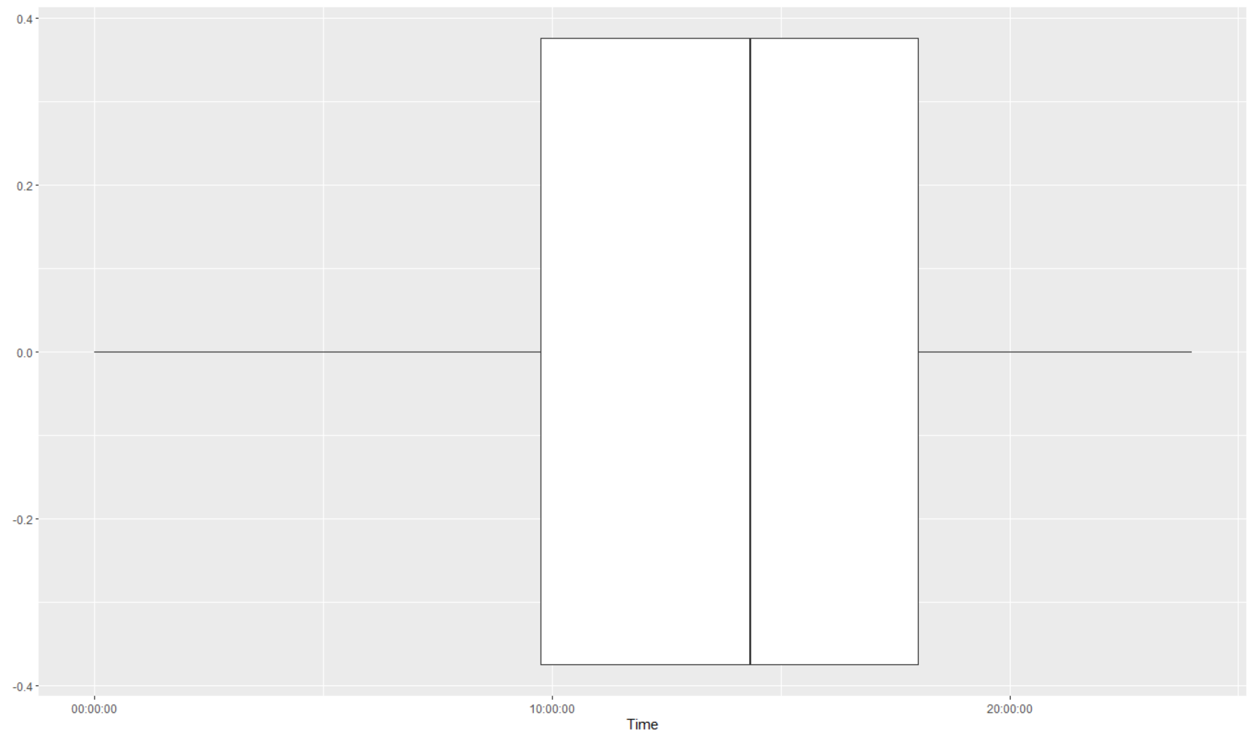
Third, there is "Contributing Factor vehicle 1" that has 5667 n/a values. This is much larger than 13 and 31, yet it is still a very low number when compared to the overall data set the size of ~1,9 million. It is also a very important feature as it gives the reasoning to the crash from which a lot of contextual information about what common causes there may be for crashes. As such, I also feel comfortable removing those 5667 entries when performing my eventual analysis.

Forth we look at the contributing factor values for vehicles 2 to 5 and vehicle type codes for 2 to 5 they scale up. This is because crashes tend to involve not too many cars, so you can see the n/a values get larger as we add more vehicles to crash. If there are 5 vehicles however we do not want to lose this information so we do not make any changes based on the high proportions of n/a values in this field.

Lastly, we have the labels that correspond to the location of the crash: Borough, Zip Code, Longitude, Latitude and Latitude. Borough and Zip Code appear to be somewhat correlated with Borough having 589361 entries and Zipcode having 589630, we need to check the correlation to be sure but both having such a close number of missing n/a values seems convincing, especially as Borough and Zip Code are intrinsically linked as Boroughs tend to have common Zip Codes. Longitude and Latitude are coordinates that are then put together to form the location. This seems to be why they all have the exact same number of n/a values being 220620. Now the interesting thing here is Borough and Zip Code could technically be derived from Longitude and Latitude. So if I desired I could update the data with the Borough and Zip Code when it is missing. Despite the location being important unless I am specifically doing analysis which relies on location, I see no reason to remove those entries, especially as it is a decent proportion of the total data set.

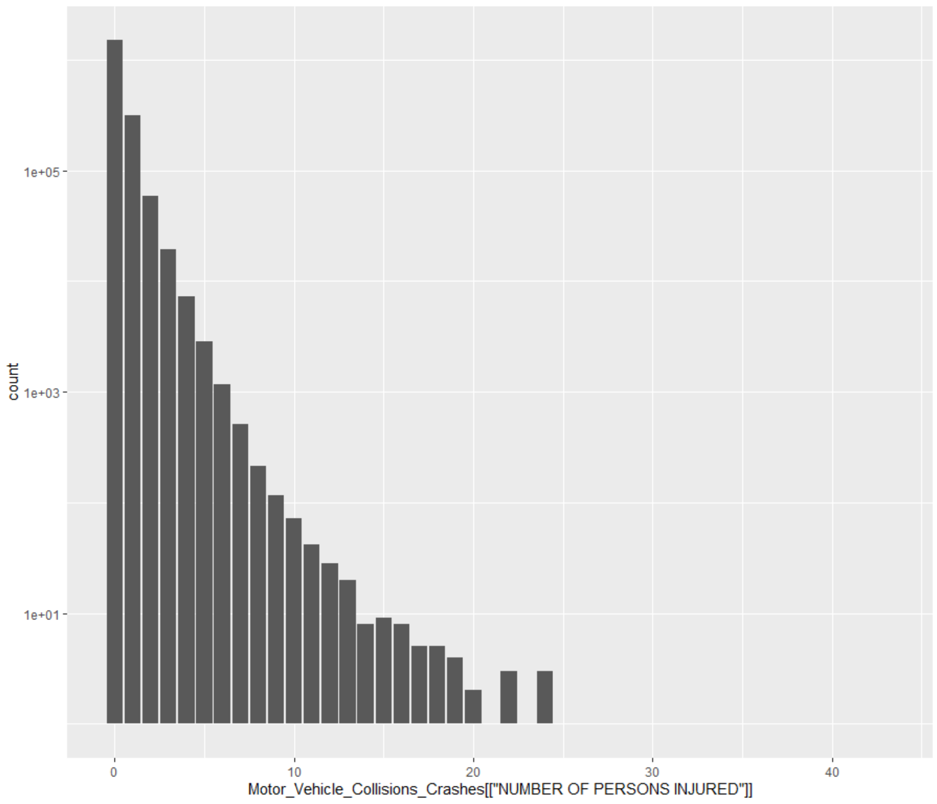
5. Checking Label distributions

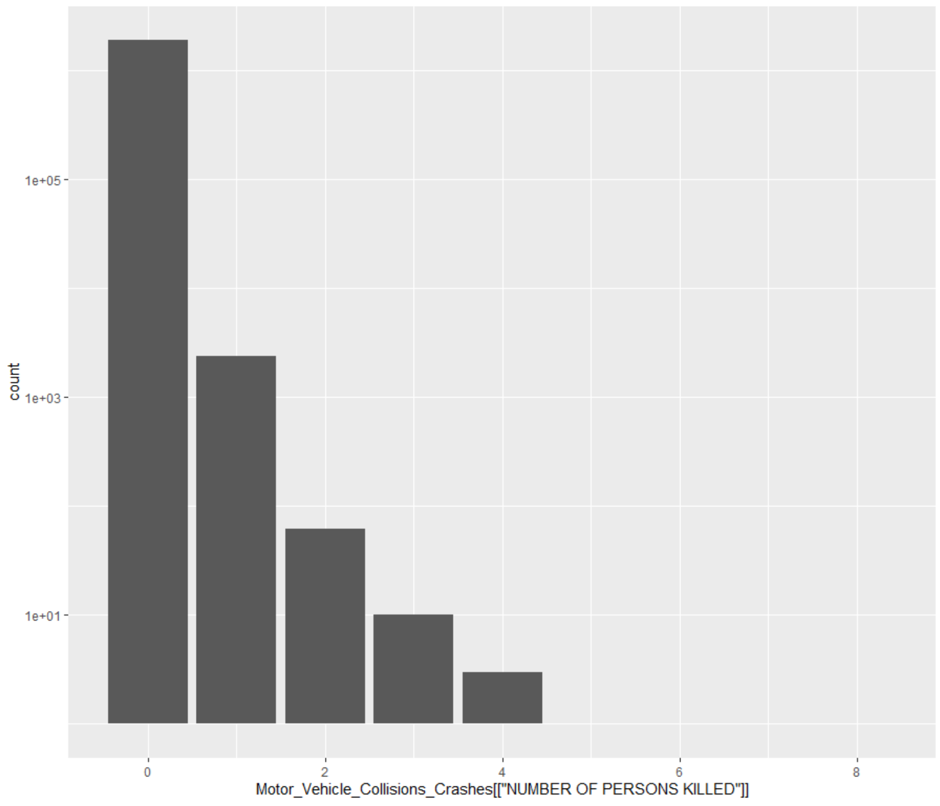
First let us check the distribution of times. There are ~1.9 million data points so plotting them all is not feasible, a box plot allows us to analyse the distribution of the data clearly.



The Boxplot shows a median average value of ~14:00 or 2 pm, and the vast majority in the times from ~10:00 to ~18:00. This is interesting as I thought it may correlate with the rush hour which tends to be earlier, however, this also makes sense as this is the range of time where the most people are awake and possibly as there is less traffic more potential for speeding and accidents. Regardless, it is good to understand how time is distributed so we can account for any bias we may see when doing analysis with time.

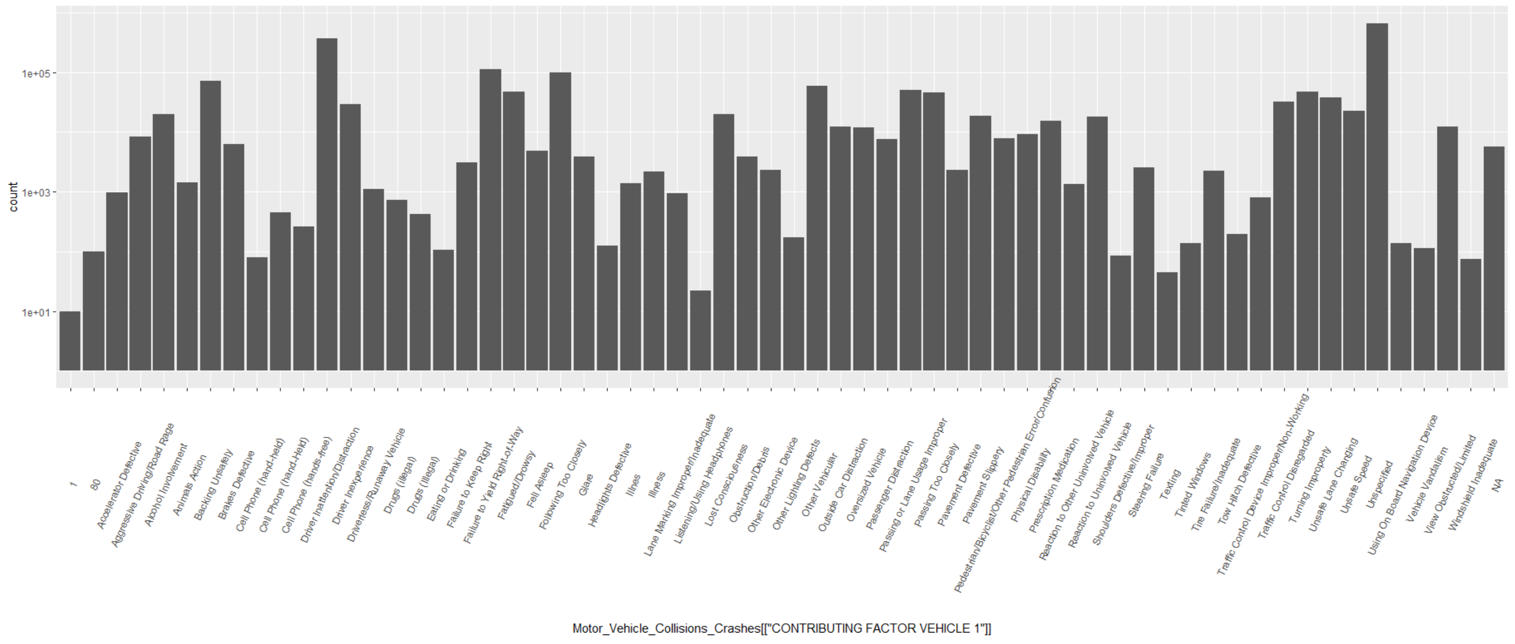
We will now analyse the distribution of the number of people Killed and Injured.

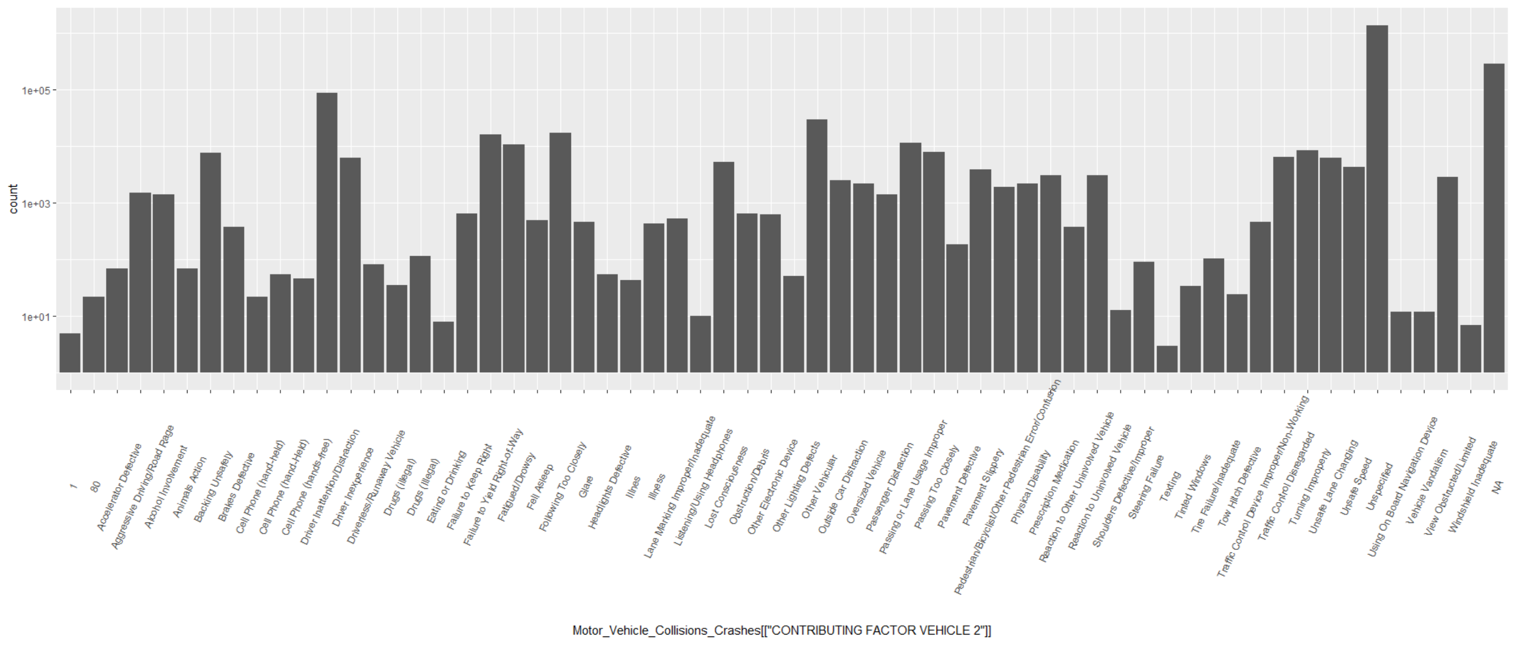


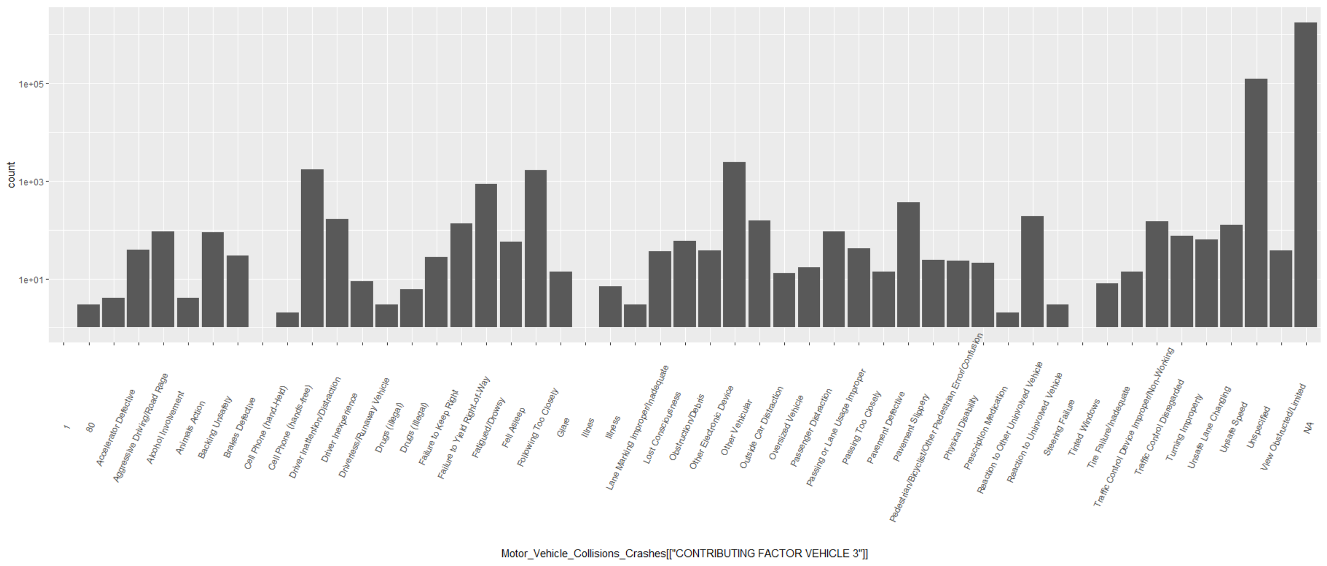


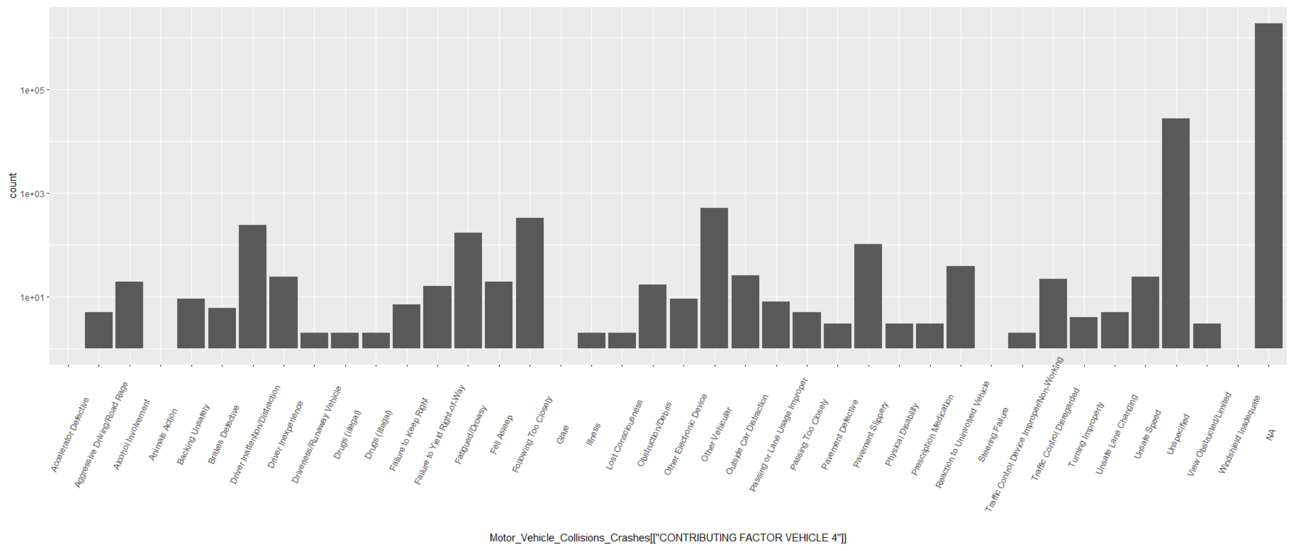
As one may expect we see that we have an exponential drop-off as the number of people injured or killed increases. It is more clear in the number of injured people as the range of the number of people injured is much greater. That is also an expected trend, people tend to be injured more than killed and that meets the expectation. What I find very surprising is how many people can be injured at once. There is a significant number of crashes where more than 20 people were injured which I did not expect as I personally have never heard of these situations and would be interested to see if there are common reasons that lead to these large numbers of injured drivers and civilians. Overall the distribution appears to be fine and no clear outliers seem present. As with the previous data analysed it is good to get an understanding of how it is distributed so that when performing future analysis I can understand the biases.

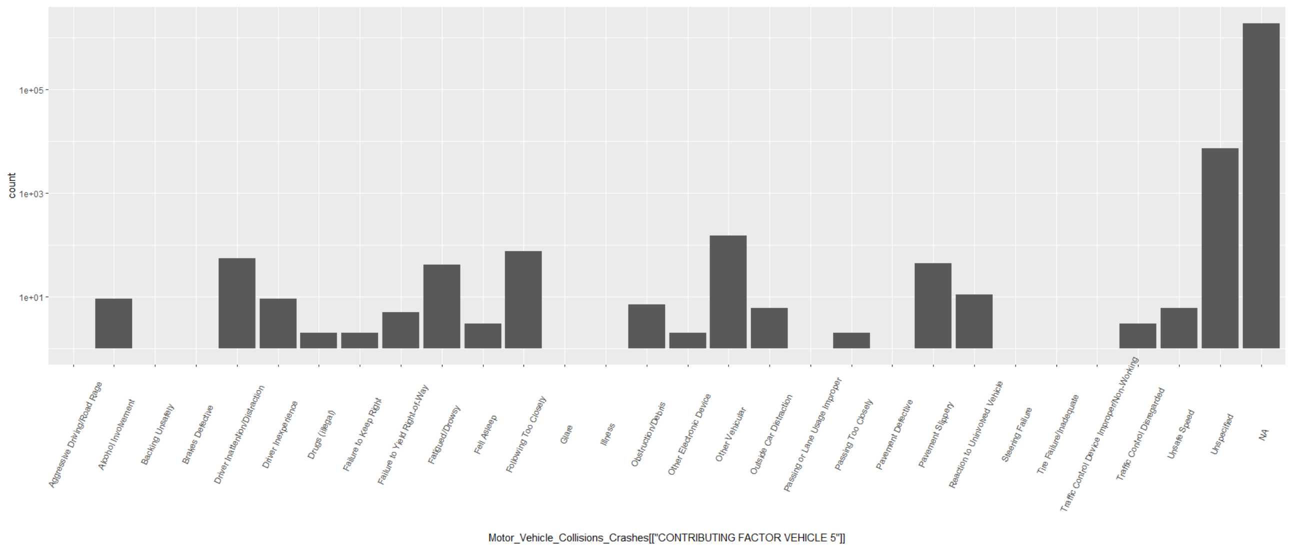
There is also an analysis of the reasons for each driver who got into the crash. The dataset includes reasons for up to 5 drivers and the causes that lead them to the crash.





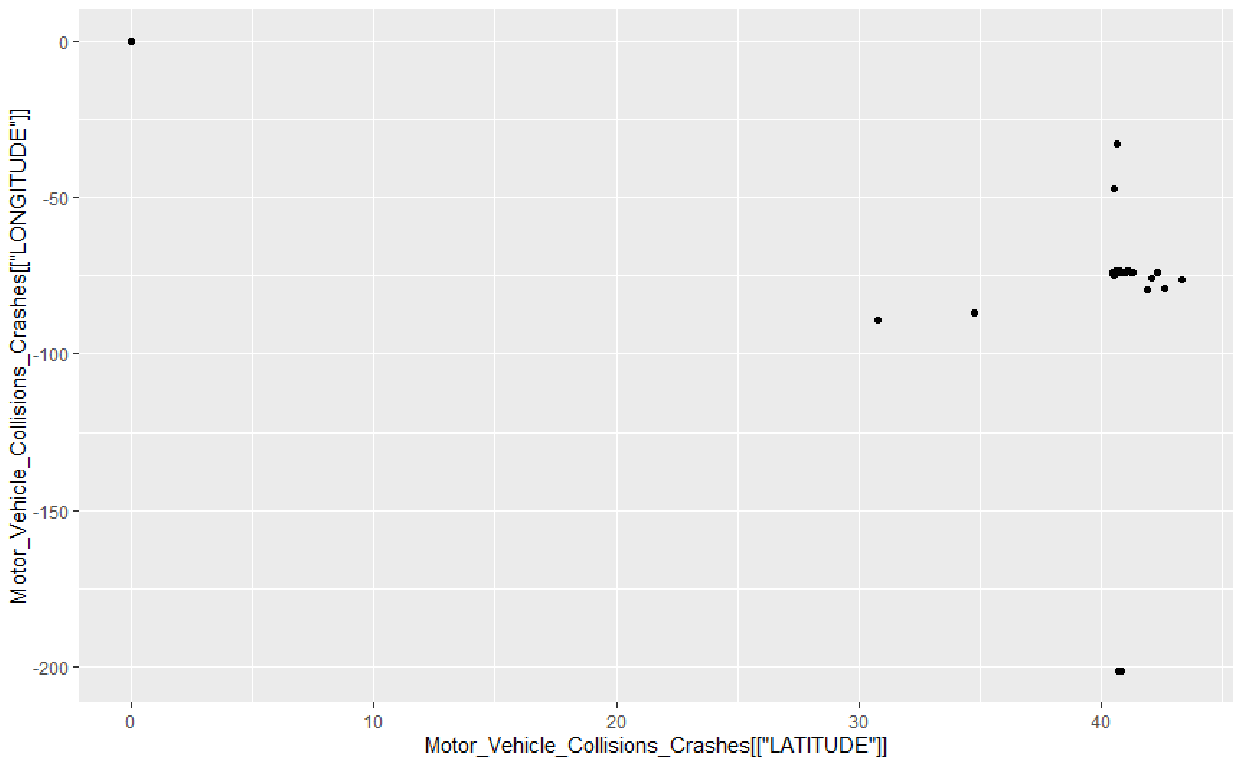

count
1e+05
1e+03
1e+01
1
80
Accelerator Defective
Aggressive Driving/Road Rage
Alcohol Involvement
Animals Action
Backing Unsafely
Brakes Defective
Cell Phone (hand-Held)
Cell Phone (hands-free)
Driver Inattention/Distraction
Driver Inexperience
Driverless/Runaway Vehicle
Drugs (Illegal)
Drugs (illegal)
Failure to Keep Right
Failure to Yield Right-of-Way
Fatigued/Drowsy
Fell Asleep
Following Too Closely
Glare
Illnes
Illness
Lane Marking Improper/Inadequate
Lost Consciousness
Obstruction/Debris
Other Electronic Device
Other Vehicular
Outside Car Distraction
Oversized Vehicle
Passenger Distraction
Passing or Lane Usage Improper
Passing Too Closely
Pavement Defective
Pavement Slippery
Pedestrian/Bicyclist/Other Pedestrian Error/Confusion
Physical Disability
Prescription Medication
Reaction to Other Uninvolved Vehicle
Reaction to Uninvolved Vehicle
Steering Failure
Tinted Windows
Tire Failure/Inadequate
Traffic Control Device Improper/Non-Working
Traffic Control Disregarded
Turning Improperly
Unsafe Lane Changing
Unsafe Speed
Unspecified
View Obstructed/Limited
NA
Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 3"]]


count
1e+05
1e+03
1e+01
Accelerator Defective
Aggressive Driving/Road Rage
Alcohol Involvement
Animals Action
Backing Unsafely
Brakes Defective
Driver Inattention/Distraction
Driver Inexperience
Driverless/Runaway Vehicle
Drugs (Illegal)
Drugs (illegal)
Failure to Keep Right
Failure to Yield Right-of-Way
Fatigued/Drowsy
Fell Asleep
Following Too Closely
Glare
Illness
Lost Consciousness
Obstruction/Debris
Other Electronic Device
Other Vehicular
Outside Car Distraction
Passing or Lane Usage Improper
Passing Too Closely
Pavement Defective
Pavement Slippery
Physical Disability
Prescription Medication
Reaction to Uninvolved Vehicle
Steering Failure
Traffic Control Device Improper/Non-Working
Traffic Control Disregarded
Turning Improperly
Unsafe Lane Changing
Unsafe Speed
Unspecified
View Obstructed/Limited
Windshield Inadequate
NA
Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 4"]]

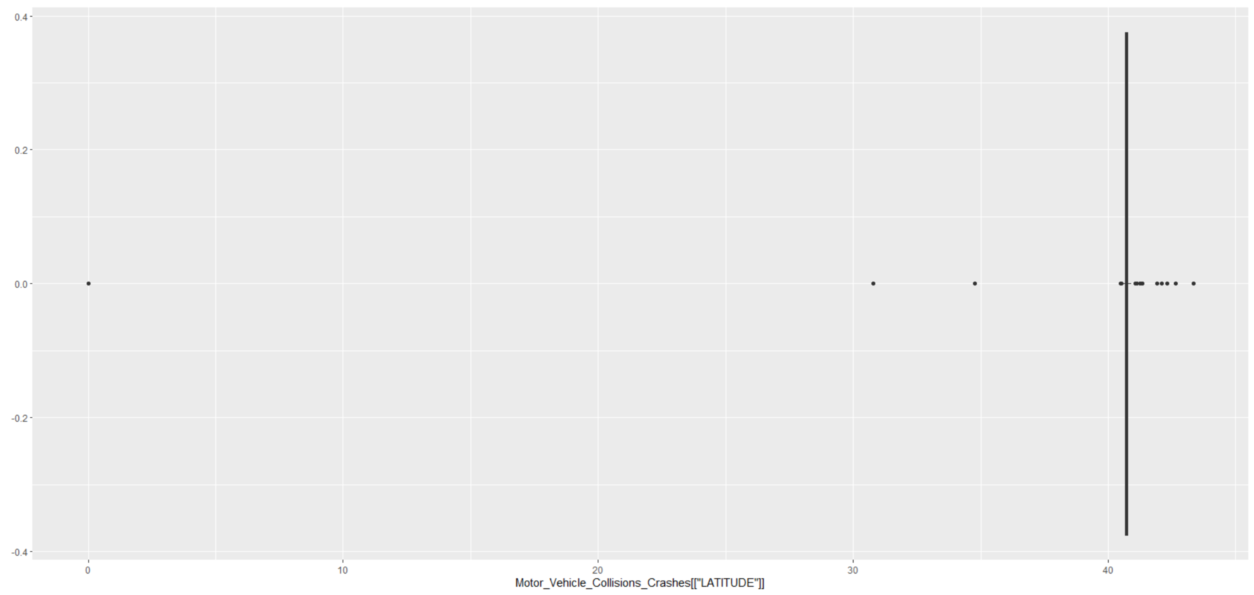

count
1e+05
1e+03
1e+01
Aggressive Driving/Road Rage
Alcohol Involvement
Backing Unsafely
Brakes Defective
Driver Inattention/Distraction
Driver Inexperience
Drugs (illegal)
Failure to Keep Right
Failure to Yield Right-of-Way
Fatigued/Drowsy
Fell Asleep
Following Too Closely
Glare
Illness
Obstruction/Debris
Other Electronic Device
Other Vehicular
Outside Car Distraction
Passing or Lane Usage Improper
Passing Too Closely
Pavement Defective
Pavement Slippery
Reaction to Uninvolved Vehicle
Steering Failure
Tire Failure/Inadequate
Traffic Control Device Improper/Non-Working
Traffic Control Disregarded
Unsafe Speed
Unspecified
NA
Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 5"]]

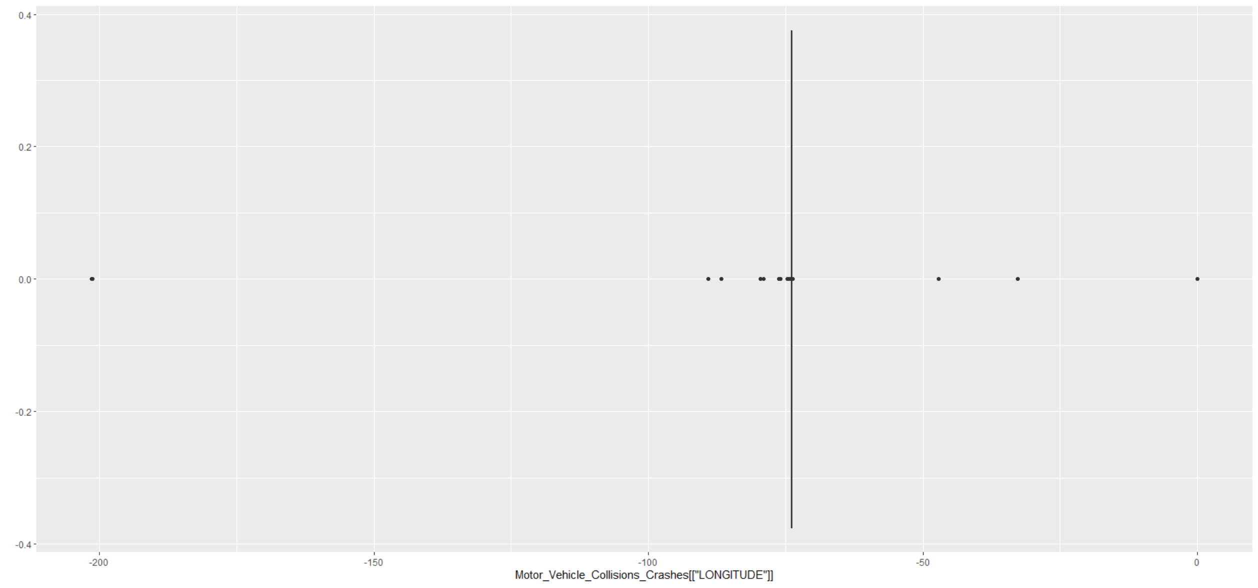
So here we have the factors for up to 5 vehicles in the crash. The most obvious trend is that the number of N/A entries increases with each vehicle i.e. vehicle 1 has a lot fewer N/A entries than vehicle 5. To be expected crashes with fewer vehicles involved are more common. This does however highlight to me that this data set has no value that specifically highlights how many vehicles were actually involved, which could be a good comparison point with this contributing factor data. The way that causes are distributed between each vehicle seems mostly similar just less quantity in each category when you move on to each successive vehicle. It is unclear from viewing this distribution whether there is any relationship between the reasons for each vehicle and further analysis will be needed to determine whether there is any.

For analysing location I created a scatterplot to show where the coordinates were mostly clustered.



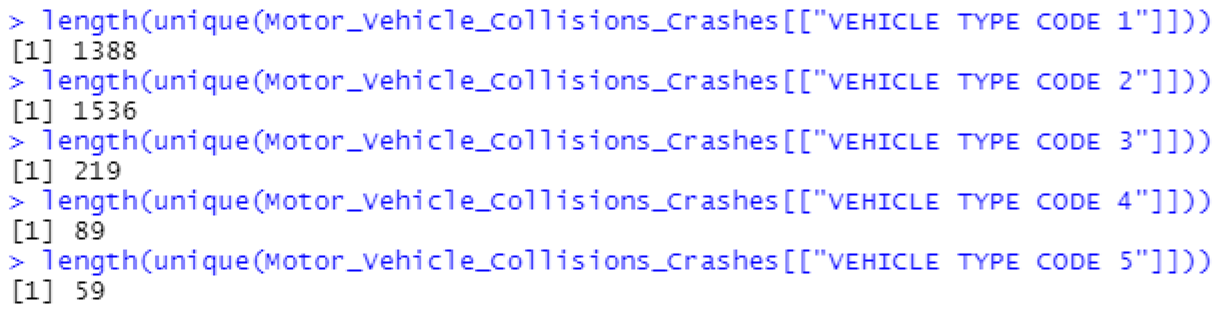
Now most points you can see are clustered around Lattitude 41 and Longitude -75. This area is New York City. There seem to be a lot of outliers that are well out of the bounds of a crash dataset for NYC. These outliers are made even clearer when we analyse the box plots for each.





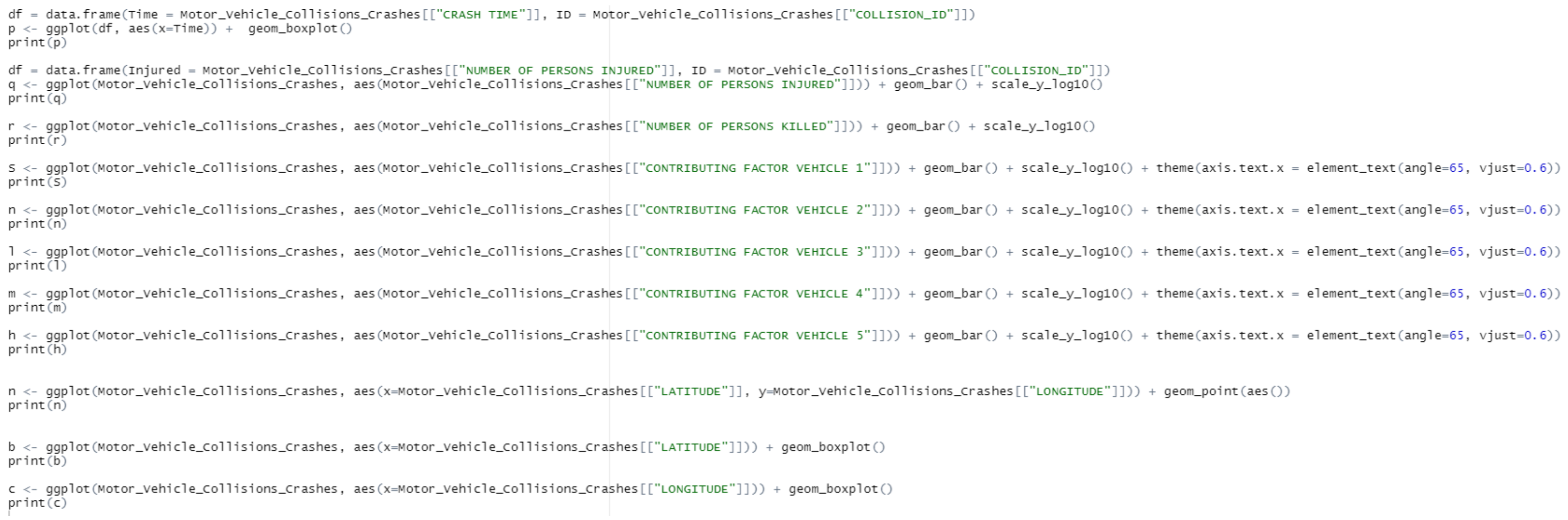
We see the vast majority of the points are in a tiny sliver of the data set and there are quite a few outliers as seen in both box plots. Therefore it seems as if it will be important to remove all these outliers. I will find these outliers later in this EDA.

Lastly, we will check the vehicle codes. Now there is a slight issue with graphing this as for Vehicle Type code 1 there are 1388 unique categories and for Vehicle type code 2 there are 1538 unique categories and Vehicle Type code 3 has 219, all too many to plot in a single bar chart so we will skip over these. From looking at what charts it created it wasn't imbalanced so we will skip it.



```
> length(unique(Motor_Vehicle_Collisions_Crashes[["VEHICLE TYPE CODE 1"]]))
[1] 1388
> length(unique(Motor_Vehicle_Collisions_Crashes[["VEHICLE TYPE CODE 2"]]))
[1] 1536
> length(unique(Motor_Vehicle_Collisions_Crashes[["VEHICLE TYPE CODE 3"]]))
[1] 219
> length(unique(Motor_Vehicle_Collisions_Crashes[["VEHICLE TYPE CODE 4"]]))
[1] 89
> length(unique(Motor_Vehicle_Collisions_Crashes[["VEHICLE TYPE CODE 5"]]))
[1] 59
```

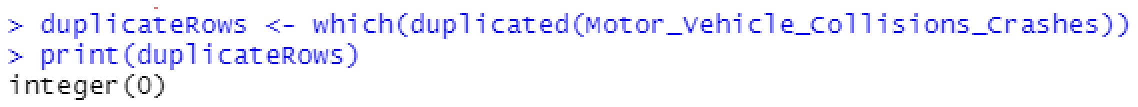
Below is the code to carry out all these plots:



```
df = data.frame(Time = Motor_Vehicle_Collisions_Crashes[["CRASH TIME"]], ID = Motor_Vehicle_Collisions_Crashes[["COLLISION_ID"]])
p <- ggplot(df, aes(x=Time)) +  geom_boxplot()
print(p)

df = data.frame(Injured = Motor_Vehicle_Collisions_Crashes[["NUMBER OF PERSONS INJURED"]], ID = Motor_Vehicle_Collisions_Crashes[["COLLISION_ID"]])
q <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["NUMBER OF PERSONS INJURED"]])) + geom_bar() + scale_y_log10()
print(q)

r <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["NUMBER OF PERSONS KILLED"]])) + geom_bar() + scale_y_log10()
print(r)

s <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 1"]])) + geom_bar() + scale_y_log10() + theme(axis.text.x = element_text(angle=65, vjust=0.6))
print(s)

n <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 2"]])) + geom_bar() + scale_y_log10() + theme(axis.text.x = element_text(angle=65, vjust=0.6))
print(n)

l <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 3"]])) + geom_bar() + scale_y_log10() + theme(axis.text.x = element_text(angle=65, vjust=0.6))
print(l)

m <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 4"]])) + geom_bar() + scale_y_log10() + theme(axis.text.x = element_text(angle=65, vjust=0.6))
print(m)

h <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(Motor_Vehicle_Collisions_Crashes[["CONTRIBUTING FACTOR VEHICLE 5"]])) + geom_bar() + scale_y_log10() + theme(axis.text.x = element_text(angle=65, vjust=0.6))
print(h)


n <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(x=Motor_Vehicle_Collisions_Crashes[["LATITUDE"]], y=Motor_Vehicle_Collisions_Crashes[["LONGITUDE"]])) + geom_point(aes())
print(n)


b <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(x=Motor_Vehicle_Collisions_Crashes[["LATITUDE"]])) + geom_boxplot()
print(b)

c <- ggplot(Motor_Vehicle_Collisions_Crashes, aes(x=Motor_Vehicle_Collisions_Crashes[["LONGITUDE"]])) + geom_boxplot()
print(c)
```

6. Checking for duplicates

I next just wanted to check if there were any duplicate rows. There actually were none which is convenient for me.



```
> duplicateRows <- which(duplicated(Motor_Vehicle_Collisions_Crashes))
> print(duplicateRows)
integer(0)
```

7. EDA Conclusion

I have achieved a few clear objectives here in my EDA. The first is ensuring my data can be adequately loaded and that there actually is enough data present for me to run a successful analysis. The second is I have been able to analyse and recognise what data can and should be removed through various factors, whether due to an abundance of n/a values or outliers in the specific sets of data. The third is I have gained a strong understanding of my data, what trends are already clearly visible in the distribution of the labels and allowing me to understand any bias that may arise from my future analysis.

I have also come to realise my data is mostly categorical with not many continuous values. The only continuous values are location. The rest is categorical, this is important to realise when picking the best analysis techniques for my data.

8. Objective

I have decided that my objective is to see based on all other data whether I can find the primary reasoning behind the crash, so the reasoning of the first vehicle is listed. If we can accurately predict the reasoning behind the crash based on all other predictors that would accurately demonstrate that location, time, date and number of people injured or killed can predict what accident occurred, allowing for possible changes to make to prevent such accidents.

9. Removing the unneeded variables

Before I begin my analyse I must remove the entries I have deemed unnecessary or unimportant to the data from the EDA. These variables include.

- Entries with n/a values for the number of people injured
- Entries with n/a values for the number of people killed
- Entries with n/a values for the Contributing Factor of vehicle 1
- Entries with n/a values for the Longitude and Latitude
- Outliers from location

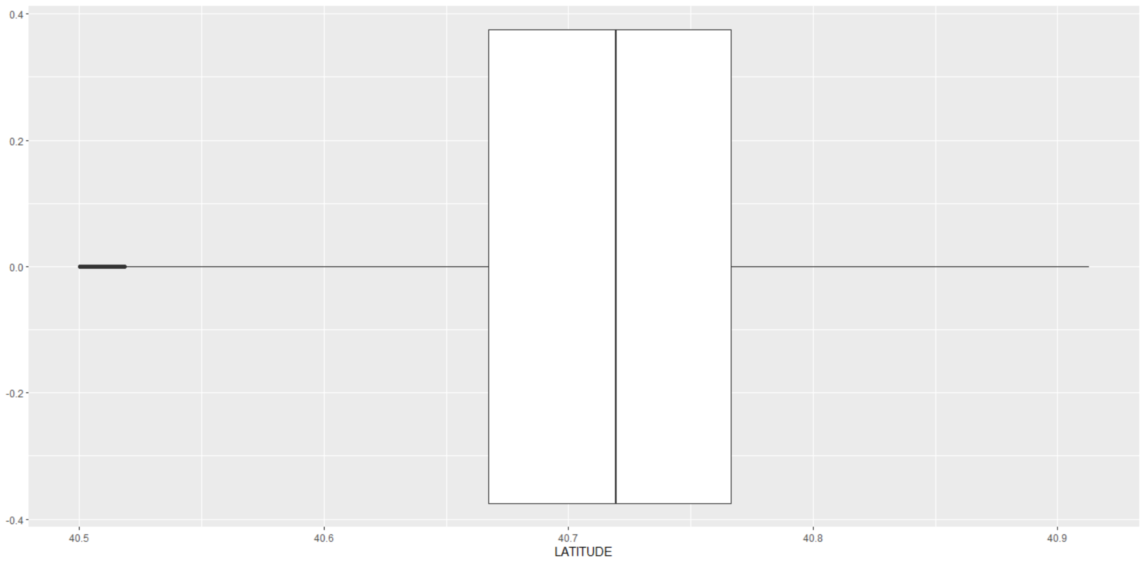
As mentioned in the EDA it may be worth removing the 220620 entries with n/a values for longitude and latitude if I wish to do location analyses. I think this overall is worth the tradeoff and as I have ~1.9 million entries it isn't that big a dent regardless.

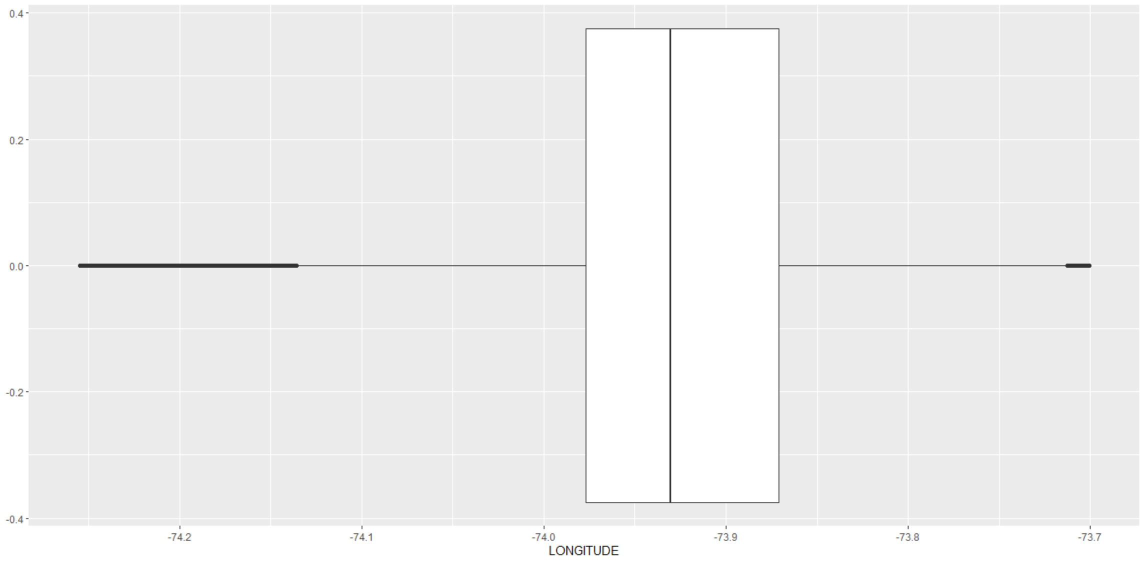
However, when going through the variables I decided to veer slightly from my original conclusions in the EDA and also remove the entries with n/a values for

- Borough
- Zip Code

This still provides us with 1273640 entries which are still over a million. I considered removing n/a values for on-street name and cross-street name but then I realised that sometimes there just might not be an on-street or cross-street and realised that would skew the data

Next is removing the outliers in the longitude and latitude. Based on the EDA we can see where the vast majority of data lies.

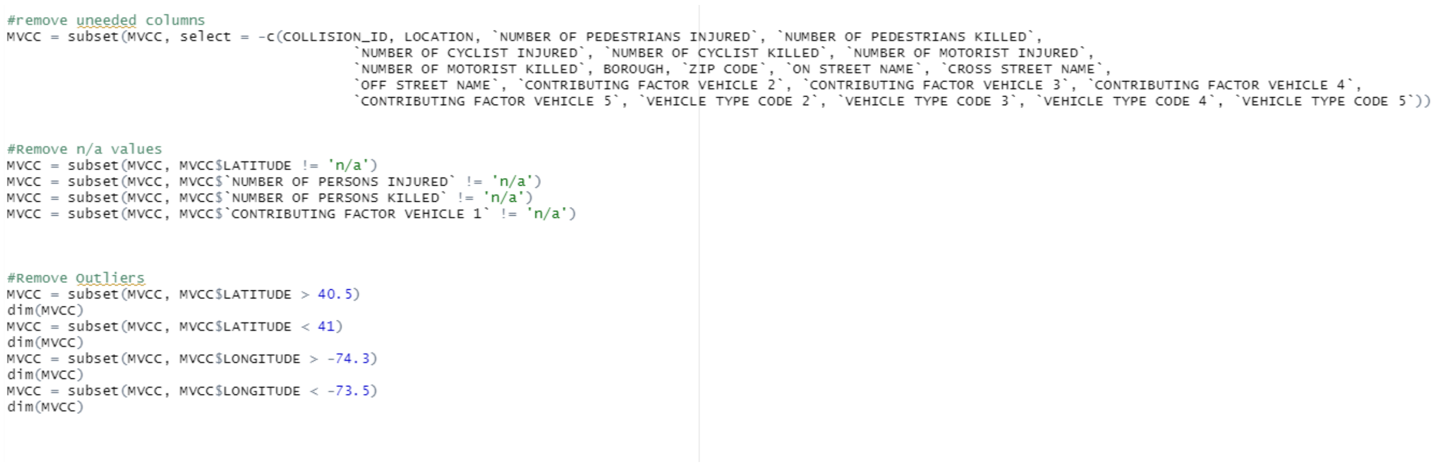




Above you can see, especially when compared with the earlier boxplots that we have a very reasonable range now.

With that our data is now ready and adequately pruned and we are left with 1270990 entries still, which is more than enough for our analysis.

I am also going to remove a lot of the columns. The first one is removing all the location values that are not Longitude and Latitude. They all essentially say the same thing, it can all be deduced from the Longitude and Latitude so we can remove Zip Code, Borough, Off Street name, Cross Street name, On Street Name. Next, we will remove the fields for whether Motorists, Pedestrians and Cyclists specifically are killed or injured. Though there could be something to be gleaned from this information, my EDA showed they do not always line up with the total injured and killed and will add unneeded complexity to the analysis. Lastly, I will remove the Vehicle codes and all the Contributing factors other than the first. This way we can treat the primary reason as the contributing factor to vehicle 1 and proceed from there. This also again reduces complexity as we need fewer dimensions to run our algorithms. I did try running it with the full dataset and it wouldn't let me as a 955GB vector would have to be created which R would not allow because of the immense size.

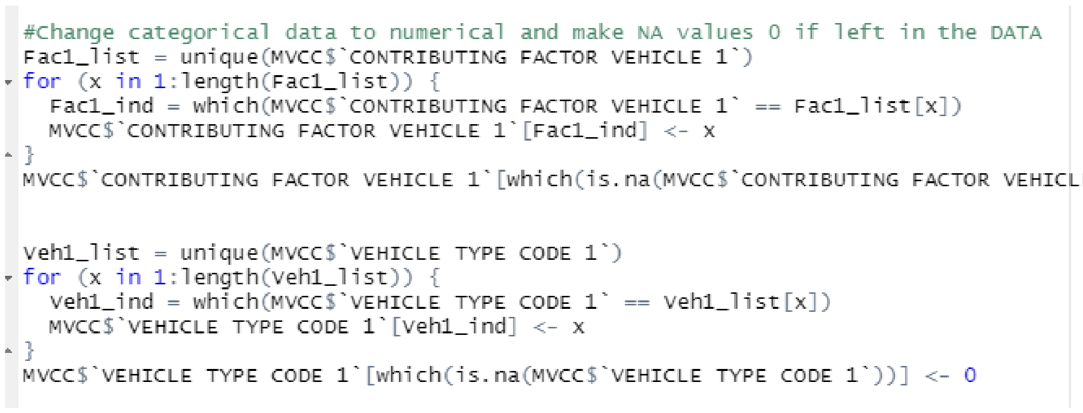


```
#remove uneeded columns
MVCC = subset(MVCC, select = -c(COLLISION_ID, LOCATION, `NUMBER OF PEDESTRIANS INJURED`, `NUMBER OF PEDESTRIANS KILLED`,
                                 `NUMBER OF CYCLIST INJURED`, `NUMBER OF CYCLIST KILLED`, `NUMBER OF MOTORIST INJURED`,
                                 `NUMBER OF MOTORIST KILLED`, BOROUGH, `ZIP CODE`, `ON STREET NAME`, `CROSS STREET NAME`,
                                 `OFF STREET NAME`, `CONTRIBUTING FACTOR VEHICLE 2`, `CONTRIBUTING FACTOR VEHICLE 3`, `CONTRIBUTING FACTOR VEHICLE 4`,
                                 `CONTRIBUTING FACTOR VEHICLE 5`, `VEHICLE TYPE CODE 2`, `VEHICLE TYPE CODE 3`, `VEHICLE TYPE CODE 4`, `VEHICLE TYPE CODE 5`))


#Remove n/a values
MVCC = subset(MVCC, MVCC$LATITUDE != 'n/a')
MVCC = subset(MVCC, MVCC$`NUMBER OF PERSONS INJURED` != 'n/a')
MVCC = subset(MVCC, MVCC$`NUMBER OF PERSONS KILLED` != 'n/a')
MVCC = subset(MVCC, MVCC$`CONTRIBUTING FACTOR VEHICLE 1` != 'n/a')


#Remove Outliers
MVCC = subset(MVCC, MVCC$LATITUDE > 40.5)
dim(MVCC)
MVCC = subset(MVCC, MVCC$LATITUDE < 41)
dim(MVCC)
MVCC = subset(MVCC, MVCC$LONGITUDE > -74.3)
dim(MVCC)
MVCC = subset(MVCC, MVCC$LONGITUDE < -73.5)
dim(MVCC)
```

Above is the code used for all described above. It is slightly less as I removed columns for some values I determined it was okay to remove n/a values for.

10. Making Categorical Variables numerical

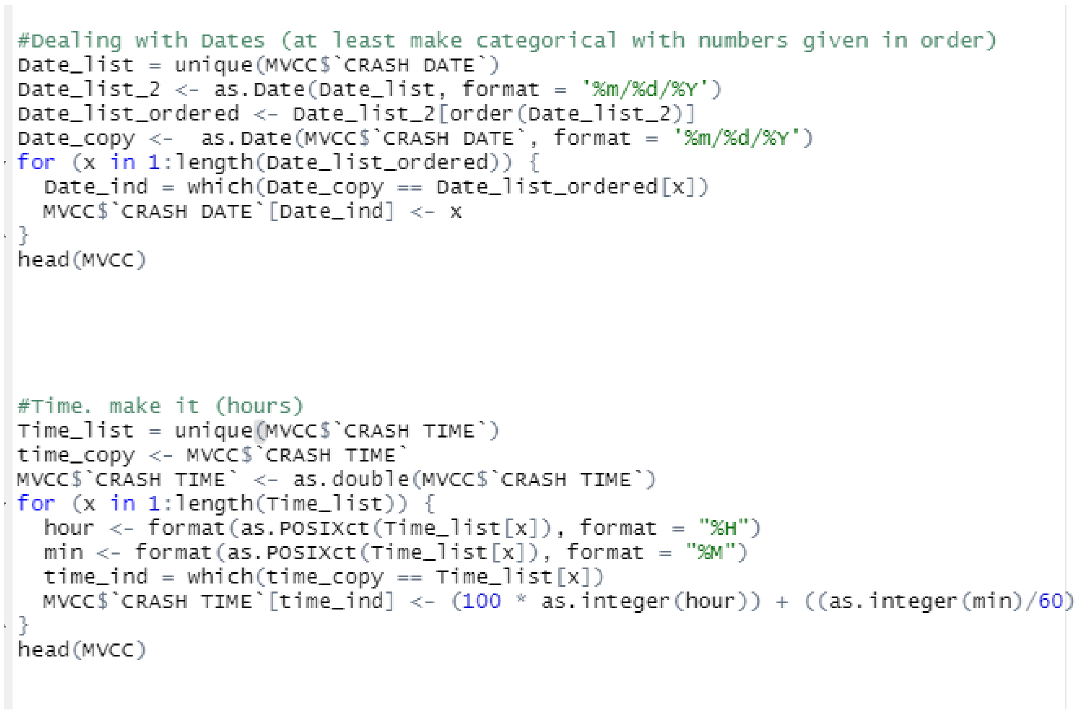
As seen in this analysis we have an abundance of categorical variables, some of which are not numeric. We need to convert these to numerical values to run our algorithms on it. To do this we run code to get a list of all the unique values of the feature, then make the index of that value in the list the representation of that categorical value. We do this for the 'Contributing factor for Vehicle 1' and `Vehicle Type Code 1` as these are the only remaining categorical variables. I also made NA values 0 if they were still some in the dataset.



```r
#Change categorical data to numerical and make NA values 0 if left in the DATA
Fac1_list = unique(MVCC$`CONTRIBUTING FACTOR VEHICLE 1`)
for (x in 1:length(Fac1_list)) {
  Fac1_ind = which(MVCC$`CONTRIBUTING FACTOR VEHICLE 1` == Fac1_list[x])
  MVCC$`CONTRIBUTING FACTOR VEHICLE 1`[Fac1_ind] <- x
}
MVCC$`CONTRIBUTING FACTOR VEHICLE 1`[which(is.na(MVCC$`CONTRIBUTING FACTOR VEHICLE 1`))] <- 0


Veh1_list = unique(MVCC$`VEHICLE TYPE CODE 1`)
for (x in 1:length(Veh1_list)) {
  Veh1_ind = which(MVCC$`VEHICLE TYPE CODE 1` == Veh1_list[x])
  MVCC$`VEHICLE TYPE CODE 1`[Veh1_ind] <- x
}
MVCC$`VEHICLE TYPE CODE 1`[which(is.na(MVCC$`VEHICLE TYPE CODE 1`))] <- 0
```

11. Time and Dates

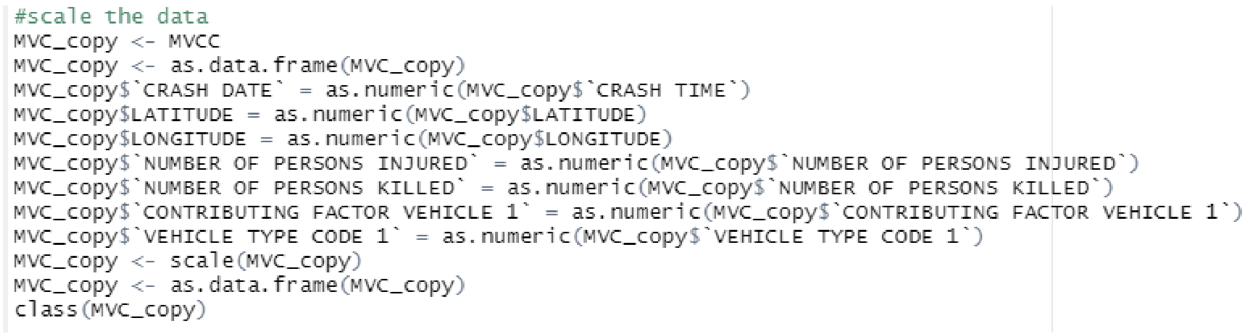
I had to think exactly what was the best way to deal with the dates and time. They provided separate fields and this suited the task well to keep them separate. For dates, I settled on ordering them in a list and giving them a categorical number based on the index of the date in the ordered list. This at least somewhat shows the passage of time with larger dates being later. For the time I made it so that it was represented as (hour * 100) + (min*100/60). This way time is evenly spaced in a range of 0 to 2400.



```r
#Dealing with Dates (at least make categorical with numbers given in order)
Date_list = unique(MVCC$`CRASH DATE`)
Date_list_2 <- as.Date(Date_list, format = '%m/%d/%Y')
Date_list_ordered <- Date_list_2[order(Date_list_2)]
Date_copy <-  as.Date(MVCC$`CRASH DATE`, format = '%m/%d/%Y')
for (x in 1:length(Date_list_ordered)) {
  Date_ind = which(Date_copy == Date_list_ordered[x])
  MVCC$`CRASH DATE`[Date_ind] <- x
}
head(MVCC)




#Time. make it (hours)
Time_list = unique(MVCC$`CRASH TIME`)
time_copy <- MVCC$`CRASH TIME`
MVCC$`CRASH TIME` <- as.double(MVCC$`CRASH TIME`)
for (x in 1:length(Time_list)) {
  hour <- format(as.POSIXct(Time_list[x]), format = "%H")
  min <- format(as.POSIXct(Time_list[x]), format = "%M")
  time_ind = which(time_copy == Time_list[x])
  MVCC$`CRASH TIME`[time_ind] <- (100 * as.integer(hour)) + ((as.integer(min)/60)*100)
}
head(MVCC)
```

12. Scaling the Data

So at this point we have all our data prepped, however, it is not scaled with drastically different values. Therefore we have to scale it so that it can work adequate well with the model. Also when I attempted it unscaled the models wouldn't run.

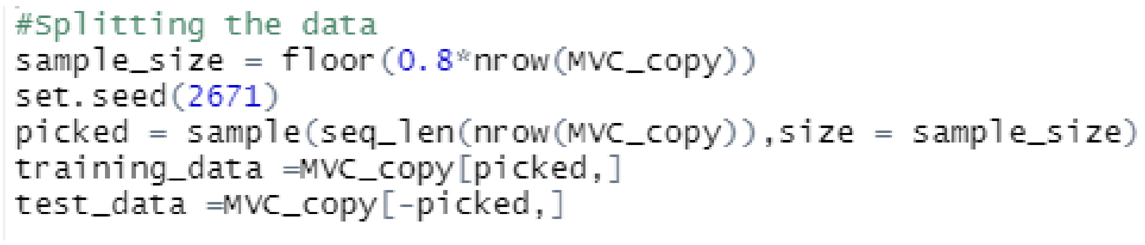


```
#scale the data
MVC_copy <- MVCC
MVC_copy <- as.data.frame(MVC_copy)
MVC_copy$`CRASH DATE` = as.numeric(MVC_copy$`CRASH TIME`)
MVC_copy$LATITUDE = as.numeric(MVC_copy$LATITUDE)
MVC_copy$LONGITUDE = as.numeric(MVC_copy$LONGITUDE)
MVC_copy$`NUMBER OF PERSONS INJURED` = as.numeric(MVC_copy$`NUMBER OF PERSONS INJURED`)
MVC_copy$`NUMBER OF PERSONS KILLED` = as.numeric(MVC_copy$`NUMBER OF PERSONS KILLED`)
MVC_copy$`CONTRIBUTING FACTOR VEHICLE 1` = as.numeric(MVC_copy$`CONTRIBUTING FACTOR VEHICLE 1`)
MVC_copy$`VEHICLE TYPE CODE 1` = as.numeric(MVC_copy$`VEHICLE TYPE CODE 1`)
MVC_copy <- scale(MVC_copy)
MVC_copy <- as.data.frame(MVC_copy)
class(MVC_copy)
```

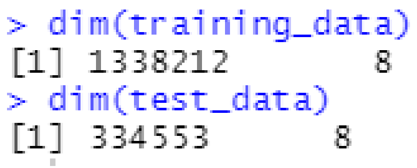
13. Splitting into training data and test data

Next to do our Machine Learning Analysis we need to split the data into the training set and test set. The training set is used to train our model and the test set is used to check the accuracy of the model we have produced.

Here you can see we take 80% of the data set as the training data and leave the remaining data for the test set. The training data has 1338212 entries and the test data 334553 entries.



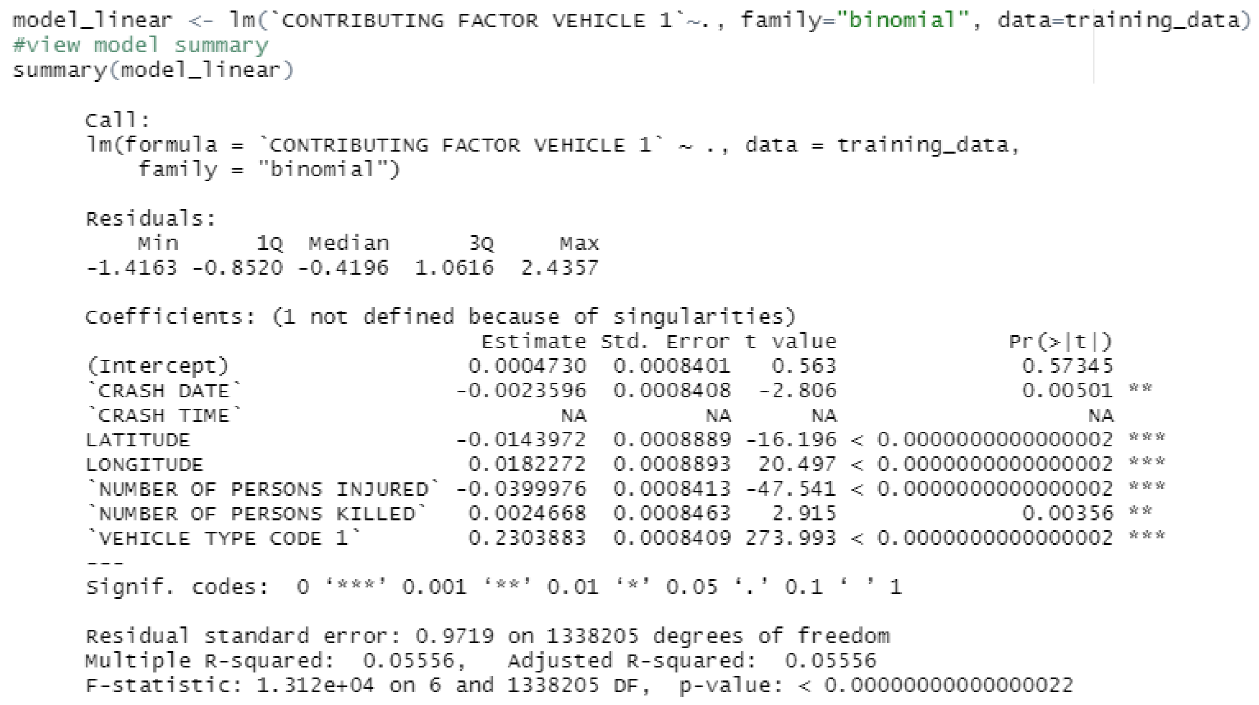
```
#Splitting the data
sample_size = floor(0.8*nrow(MVC_copy))
set.seed(2671)
picked = sample(seq_len(nrow(MVC_copy)),size = sample_size)
training_data =MVC_copy[picked,]
test_data =MVC_copy[-picked,]
```



```
> dim(training_data)
[1] 1338212       8
> dim(test_data)
[1] 334553       8
```

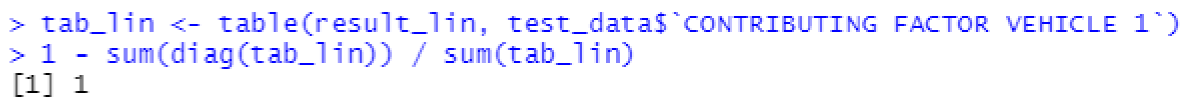
14. Linear Regression

We will begin by trying linear regression using the lm() function provided in the library caret. Even though this isn't required for this hw I felt seeing how it went.



```
model_linear <- lm(`CONTRIBUTING FACTOR VEHICLE 1`~., family="binomial", data=training_data)
#view model summary
summary(model_linear)

      Call:
      lm(formula = `CONTRIBUTING FACTOR VEHICLE 1` ~ ., data = training_data,
          family = "binomial")

      Residuals:
          Min      1Q  Median      3Q     Max
      -1.4163 -0.8520 -0.4196  1.0616  2.4357

      Coefficients: (1 not defined because of singularities)
                                 Estimate Std. Error t value             Pr(>|t|)
      (Intercept)               0.0004730  0.0008401   0.563              0.57345
      `CRASH DATE`             -0.0023596  0.0008408  -2.806              0.00501 **
      `CRASH TIME`                     NA         NA      NA                   NA
      LATITUDE                 -0.0143972  0.0008889 -16.196 < 0.0000000000000002 ***
      LONGITUDE                 0.0182272  0.0008893  20.497 < 0.0000000000000002 ***
      `NUMBER OF PERSONS INJURED` -0.0399976  0.0008413 -47.541 < 0.0000000000000002 ***
      `NUMBER OF PERSONS KILLED`   0.0024668  0.0008463   2.915              0.00356 **
      `VEHICLE TYPE CODE 1`      0.2303883  0.0008409 273.993 < 0.0000000000000002 ***
      ---
      Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      Residual standard error: 0.9719 on 1338205 degrees of freedom
      Multiple R-squared:  0.05556,   Adjusted R-squared:  0.05556
      F-statistic: 1.312e+04 on 6 and 1338205 DF,  p-value: < 0.00000000000000022
```

Things of note here. Pr(>t) essentially shows how useful a variable is in relation to what we are trying to predict, in this case, "Contributing Factor Vehicle1' They all are very low showing little to no relationship apparently. The largest is for Crash Date yet even then it is still very low. Next, is the number of people killed, which makes sense as it is most likely a good predictor of the severity of the crash. We then create a confusion matrix and check the misclassification rate.

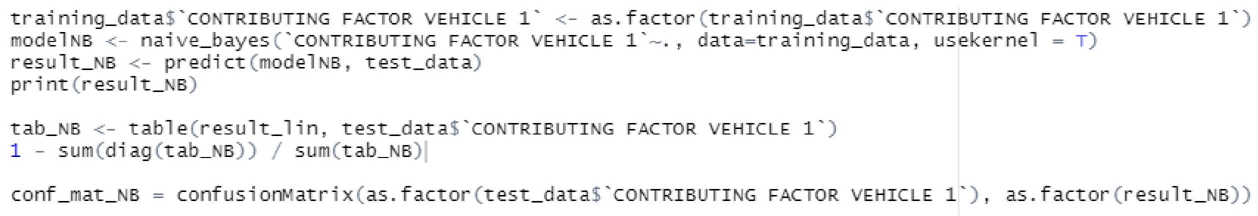


```
> tab_lin <- table(result_lin, test_data$`CONTRIBUTING FACTOR VEHICLE 1`)
> 1 - sum(diag(tab_lin)) / sum(tab_lin)
[1] 1
```

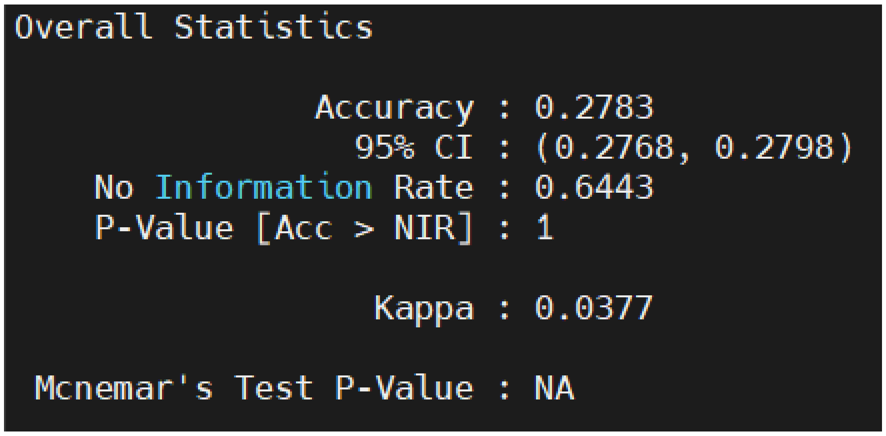
The model misclassifies 100% of the time which is obviously not good. My reasoning as to why is we are utilising a linear regression model. This is very likely not a linear model and trying linear regression in it has proved detrimental.

15. Naive Bayes

Next I try the Naive Bayes model. I am utilising the libraries naivebayes, dplyr and psych.

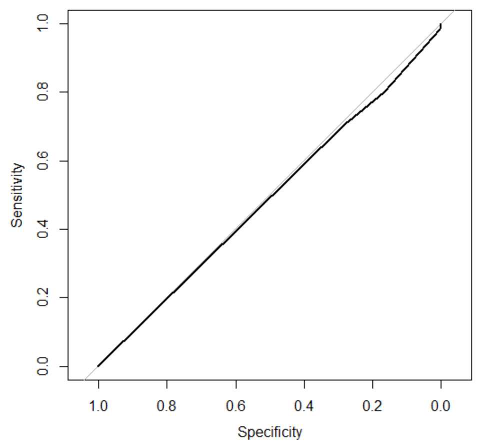


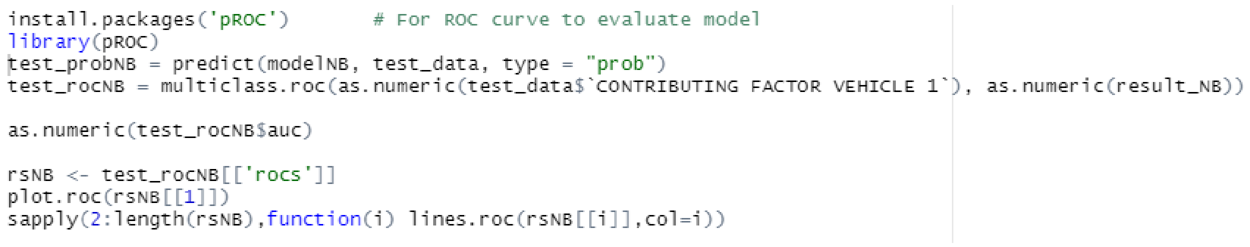
```
training_data$`CONTRIBUTING FACTOR VEHICLE 1` <- as.factor(training_data$`CONTRIBUTING FACTOR VEHICLE 1`)
modelNB <- naive_bayes(`CONTRIBUTING FACTOR VEHICLE 1`~., data=training_data, usekernel = T)
result_NB <- predict(modelNB, test_data)
print(result_NB)

tab_NB <- table(result_lin, test_data$`CONTRIBUTING FACTOR VEHICLE 1`)
1 - sum(diag(tab_NB)) / sum(tab_NB)

conf_mat_NB = confusionMatrix(as.factor(test_data$`CONTRIBUTING FACTOR VEHICLE 1`), as.factor(result_NB))
```



```
Overall Statistics

               Accuracy : 0.2783
                 95% CI : (0.2768, 0.2798)
    No Information Rate : 0.6443
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0377

 Mcnemar's Test P-Value : NA
```

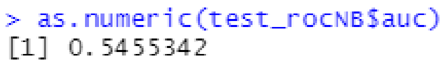
Straight away this is an improvement of having an accuracy of 0.2783 as opposed to 0 as seen in the linear regression model. The Kappa value shows us how well our model matched while also controlling for possible random variables. Kappa is in a range of 0 on 1, 0.0377 is a very low value but makes sense with an accuracy of only 0.2783%. Overall this clearly better than the linear regression model however still not great. I would say it did better because the assumptions of the model match up better with the situation we are analysing. It being multivariate help and allows for better accuracy. Also, it assumes that covariates are "conditionally" independent, this may demonstrate that my covariates are independent.





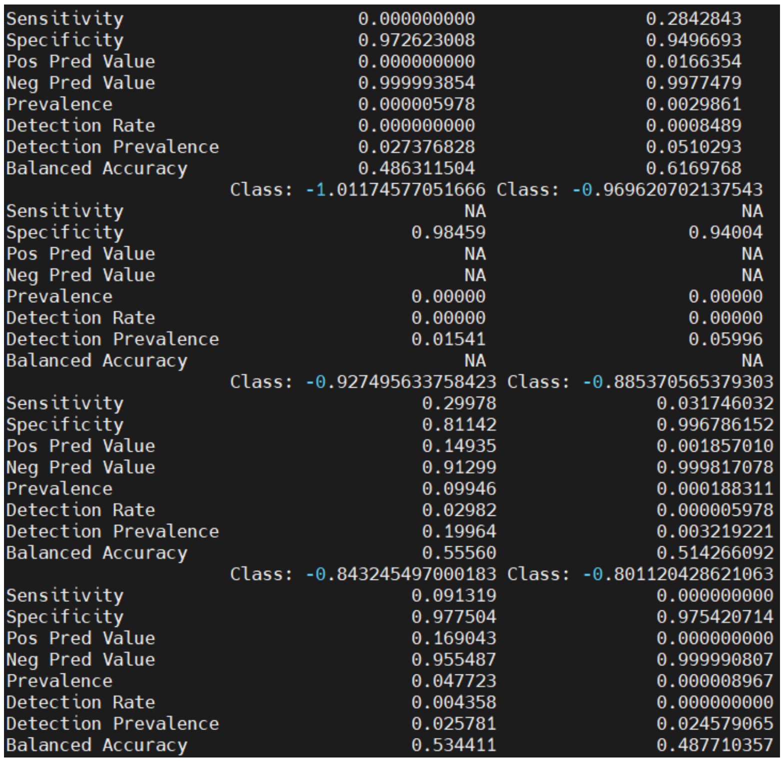
```
install.packages('pROC')       # For ROC curve to evaluate model
library(pROC)
test_probNB = predict(modelNB, test_data, type = "prob")
test_rocNB = multiclass.roc(as.numeric(test_data$`CONTRIBUTING FACTOR VEHICLE 1`), as.numeric(result_NB))

as.numeric(test_rocNB$auc)

rsNB <- test_rocNB[['rocs']]
plot.roc(rsNB[[1]])
sapply(2:length(rsNB),function(i) lines.roc(rsNB[[i]],col=i))
```

The ROC curve quite frankly demonstrates the same trend that the model has very bad classification abilities. We can also calculate our AUC.



```
> as.numeric(test_rocNB$auc)
[1] 0.5455342
```

It is 0.5455342, again demonstrating it has a very poor classification ability.

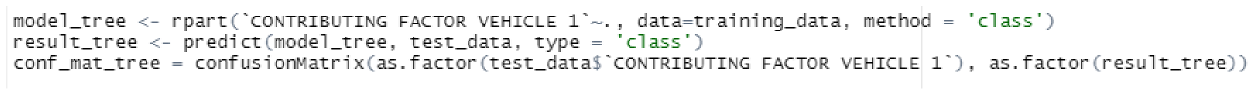


```
Sensitivity                     0.000000000              0.2842843
Specificity                     0.972623008              0.9496693
Pos Pred Value                  0.000000000              0.0166354
Neg Pred Value                  0.999993854              0.9977479
Prevalence                      0.000005978              0.0029861
Detection Rate                  0.000000000              0.0008489
Detection Prevalence            0.027376828              0.0510293
Balanced Accuracy               0.486311504              0.6169768
                     Class: -1.01174577051666 Class: -0.969620702137543
Sensitivity                                NA                        NA
Specificity                           0.98459                   0.94004
Pos Pred Value                             NA                        NA
Neg Pred Value                             NA                        NA
Prevalence                            0.00000                   0.00000
Detection Rate                        0.00000                   0.00000
Detection Prevalence                  0.01541                   0.05996
Balanced Accuracy                          NA                        NA
                     Class: -0.927495633758423 Class: -0.885370565379303
Sensitivity                             0.29978               0.031746032
Specificity                             0.81142               0.996786152
Pos Pred Value                          0.14935               0.001857010
Neg Pred Value                          0.91299               0.999817078
Prevalence                              0.09946               0.000188311
Detection Rate                          0.02982               0.000005978
Detection Prevalence                    0.19964               0.003219221
Balanced Accuracy                       0.55560               0.514266092
                     Class: -0.843245497000183 Class: -0.801120428621063
Sensitivity                            0.091319               0.000000000
Specificity                            0.977504               0.975420714
Pos Pred Value                         0.169043               0.000000000
Neg Pred Value                         0.955487               0.999990807
Prevalence                             0.047723               0.000008967
Detection Rate                         0.004358               0.000000000
Detection Prevalence                   0.025781               0.024579065
Balanced Accuracy                      0.534411               0.487710357
```

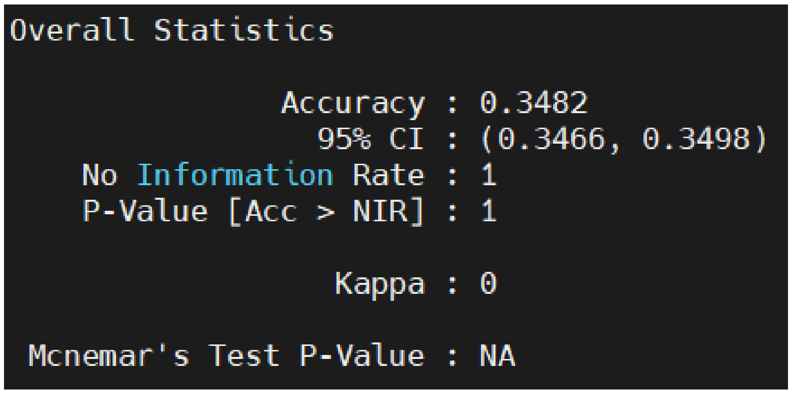
Looking at a selection of Sensitivity and Specificity we see that the sensitivity is close to 0 and the Specificity is close to ~0.97 in general. Essentially this means we can correctly identify what the 'Contributing Factor Vehicle 1' was almost never and we can predict when it isn't a certain value 97% of the time. However, this is mostly because the model does not perform well so it's wrong most of the time anyway. Ideally, both should be close to 1.

16. Decision Tree

Next we try the decision tree model. I decided to utilise this model as my target variable is categorical, as is the number of people killed, number of people injured and the vehicle code of vehicle 1. As such, I felt this model may perform well. Here we are utilising the model library rpart.

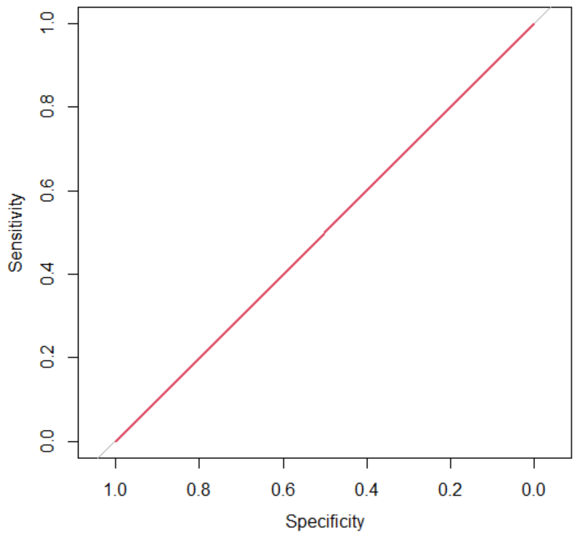


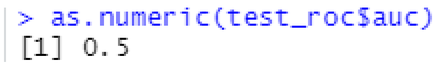
```
model_tree <- rpart(`CONTRIBUTING FACTOR VEHICLE 1`~., data=training_data, method = 'class')
result_tree <- predict(model_tree, test_data, type = 'class')
conf_mat_tree = confusionMatrix(as.factor(test_data$`CONTRIBUTING FACTOR VEHICLE 1`), as.factor(result_tree))
```



```
Overall Statistics

               Accuracy : 0.3482
                 95% CI : (0.3466, 0.3498)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : NA
```

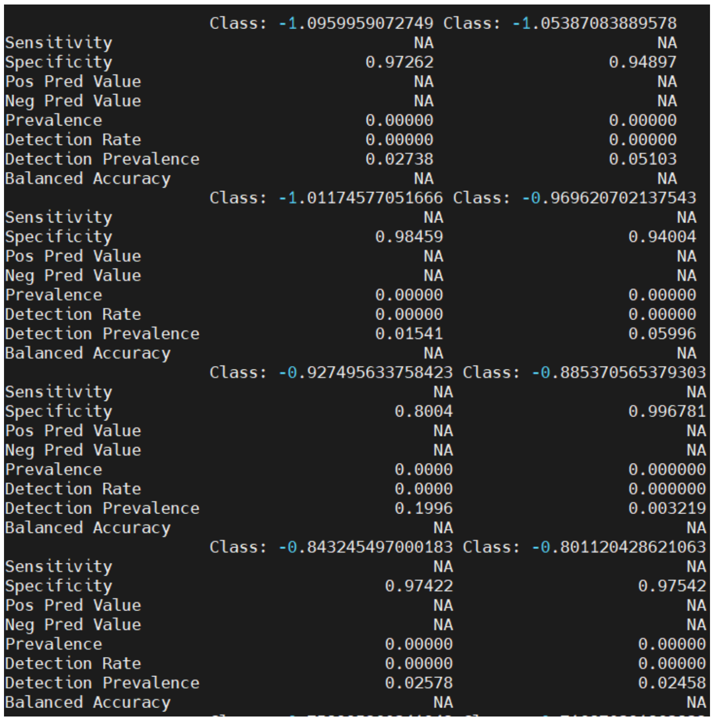
The decision tree model did have the best performance as expected. It had an accuracy of 0.3482 which is the highest yet, but still not great. What is interesting is the kappa is 0 for some reason. I am unsure exactly why that is because it clearly does predict well some of the times. It may have to do with the decision tree model itself but I am unsure.





```
> as.numeric(test_roc$auc)
[1] 0.5
```

Despite the higher accuracy, the ROC curve still shows poor classification and the AUC is exactly 0.5, also showing poor classification.

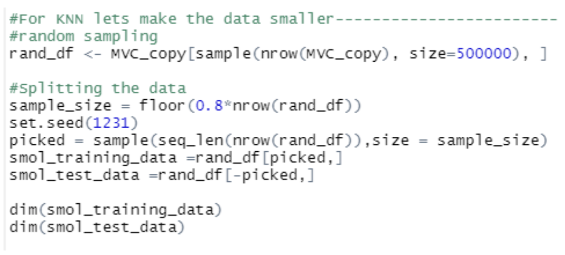


```
                     Class: -1.0959959072749 Class: -1.05387083889578
Sensitivity                               NA                       NA
Specificity                          0.97262                  0.94897
Pos Pred Value                            NA                       NA
Neg Pred Value                            NA                       NA
Prevalence                           0.00000                  0.00000
Detection Rate                       0.00000                  0.00000
Detection Prevalence                 0.02738                  0.05103
Balanced Accuracy                         NA                       NA
                     Class: -1.01174577051666 Class: -0.969620702137543
Sensitivity                                NA                        NA
Specificity                           0.98459                   0.94004
Pos Pred Value                             NA                        NA
Neg Pred Value                             NA                        NA
Prevalence                            0.00000                   0.00000
Detection Rate                        0.00000                   0.00000
Detection Prevalence                  0.01541                   0.05996
Balanced Accuracy                          NA                        NA
                     Class: -0.927495633758423 Class: -0.885370565379303
Sensitivity                                 NA                        NA
Specificity                             0.8004                  0.996781
Pos Pred Value                              NA                        NA
Neg Pred Value                              NA                        NA
Prevalence                              0.0000                  0.000000
Detection Rate                          0.0000                  0.000000
Detection Prevalence                    0.1996                  0.003219
Balanced Accuracy                           NA                        NA
                     Class: -0.843245497000183 Class: -0.801120428621063
Sensitivity                                 NA                        NA
Specificity                            0.97422                   0.97542
Pos Pred Value                              NA                        NA
Neg Pred Value                              NA                        NA
Prevalence                             0.00000                   0.00000
Detection Rate                         0.00000                   0.00000
Detection Prevalence                   0.02578                   0.02458
Balanced Accuracy                           NA                        NA
```

Looking at the Sensitivity and Speicfivty, we see a high specificity, but sensitivity is NA for some reason. I am not sure exactly why. But overall seems to have many overlaps with Naive Bayes but has a better accuracy overall.

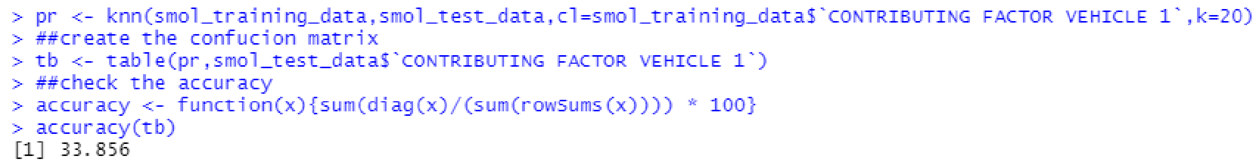
17. KNN

Lastly we will try KNN. To begin I have cut down my dataset and sampled it to 500,000 entries for performance purposes. It was taking too long with the full data set so I had to make it smaller.

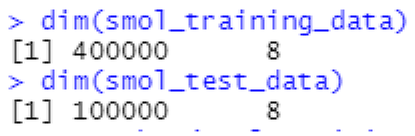


```
#For KNN lets make the data smaller------------------------
#random sampling
rand_df <- MVC_copy[sample(nrow(MVC_copy), size=500000), ]

#Splitting the data
sample_size = floor(0.8*nrow(rand_df))
set.seed(1231)
picked = sample(seq_len(nrow(rand_df)),size = sample_size)
smol_training_data =rand_df[picked,]
smol_test_data =rand_df[-picked,]

dim(smol_training_data)
dim(smol_test_data)
```

Now KNN tends to perform better with multi-label models therefore I had good hopes that it would perform the best so far.

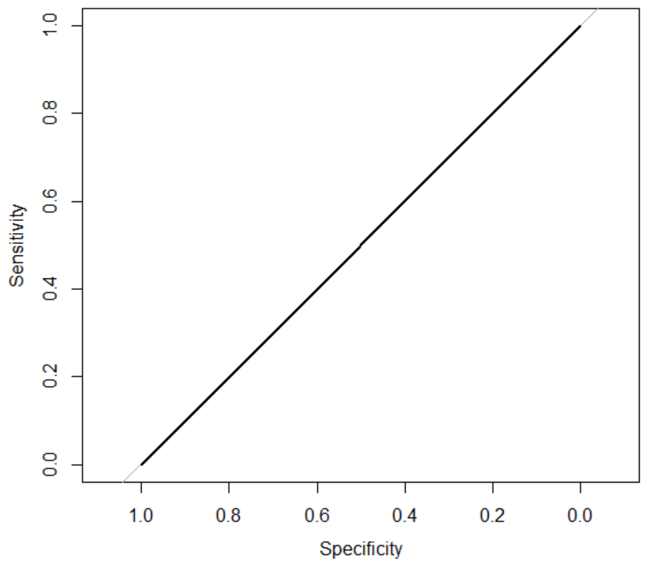


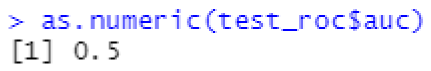
```
> pr <- knn(smol_training_data,smol_test_data,cl=smol_training_data$`CONTRIBUTING FACTOR VEHICLE 1`,k=20)
> ##create the confucion matrix
> tb <- table(pr,smol_test_data$`CONTRIBUTING FACTOR VEHICLE 1`)
> ##check the accuracy
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(tb)
[1] 33.856
```



```
> dim(smol_training_data)
[1] 400000      8
> dim(smol_test_data)
[1] 100000      8
```

Here we see the accuracy is 33.856. Still not high, actually lower than the accuracy for the Decision Tree model.





```
> as.numeric(test_roc$auc)
[1] 0.5
```

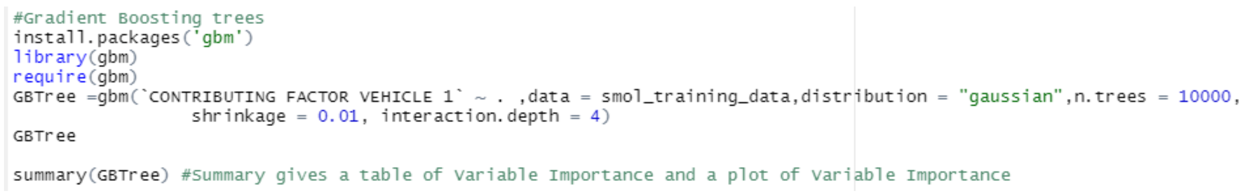
KNN keeps up the same trend of an AUC and ROC curve showing very bad classification. Overall appears all these models share the same shortcomings.

18. Hw1 Conclusion

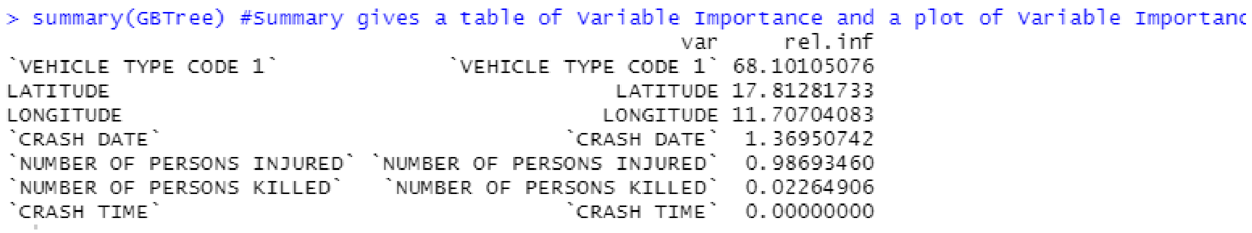
Overall, none of my models performed very well. I used progressively better-suited models and this is seen in that the accuracy does improve at each step. It could be a possibility that my data may not be very correlated at all and that the metrics I am picking to use do not predict the ‘Contributing Factor for Vehicle 1’ well at all. There is a lot of consistency in what the models are getting wrong which makes me believe the data just may not be well correlated in general, with all having bad ROC curves and AUC values. I will see what can be improved with further techniques in Homework 2.

19. Gradient Boosted Trees

So here I decided to try the Gradient Boosted Tree method. The Decision Tree method had the highest accuracy so I wanted to try it with gradient boosting. Now the main issue I had was my full dataset was too large to work so I used the smaller data set of 500,000 entries, split into a 400,000 training set and 100,000 test set.



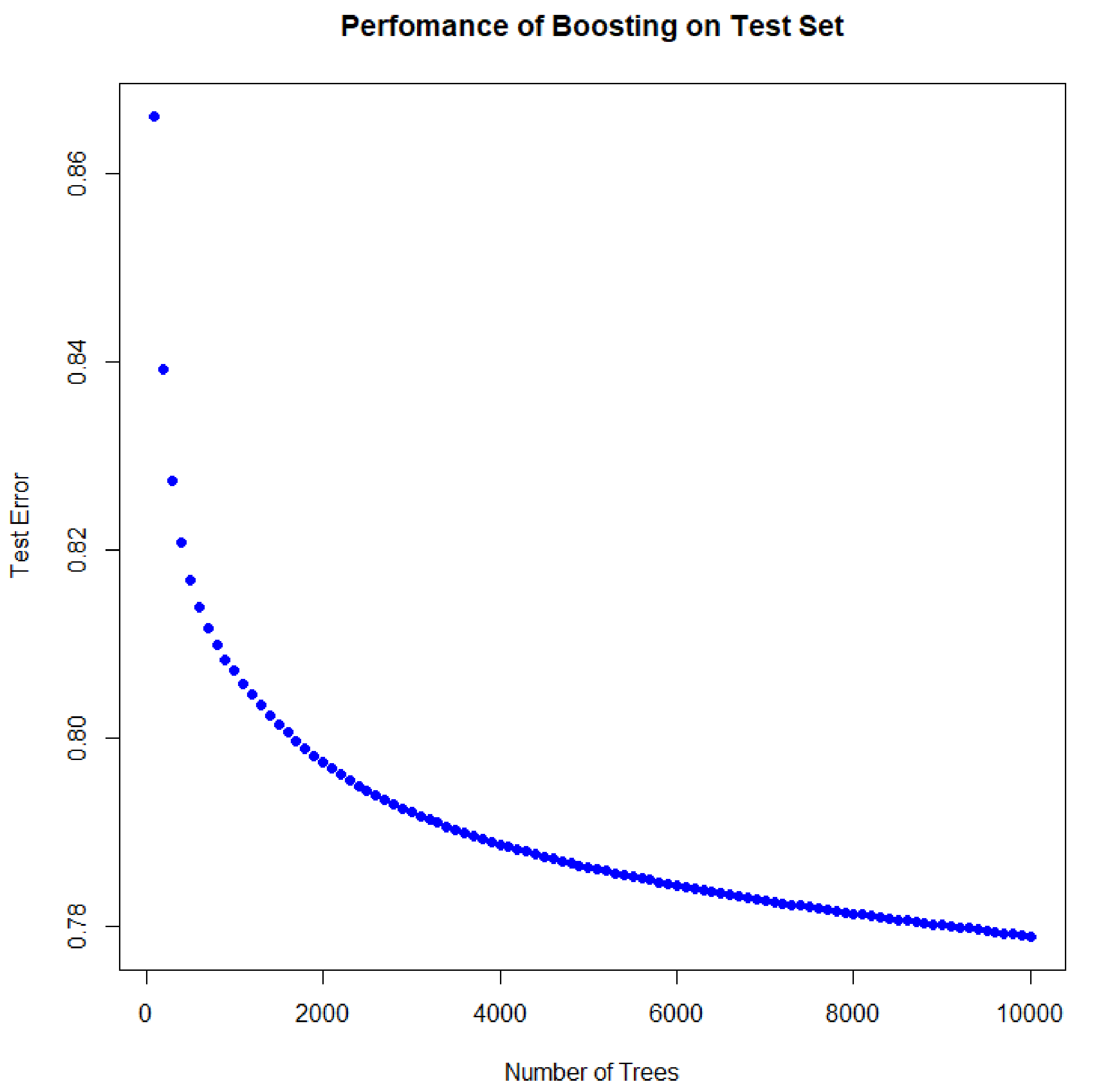
```
#Gradient Boosting trees
install.packages('gbm')
library(gbm)
require(gbm)
GBTree =gbm(`CONTRIBUTING FACTOR VEHICLE 1` ~ . ,data = smol_training_data,distribution = "gaussian",n.trees = 10000,
              shrinkage = 0.01, interaction.depth = 4)
GBTree

summary(GBTree) #Summary gives a table of Variable Importance and a plot of Variable Importance
```



```
> summary(GBTree) #Summary gives a table of Variable Importance and a plot of Variable Importanc
                                                          var      rel.inf
`VEHICLE TYPE CODE 1`                   `VEHICLE TYPE CODE 1` 68.10105076
LATITUDE                                             LATITUDE 17.81281733
LONGITUDE                                           LONGITUDE 11.70704083
`CRASH DATE`                                     `CRASH DATE`  1.36950742
`NUMBER OF PERSONS INJURED` `NUMBER OF PERSONS INJURED`  0.98693460
`NUMBER OF PERSONS KILLED`    `NUMBER OF PERSONS KILLED`  0.02264906
`CRASH TIME`                                     `CRASH TIME`  0.00000000
```

Initially what is interesting is that we can see what variables it finds to be the most important with the Vehicle Type being the highest. This is interesting as it shows that there may actually be a combination between certain vehicles and the occurrence of crashes. The next to are Latitude and Longitude. Also demonstrates that location plays an important role. The factors had a very low impact seemingly.

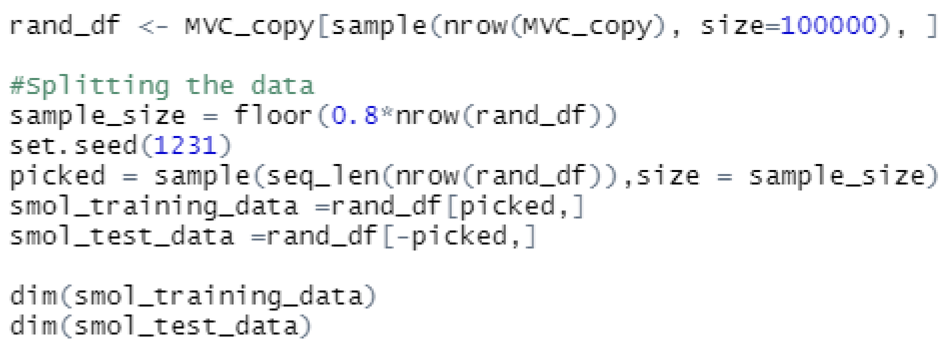
```r
n.trees = seq(from=100 ,to=10000, by=100) #no of trees-a vector of 100 values

#Generating a Prediction matrix for each Tree
predmatrix<-predict(GBTree, smol_training_data, n.trees = n.trees)
dim(predmatrix) #dimentions of the Prediction Matrix

#Calculating The Mean squared Test Error
test.error<-with(smol_training_data,apply( (predmatrix-`CONTRIBUTING FACTOR VEHICLE 1`)^2,2,mean))
head(test.error) #contains the Mean squared test error for each of the 100 trees averaged

#Plotting the test error vs number of trees

plot(n.trees , test.error , pch=19,col="blue",xlab="Number of Trees",ylab="Test Error", main = "Perfomance of Boosting on Test Set")
```



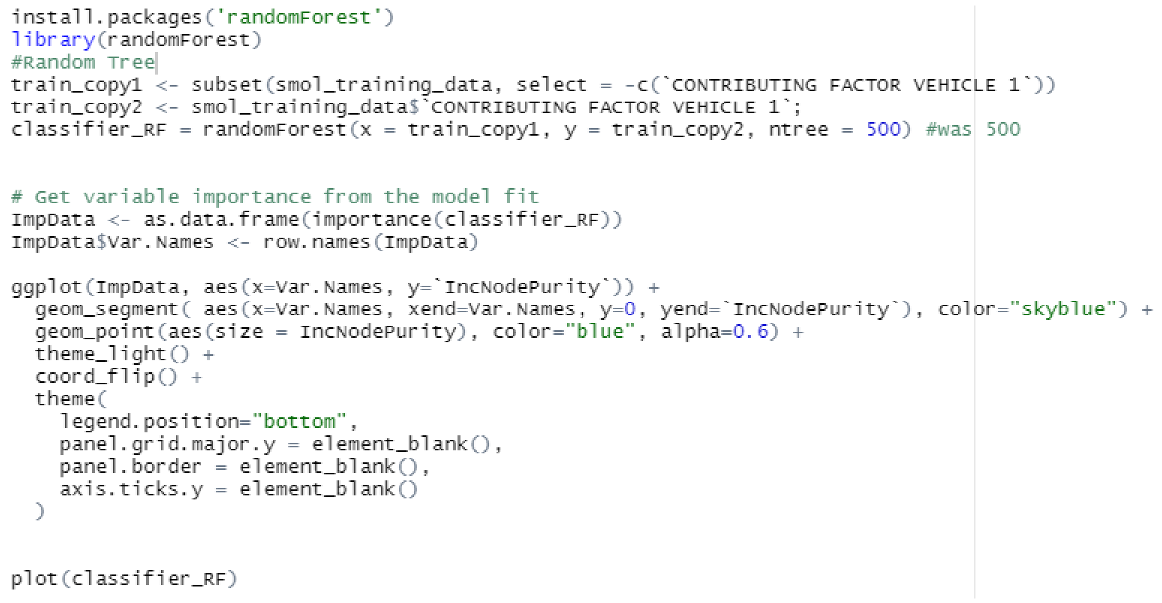
Next, we plot the test error and we see we get about as low as ~0.78. This is actually worse than our base decision tree model, which had an accuracy of ~0.348 which is a test error of ~0.652. So gradient boosting doesn't make up for having more than double the data points. As noted earlier, these techniques help when there is a lack of data, but with ~1.6 million entries my original project was not lacking in data so this technique has made little difference. However, it is interesting to see it come close with less than half as much data.

20. Random Forest

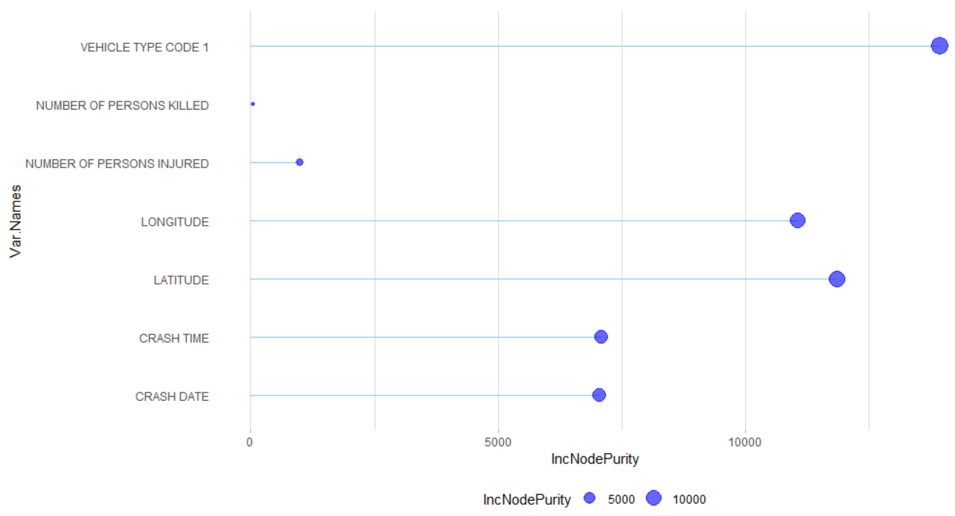
Now the random forest method was a technique I had a lot of hope for as it also pulls from the Decision Tree method which proved the best. However, I came to a similar issue as with the Gradient Boosting Trees where I just had too much data for the method to actually run, causing me to cut my data down to 100,000 entries and having the method run with 500 trees. This is not more than just an experiment to see if these other methods can match having much more data.

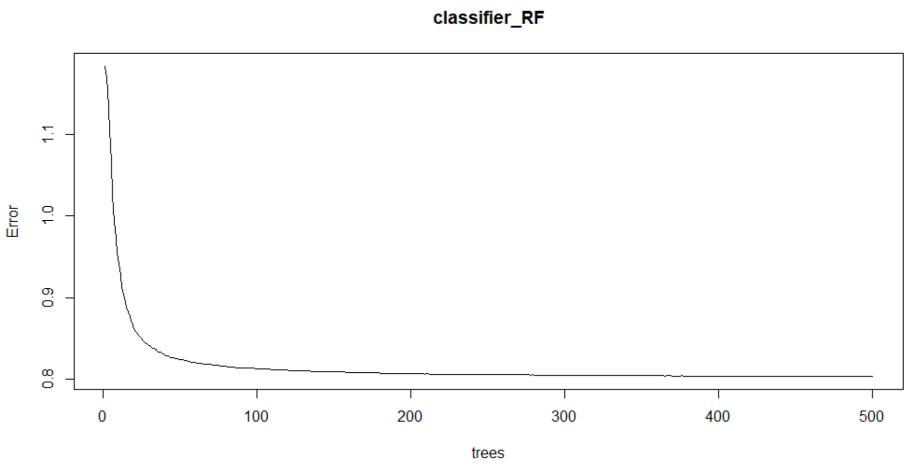


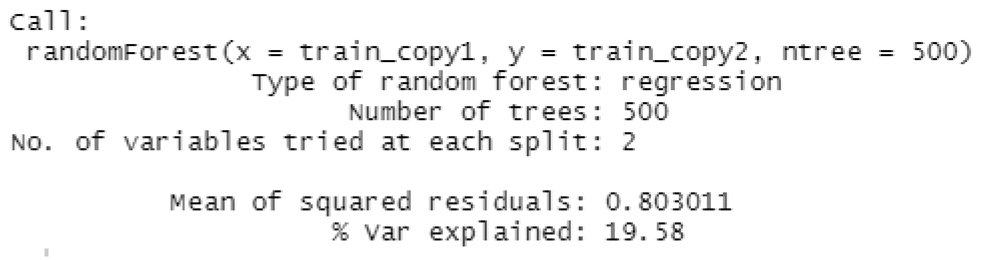
```
rand_df <- MVC_copy[sample(nrow(MVC_copy), size=100000), ]

#Splitting the data
sample_size = floor(0.8*nrow(rand_df))
set.seed(1231)
picked = sample(seq_len(nrow(rand_df)),size = sample_size)
smol_training_data =rand_df[picked,]
smol_test_data =rand_df[-picked,]

dim(smol_training_data)
dim(smol_test_data)
```



```
install.packages('randomForest')
library(randomForest)
#Random Tree
train_copy1 <- subset(smol_training_data, select = -c(`CONTRIBUTING FACTOR VEHICLE 1`))
train_copy2 <- smol_training_data$`CONTRIBUTING FACTOR VEHICLE 1`;
classifier_RF = randomForest(x = train_copy1, y = train_copy2, ntree = 500) #was 500


# Get variable importance from the model fit
ImpData <- as.data.frame(importance(classifier_RF))
ImpData$Var.Names <- row.names(ImpData)

ggplot(ImpData, aes(x=Var.Names, y=`IncNodePurity`)) +
  geom_segment( aes(x=Var.Names, xend=Var.Names, y=0, yend=`IncNodePurity`), color="skyblue") +
  geom_point(aes(size = IncNodePurity), color="blue", alpha=0.6) +
  theme_light() +
  coord_flip() +
  theme(
    legend.position="bottom",
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  )


plot(classifier_RF)
```







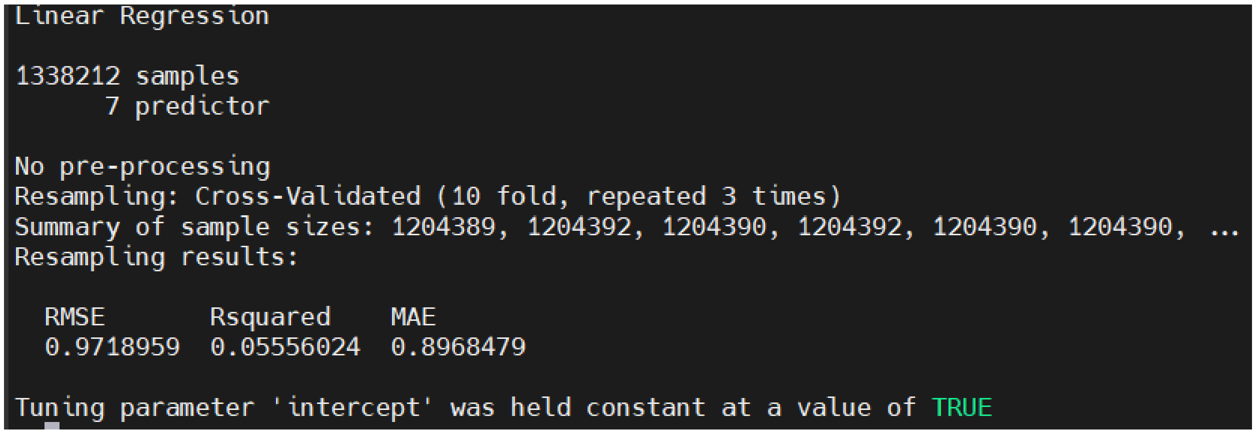
```
Call:
 randomForest(x = train_copy1, y = train_copy2, ntree = 500)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.803011
                    % Var explained: 19.58
```

Few things to note from this experience. The first main one is that the test error is still worse than the decision tree. At 500 Trees we are at ~0.8 error whereas the Decision tree had a test error of ~0.652. Looking at the node purity we see that VehicleType Code 1 is the purest by far with Longitude and Latitude next, then crash date and crash time. I will say that I didn't expect the number of people killed and injured to be the least. When I began this experiment I assumed the number of people injured or killed would be one of the clearest indications of what may cause a crash or the reasons that lead to it, so it is interesting to see every other node is purer. After the Vehicle itself, we still see how the next important thing is the location itself, showing certain locations may be more prone to certain crashes. It could be an interesting further study to see if you can predict location based on all other factors. We see the % var explained is 19.58. This is essentially the accuracy of the model, showing that it has a very poor accuracy as seen in the test error as well. If anything has been clear from these experiments, more data seems to beat out these techniques by far, which makes sense. These techniques improve your results on the current data set. If I could run these models on my full data set I am sure they may provide better results, though as you get more data it will most likely have diminishing returns.

21.10-fold Cross Validation and Linear Regression

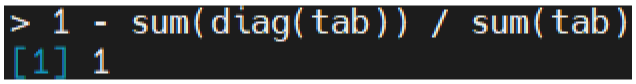
In all honesty I doubt cross-validation will do much to my accuracy for one main reason, my data set is already massive. For Linear Regression, Naive Bayes and Decision Tree method my training set had 1338212 entries, which is over a million. However, for the sake of experimentation, I will test whether cross-validation can improve the abysmal performance of Linear Regression from my first one. We do this using the Caret Library.

```
train_control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)

model <- train(`CONTRIBUTING FACTOR VEHICLE 1` ~., data = training_data,
               method = "lm",
               trControl = train_control)

print(model)
```



```
Linear Regression

1338212 samples
      7 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1204389, 1204392, 1204390, 1204392, 1204390, 1204390,  ...
Resampling results:

  RMSE       Rsquared    MAE
  0.9718959  0.05556024  0.8968479

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
results <- predict(model, test_data)
tab <- table(results, test_data$`CONTRIBUTING FACTOR VEHICLE 1`)
1 - sum(diag(tab)) / sum(tab)
```



```
> 1 - sum(diag(tab)) / sum(tab)
[1] 1
```

Here we try 10-fold cross-validation on the full training set which is 1338212 entries. We see pretty much the same results however with the original linear regression where the model misclassifies 100% of the time which is obviously not ideal.

22. Bias and Variance in Cross Validation

Let us analyse how bias and variance in the Cross Validation method. Here we utilised the k-fold cross validation method, specifically for k=10. This means that we split our data into 10 subsets. We then train the model using 9 of the subsets and using the remaining subset for the validation and testing. This means every data point is used to train k-1 times and used to validate 1 time. This reduces bias every data point is being used for training and reduces variance as all the data is also being used to validate the methods. Of course, there is always a bias-variance trade-off, but it is effective at getting both as small as possible. However, I feel as if my project demonstrated this is a great method for when you have less data, but when you have an abundance of data these techniques may be superfluous.

23. Homework 2 Conclusion

Overall, I have found that none of these methods has given me better results than in hw1 and for good reason. My data set was massive, to begin with, and to allow for most of these methods to run without R-Studio crashing or the IBM machine timing out I had to cut down the data I was using. However, Gradient Boosting and Random Forest did relatively well given how much less of the data they were given and therefore showed promise, however, did not make up for the difference in the number of entries. Analysis of the data set showed that the best predictor was in fact type of vehicle or vehicle code, even if the best accuracy I could get was ~30%. This could demonstrate that there is at least some correlation between certain vehicles and crashes. Next was the location, showing that there may be some locations that could be problem points. Again though we have such a low accuracy I can’t say any of this with confidence. This may be showing that there is a more random element to crashes that can not be predicted with the given data and show that we may need other data points.

24. Future Work and What I have learned

As brought up, it could be interesting to see if we can predict other things any better, such as crash location or vehicle type. However, I do feel as if overall the data set I picked might not have been the best for this type of project. It met the requirements set, but with mostly categorical data, features of low value with mostly n/a values and more it proved problematic. I wanted to explore something I found interesting but overall I learned how important the quality of data is. Furthermore, how to prep data, run the models and analyse the results were invaluable skills I learned, as well as coding in R, which was more intuitive than I at first expected.

25. References

- https://www.guru99.com/r-decision-trees.html
- http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Histogram
- https://www.r-bloggers.com/2021/04/naive-bayes-classification-in-r/
- https://www.tutorialspoint.com/r/r_linear_regression.htm
- https://www.journaldev.com/46732/confusion-matrix-in-r#:~:text=A%20confusion%20matrix%20in%20R,will%20represent%20the%20actual%20values.
- https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/
- https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
- https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c

- https://datascienceplus.com/gradient-boosting-in-r/

- https://hackernoon.com/random-forest-regression-in-r-code-and-interpretation

- https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/#:~:text=Random%20Forest%20in%20R%20Programming,when%20employed%20on%20its%20own.

- https://www.geeksforgeeks.org/cross-validation-in-r-programming/

- https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f#:~:text=This%20significantly%20reduces%20bias%20as,being%20used%20in%20validation%20set.