Exploratory Data Analysis and Preliminary Statistical Modeling on Vehicle Sensor Data

Nyasha M

26/04/2021

This is an R Markdown file which conducts an exploratory analysis of various .csv files from vehicle sensor data (e.g., car movement speed, internal car fluid temperature).

I was supplied with 100 .csv files and was tasked with proposing an appropriate analysis plan/machine learning or statistical (regression) model for analysing the data. Each .csv file represented one operating block for the vehicle.

I chose to build a function to quicken the import of .csv files and later, decided to concatenate all 100 files into one (low MB size, thus plausible), due to a large percentage of missing data (80-90% missing) in each of the individual .csv files.

Loading Files/ Importing Data

Load a data dictionary I obtained for figuring out what each of the columns stand for, in each .csv file. Confirm that there are 100 .csv files to work with in the directory where I stored the files.

```
js<-fromJSON(txt="./DataScienceTestFiles/column_dict.json", flatten = TRUE)

files_csv<-list.files(path = "./DataScienceTestFiles", pattern = c("^O","\\.csv$"))
length(grep("^O", files_csv))
```

```
## [1] 100
```

Create a function to try reading in a .csv file. Upon previewing the data, I noticed that we were dealing with a UNIX format for the time column of the dataset (`X`). I converted this column into a 'YYYY-MM-DD' format and previewed the converted sample .csv file.

```
load_data<-function(num){
  x<-read.csv(paste("./DataScienceTestFiles/",files_csv[num+1],sep = ""), header = TRUE)
}
df<-load_data(0)

newX<-as.POSIXct(df$X, origin="1970-01-01")
df$X<-newX
head(df)
```

```
##                       X X1      X2    X3   X4 X5 X6 X7   X8 X9 X10 X11 X12 X13
## 1 2018-05-21 08:29:41 NA      NA    NA   NA NA NA NA   NA NA  NA  NA  NA  NA
## 2 2018-05-21 08:29:42 NA      NA    NA   NA NA NA NA 6.15 NA  NA  NA  NA  NA
## 3 2018-05-21 08:29:43  0 694.125 292.3 72.7 NA NA  0   NA NA  NA  NA  NA  NA
```

```
## 4 2018-05-21 08:29:45  0 694.875    NA   NA NA NA NA   NA NA   NA   NA   NA   NA
## 5 2018-05-21 08:29:46 NA      NA    NA   NA NA NA NA   NA NA   NA   NA   NA   NA
## 6 2018-05-21 08:29:47 NA      NA    NA   NA NA NA  0 6.15 NA   NA   NA   NA   NA
##   X14 X15 X16 X17 X18 X19 X20 X21 X22 X23 X24 X25 X26 X27
## 1  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
## 2  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
## 3  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
## 4   0 504  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
## 5  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
## 6  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA   0  NA  NA
```

Now I am getting a closer look at the data.

```
print(diagnose(df))
```

```
## # A tibble: 28 x 6
##    variables types   missing_count missing_percent unique_count unique_rate
##    <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
##  1 X         POSIXct             0             0          10196    1
##  2 X1        numeric          6673            65.4          102    0.0100
##  3 X2        numeric          6673            65.4         1339    0.131
##  4 X3        numeric          8106            79.5         1229    0.121
##  5 X4        numeric          8106            79.5         1072    0.105
##  6 X5        numeric          9184            90.1          146    0.0143
##  7 X6        numeric          9184            90.1           80    0.00785
##  8 X7        numeric          8216            80.6          136    0.0133
##  9 X8        numeric          9274            91.0          200    0.0196
## 10 X9        numeric         10088            98.9            4    0.000392
## # ... with 18 more rows
```

```
print(object.size(df), standard="auto", units="Mb")
```

```
## 2.1 Mb
```

I noticed that my one sample contained a lot of missing data. The file size of the .csv was also fairly small. I looked at a few more .csv using my above loading function and noticed that many of the other .csv files in the folder were also like this.

I decided to do a loop to concatenate all the .csv files together, and improve the sample size of my overall data.

```
Main_df<-load_data(0)
for (i in files_csv[2:length(files_csv)]){
  x<-read.csv(paste("./DataScienceTestFiles/",i,sep = ""), header = TRUE)
  Main_df<-rbind(Main_df,x)
}
print(object.size(Main_df), standard="auto", units="Mb")
```

```
## 155.6 Mb
```

```
newX<-as.POSIXct(Main_df$X, origin="1970-01-01")
Main_df$X<-newX
```

Now obtaining a preview of my concatenated dataset to see if all the variables were parameterized and appeared to have been loaded in correctly.

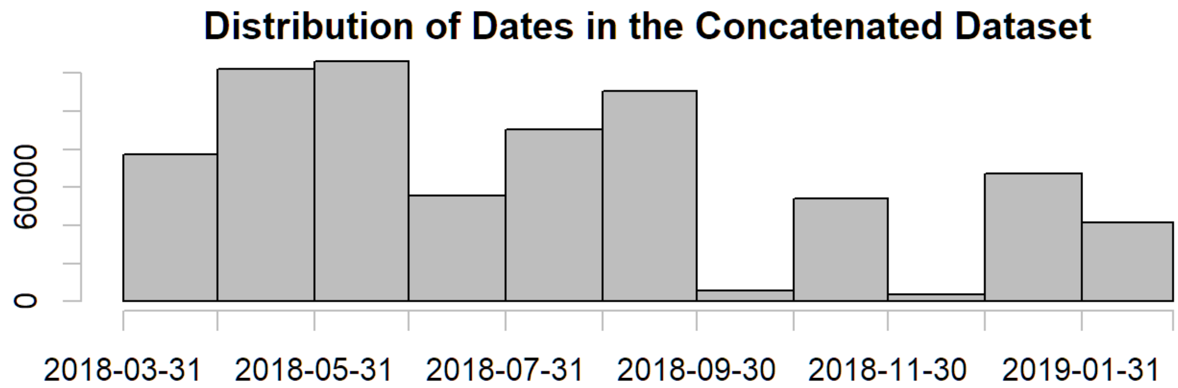
```
print(diagnose(Main_df), n=Inf)
```

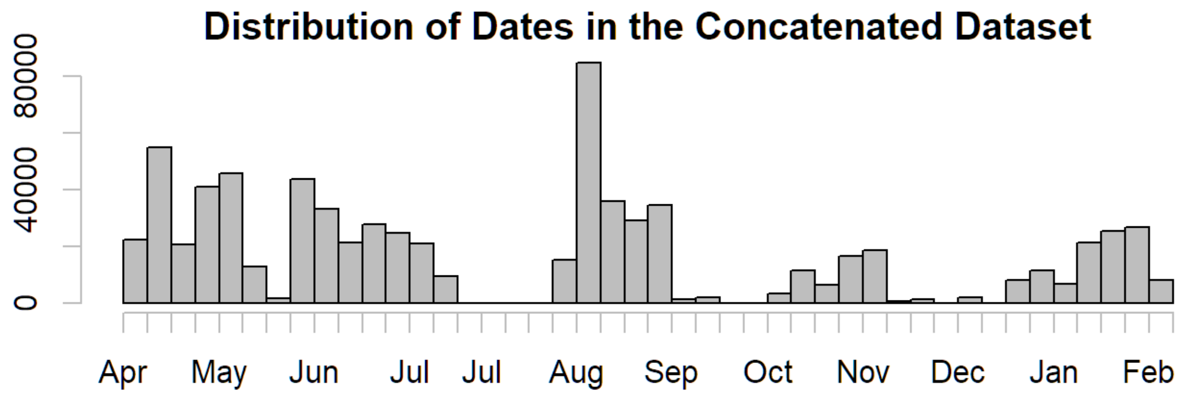
```
## # A tibble: 28 x 6
##    variables types   missing_count missing_percent unique_count unique_rate
##    <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
##  1 X         POSIXct             0             0         755550  1
##  2 X1        numeric        467540            61.9          102  0.000135
##  3 X2        numeric        467540            61.9         5042  0.00667
##  4 X3        numeric        461742            61.1         5479  0.00725
##  5 X4        numeric        461742            61.1         2880  0.00381
##  6 X5        numeric        538881            71.3          227  0.000300
##  7 X6        numeric        538881            71.3          122  0.000161
##  8 X7        numeric        465689            61.6         1390  0.00184
##  9 X8        numeric        541641            71.7          766  0.00101
## 10 X9        numeric        735120            97.3           53  0.0000701
## 11 X10       numeric        734551            97.2        11116  0.0147
## 12 X11       logical        755550           100              1  0.00000132
## 13 X12       numeric        734819            97.3          119  0.000158
## 14 X13       numeric        734819            97.3          111  0.000147
## 15 X14       numeric        712396            94.3            8  0.0000106
## 16 X15       numeric        712396            94.3          111  0.000147
## 17 X16       numeric        666191            88.2         8872  0.0117
## 18 X17       numeric        533714            70.6        11832  0.0157
## 19 X18       numeric        533714            70.6          150  0.000199
## 20 X19       numeric        712874            94.4         3616  0.00479
## 21 X20       numeric        536098            71.0        10353  0.0137
## 22 X21       numeric        536098            71.0        11827  0.0157
## 23 X22       numeric        716232            94.8          159  0.000210
## 24 X23       numeric        715000            94.6         3123  0.00413
## 25 X24       numeric        715000            94.6          142  0.000188
## 26 X25       numeric             0             0              4  0.00000529
## 27 X26       numeric        712284            94.3         3467  0.00459
## 28 X27       numeric        712284            94.3         3553  0.00470
```

What is the time/date range of the concatenated dataset?

```
par(mfrow=c(2,1),mai = c(0.6, 0.5, 0.3, 0.1))

hist(Main_df$X, breaks="months", freq = TRUE, col="grey",
     main = "Distribution of Dates in the Concatenated Dataset")
hist(Main_df$X, breaks="weeks", freq = TRUE, col="grey",
     main = "Distribution of Dates in the Concatenated Dataset")
```






```
range(Main_df$X)
```

```
## [1] "2018-04-19 08:36:03 EDT" "2019-02-13 07:17:36 EST"
```

Begin Actual Analysis of Data

How much missing data is there after combining all 100 .csv operating blocks into one?

```
diag_df<-diagnose(Main_df)
diag_df
```

```
## # A tibble: 28 x 6
##    variables types   missing_count missing_percent unique_count unique_rate
##    <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
##  1 X         POSIXct             0             0         755550   1
##  2 X1        numeric        467540            61.9          102   0.000135
##  3 X2        numeric        467540            61.9         5042   0.00667
##  4 X3        numeric        461742            61.1         5479   0.00725
##  5 X4        numeric        461742            61.1         2880   0.00381
##  6 X5        numeric        538881            71.3          227   0.000300
##  7 X6        numeric        538881            71.3          122   0.000161
##  8 X7        numeric        465689            61.6         1390   0.00184
##  9 X8        numeric        541641            71.7          766   0.00101
## 10 X9        numeric        735120            97.3           53   0.0000701
## # ... with 18 more rows
```

There's still a lot of missing data. Let's choose to remove variables with >90% missing data, as suggested by the results of the simulation study by Madley-Dowd et al (2019). Multiple imputation (MI) still performed well when 90% or less of the data was missing.

```
low_missing<-diag_df$variables[diag_df$missing_percent<=90]

Reduced_df<-Main_df[low_missing]
diagnose(Reduced_df)
```

```
## # A tibble: 15 x 6
##    variables types   missing_count missing_percent unique_count unique_rate
##    <chr>     <chr>           <int>           <dbl>        <int>       <dbl>
##  1 X         POSIXct             0             0         755550  1
##  2 X1        numeric        467540            61.9          102  0.000135
##  3 X2        numeric        467540            61.9         5042  0.00667
##  4 X3        numeric        461742            61.1         5479  0.00725
##  5 X4        numeric        461742            61.1         2880  0.00381
##  6 X5        numeric        538881            71.3          227  0.000300
##  7 X6        numeric        538881            71.3          122  0.000161
##  8 X7        numeric        465689            61.6         1390  0.00184
##  9 X8        numeric        541641            71.7          766  0.00101
## 10 X16       numeric        666191            88.2         8872  0.0117
## 11 X17       numeric        533714            70.6        11832  0.0157
## 12 X18       numeric        533714            70.6          150  0.000199
## 13 X20       numeric        536098            71.0        10353  0.0137
## 14 X21       numeric        536098            71.0        11827  0.0157
## 15 X25       numeric             0             0              4  0.00000529
```

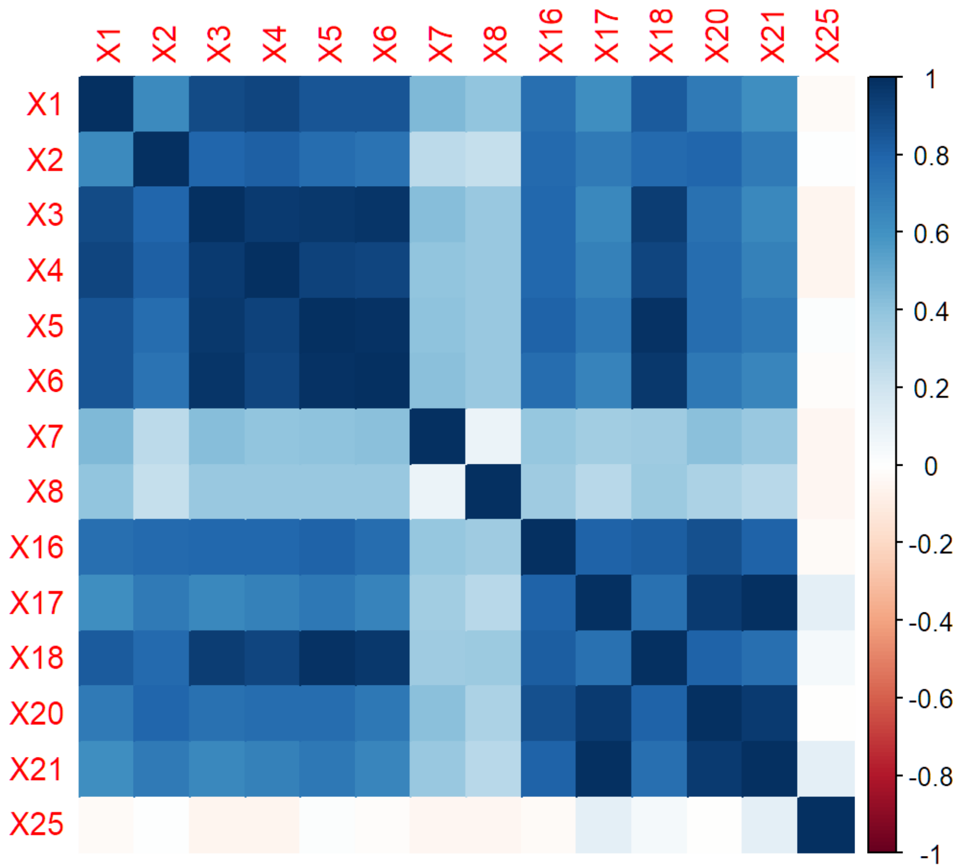
Much better.

Note, the data seems to be all continuous. However, there's too much data to make a scatterplot matrix across everything right now. Let's make a correlation matrix for some quick visualizations.

We have to use "pairwise complete observations" or something similar for the corrplot package or it won't work.

```
cor.df<-cor(Reduced_df[2:15], use="pairwise.complete.obs")
sig.df<-cor.mtest(Reduced_df[2:15])

# Make a corrplot--originally had it show p-values but font too small to see.
corrplot(cor.df, method="color")
```



There's relatively high correlations among the variables.

At this point, we need to start examining some sort of objective or test what sort of model we might like to use. Vehicle speed (variable X16) seems like it might be an interesting outcome to work with. X16 is associated with everything BUT X7, X8, X25.

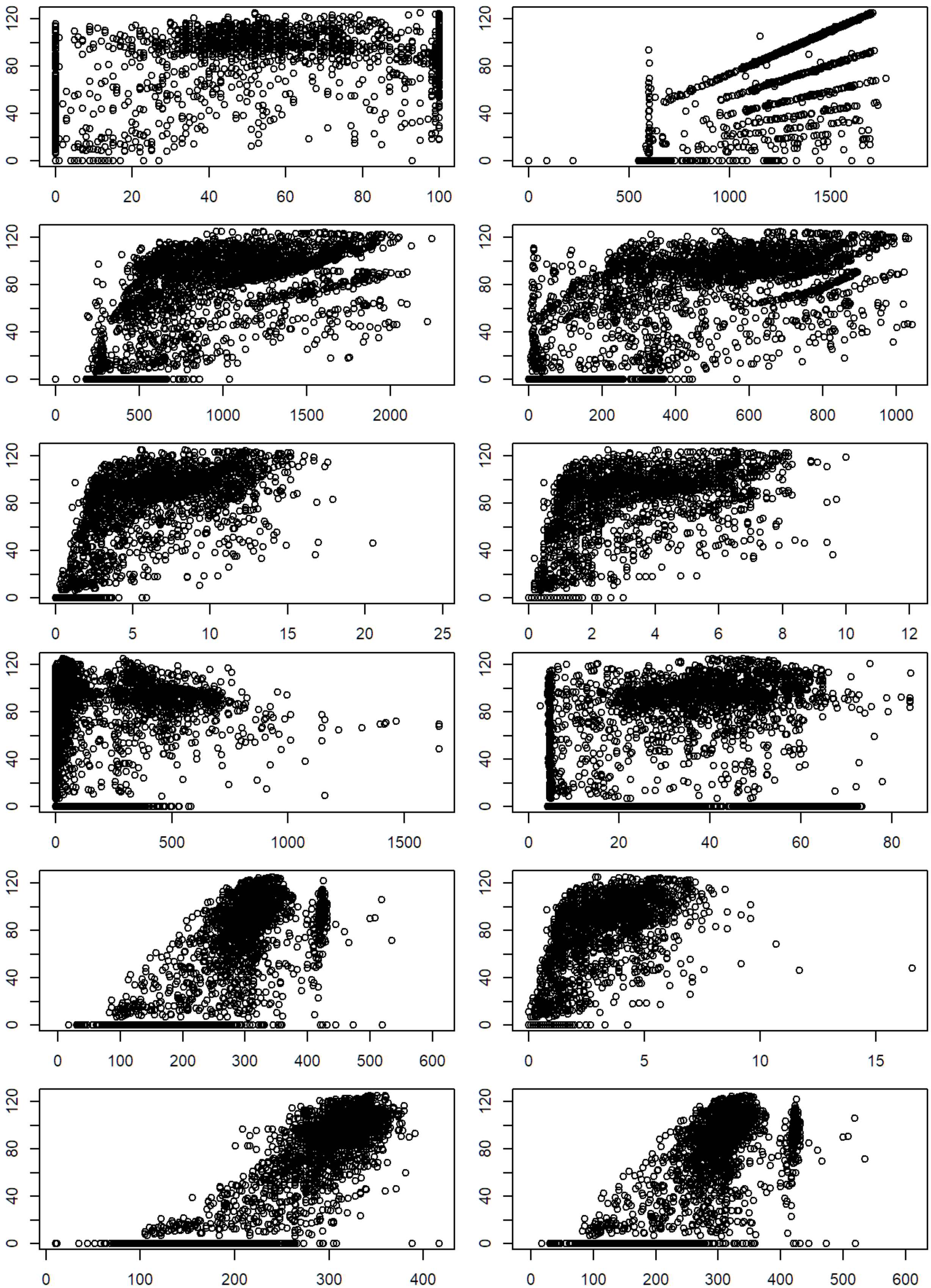
Removing the date variable first to make it easier to generate a model.

```
Reduced_df<-Reduced_df[2:ncol(Reduced_df)]
```

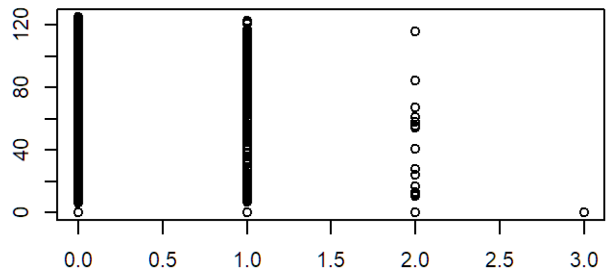
Making another scatterplot matrix, this time on a reduced sample of the data with vehicle speed as an outcome to see what type of model might be appropriate to use if vehicle speed was set as our outcome.

```
set.seed(3)
smaller<-Reduced_df[sample(nrow(Reduced_df),nrow(Reduced_df)*.1),  ]

par(mfrow=c(3,2), mai = c(0.3, 0.3, 0.1, 0.1))
plot(smaller$X16~., data=smaller, ylab = "Vehicle speed (km/h)")
```



0 40 80 120
0 20 40 60 80 100
0 40 80 120
0 500 1000 1500
0 40 80 120
0 500 1000 1500 2000
0 40 80 120
0 200 400 600 800 1000
0 40 80 120
0 5 10 15 20 25
0 40 80 120
0 2 4 6 8 10 12
0 40 80 120
0 500 1000 1500
0 40 80 120
0 20 40 60 80
0 40 80 120
0 100 200 300 400 500 600
0 40 80 120
0 5 10 15
0 40 80 120
0 100 200 300 400
0 40 80 120
0 100 200 300 400 500 600



Multiple linear regression might not be the best, but let's try try it for now and see what happens (or what the diagnostic plot post-modeling might suggest. This is just a test model).

```
Reduced_df.lm<-lm(Reduced_df$X16~., Reduced_df)
summary(Reduced_df.lm)
```

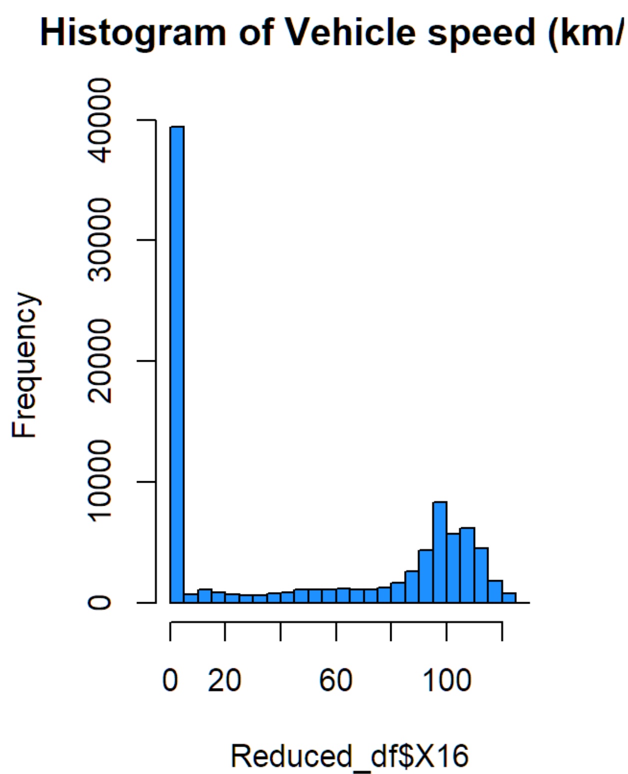
```
##
## Call:
## lm(formula = Reduced_df$X16 ~ ., data = Reduced_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -116.73  -12.48    3.17   12.72   76.03
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.053e+01  8.320e-01 -60.735  < 2e-16 ***
## X1           2.584e-01  1.353e-02  19.103  < 2e-16 ***
## X2           7.850e-03  1.157e-03   6.782 1.23e-11 ***
## X3           3.718e-02  2.727e-03  13.635  < 2e-16 ***
## X4          -4.977e-02  2.574e-03 -19.340  < 2e-16 ***
## X5          -1.196e+01  3.043e+00  -3.929 8.56e-05 ***
## X6           6.375e+00  3.078e+00   2.071   0.0383 *
## X7          -8.724e-04  8.663e-04  -1.007   0.3139
## X8           8.620e-02  8.902e-03   9.683  < 2e-16 ***
## X17          2.329e+00  9.177e+00   0.254   0.7997
```

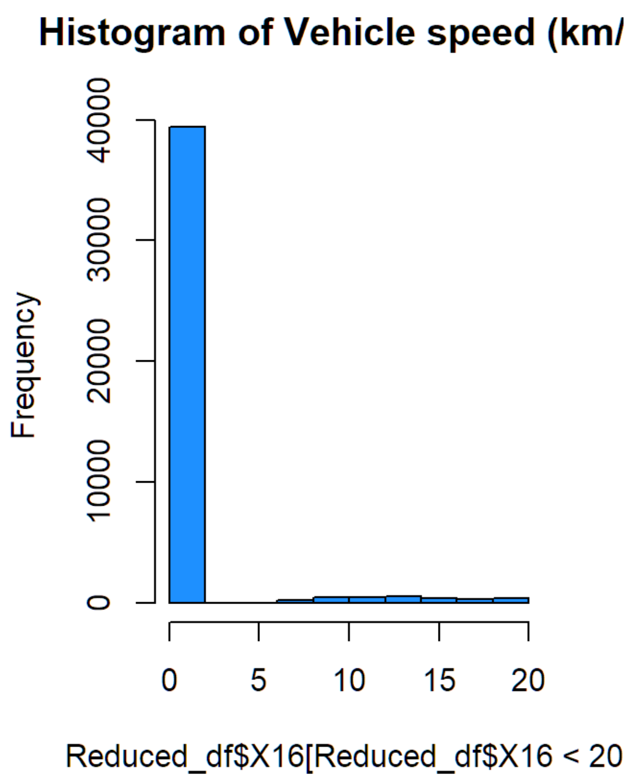
```
## X18          1.957e+01  3.072e+00   6.370 1.94e-10 ***
## X20          3.345e-01  9.050e-03  36.962  < 2e-16 ***
## X21         -2.377e+00  9.177e+00  -0.259   0.7956
## X25         -2.155e+00  4.162e-01  -5.177 2.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.05 on 14525 degrees of freedom
##   (741011 observations deleted due to missingness)
## Multiple R-squared:  0.829,  Adjusted R-squared:  0.8289
## F-statistic:  5417 on 13 and 14525 DF,  p-value: < 2.2e-16
```

Output here has a lot of exponents in many of the numbers. Let's look at the distribution of our outcome variable to start (should have done this at the beginning).

```
par(mfrow=c(1,2))

hist(Reduced_df$X16, breaks = 40, col="dodgerblue1",
     main = "Histogram of Vehicle speed (km/h)")
hist(Reduced_df$X16[Reduced_df$X16<20], breaks=10, col="dodgerblue1",
     main = "Histogram of Vehicle speed (km/h)")
```

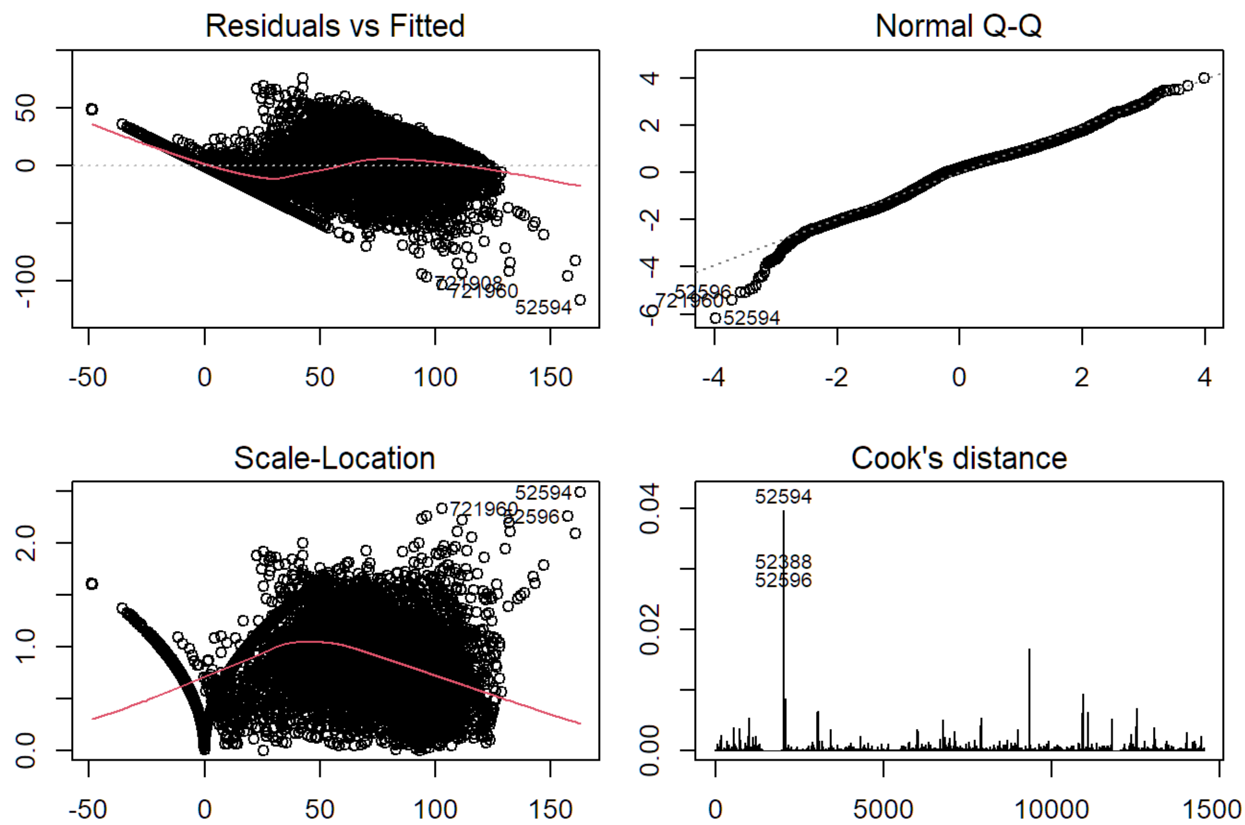




We have an extremely high frequency of 0 km/h. but this is also a pretty "natural"/normal speed for a vehicle and thus the high frequently of this value is not necessarily an error/outlier–reasonable that a vehicle would be at 0 km/h often.

Perhaps we might need to try some sort of dichotomous model? But let's look at the model diagnostics first to get more information.

```
par(mfrow=c(2,2), mai = c(0.5, 0.4, 0.3, 0.1))
plot(Reduced_df.lm, which=1:4)
```



The residuals vs fitted and Normal Q-Q plots look okay. Scale-location suggests that multiple linear regression might not be best, or that we might want to try transforming our outcome variable. Cook's distance suggests some outliers to look into.

Next steps?

- Possibly try transforming the outcome variable, dichotomizing it (risky), some sort of polynomial model, or a support vector machine model.
- Check for multicollinearity if using a traditional statistical model.
- Use stepwise variable selection to remove/add variables.
- Conduct likelihood ratio (LR) test between every removal/addition of a variable if using a traditional statistical model (nested)