Regression - Linear Regression

Hiep Nguyen

02/14/2023

How does linear regression work?

Linear regression tries to find a line of best fit within the data. It's important to note that the line doesn't have to be straight, as polynomial linear regression exists and can be used to fit a line on any degree. Linear regression is great when it comes to linear data, and it can be easily adjusted to avoid overfitting the data. However, it's not so good when dealing with non-linear data and it's only used for qualitative data. It's also common to see linear regression underfit the data due to its high bias and low variance approach.

Data Cleaning

Looking at the original dataset, I've decided to do some basic cleaning that won't affect the training or testing. I see that the column Vehicle Class has some duplicate levels, so I've consolodated those into one. I'm also seeing some values, e.g. fuel and transmission, that could be factorized.

```
df <- read.csv("Fuel_Consumption_2000-2022.csv")

str(df)
```

```
## 'data.frame':    22556 obs. of  13 variables:
##  $ YEAR            : int  2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ MAKE            : chr  "ACURA" "ACURA" "ACURA" "ACURA" ...
##  $ MODEL           : chr  "1.6EL" "1.6EL" "3.2TL" "3.5RL" ...
##  $ VEHICLE.CLASS   : chr  "COMPACT" "COMPACT" "MID-SIZE" "MID-SIZE" ...
##  $ ENGINE.SIZE     : num  1.6 1.6 3.2 3.5 1.8 1.8 1.8 3 3.2 1.8 ...
##  $ CYLINDERS       : int  4 4 6 6 4 4 4 6 6 4 ...
##  $ TRANSMISSION    : chr  "A4" "M5" "AS5" "A4" ...
##  $ FUEL            : chr  "X" "X" "Z" "Z" ...
##  $ FUEL.CONSUMPTION: num  9.2 8.5 12.2 13.4 10 9.3 9.4 13.6 13.8 11.4 ...
##  $ HWY..L.100.km.  : num  6.7 6.5 7.4 9.2 7 6.8 7 9.2 9.1 7.2 ...
##  $ COMB..L.100.km. : num  8.1 7.6 10 11.5 8.6 8.2 8.3 11.6 11.7 9.5 ...
##  $ COMB..mpg.      : int  35 37 28 25 33 34 34 24 24 30 ...
##  $ EMISSIONS       : int  186 175 230 264 198 189 191 267 269 218 ...
```

```
head(df)
```

```
##   YEAR  MAKE   MODEL VEHICLE.CLASS ENGINE.SIZE CYLINDERS TRANSMISSION FUEL
## 1 2000 ACURA   1.6EL       COMPACT         1.6         4           A4    X
## 2 2000 ACURA   1.6EL       COMPACT         1.6         4           M5    X
## 3 2000 ACURA   3.2TL      MID-SIZE         3.2         6          AS5    Z
## 4 2000 ACURA   3.5RL      MID-SIZE         3.5         6           A4    Z
```

```
## 5 2000 ACURA INTEGRA    SUBCOMPACT         1.8         4           A4    X
## 6 2000 ACURA INTEGRA    SUBCOMPACT         1.8         4           M5    X
##   FUEL.CONSUMPTION HWY..L.100.km. COMB..L.100.km. COMB..mpg. EMISSIONS
## 1              9.2            6.7             8.1         35       186
## 2              8.5            6.5             7.6         37       175
## 3             12.2            7.4            10.0         28       230
## 4             13.4            9.2            11.5         25       264
## 5             10.0            7.0             8.6         33       198
## 6              9.3            6.8             8.2         34       189
```

```
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "COMPACT"] <- "Compact"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "FULL-SIZE"] <- "Full-size"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "MID-SIZE"] <- "Mid-size"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "MINICOMPACT"] <- "Minicompact"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "MINIVAN"] <- "Minivan"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "PICKUP TRUCK - SMALL"] <- "Pickup truck: Small"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "PICKUP TRUCK - STANDARD"] <- "Pickup truck: Standard"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "SPECIAL PURPOSE VEHICLE"] <- "Special purpose vehicle"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "STATION WAGON - MID-SIZE"] <- "Station wagon: Mid-size"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "STATION WAGON - SMALL"] <- "Station wagon: Small"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "SUBCOMPACT"] <- "Subcompact"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "SUV - SMALL"] <- "SUV: Small"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "SUV - STANDARD"] <- "SUV: Standard"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "TWO-SEATER"] <- "Two-seater"
df["VEHICLE.CLASS"][df["VEHICLE.CLASS"] == "VAN - PASSENGER"] <- "Van: Passenger"

df$FUEL <- as.factor(df$FUEL)
df$TRANSMISSION <- as.factor(df$TRANSMISSION)
df$MAKE <- as.factor(df$MAKE)
df$VEHICLE.CLASS <- as.factor(df$VEHICLE.CLASS)
```

80/20 Train/Test

Now that our data has been cleaned up, I will do an 80/20 train/test split on the data frame.

```
set.seed(123)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=TRUE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

For this data, we're going to have the target variable be Fuel Consumption. Our independent variables will be Engine Size, Cylinders, Transmission, and Fuel type. The other variables depend on either these variables or fuel consumption, so we're not going to be using those. Looking into transmission, we can see that some of the levels don't have many samples. This will be good to keep in mind when creating the linear model.

```
str(train)
```

```
## 'data.frame':    18044 obs. of  13 variables:
##  $ YEAR            : int  2019 2019 2004 2002 2004 2012 2005 2007 2016 2003 ...
```

```
##  $ MAKE            : Factor w/ 87 levels "Acura","ACURA",..: 18 23 11 64 43 38 87 85 57 81 ...
##  $ MODEL           : chr  "Silverado 4WD FFV" "Charger FFV" "Z4 ROADSTER" "FRONTIER V6 #" ...
##  $ VEHICLE.CLASS   : Factor w/ 17 levels "Compact","Full-size",..: 7 2 15 7 12 11 1 11 13 10 ...
##  $ ENGINE.SIZE     : num  5.3 3.6 3 3.3 4 3.7 2.4 3.2 2 2 ...
##  $ CYLINDERS       : int  8 6 6 6 6 6 5 6 4 4 ...
##  $ TRANSMISSION    : Factor w/ 30 levels "A10","A3","A4",..: 5 4 29 3 3 29 16 17 17 3 ...
##  $ FUEL            : Factor w/ 5 levels "D","E","N","X",..: 4 4 5 5 4 5 5 5 4 4 ...
##  $ FUEL.CONSUMPTION: num  16.1 13.7 11.4 15.5 14.7 12.3 10.8 10.8 9.3 9.1 ...
##  $ HWY..L.100.km.  : num  12.5 9 7.4 12.2 10.8 7.9 7.3 7.5 7.6 7 ...
##  $ COMB..L.100.km. : num  14.4 11.6 9.6 14 12.9 10.3 9.3 9.3 8.5 8.2 ...
##  $ COMB..mpg.      : int  20 24 29 20 22 27 30 30 33 34 ...
##  $ EMISSIONS       : int  339 271 221 322 297 237 214 214 199 189 ...
```

```
head(train)
```

```
##       YEAR      MAKE             MODEL          VEHICLE.CLASS ENGINE.SIZE
## 18847 2019 Chevrolet Silverado 4WD FFV Pickup truck: Standard         5.3
## 18895 2019     Dodge       Charger FFV              Full-size         3.6
## 2986  2004       BMW       Z4 ROADSTER             Two-seater         3.0
## 1842  2002    NISSAN     FRONTIER V6 # Pickup truck: Standard         3.3
## 3371  2004      JEEP            TJ 4X4                   SUV         4.0
## 11638 2012  INFINITI         G37 COUPE             Subcompact         3.7
##       CYLINDERS TRANSMISSION FUEL FUEL.CONSUMPTION HWY..L.100.km.
## 18847         8           A6    X             16.1           12.5
## 18895         6           A5    X             13.7            9.0
## 2986          6           M6    Z             11.4            7.4
## 1842          6           A4    Z             15.5           12.2
## 3371          6           A4    X             14.7           10.8
## 11638         6           M6    Z             12.3            7.9
##       COMB..L.100.km. COMB..mpg. EMISSIONS
## 18847            14.4         20       339
## 18895            11.6         24       271
## 2986              9.6         29       221
## 1842             14.0         20       322
## 3371             12.9         22       297
## 11638            10.3         27       237
```

```
colSums(is.na(train))
```

```
##             YEAR             MAKE            MODEL    VEHICLE.CLASS
##                0                0                0                0
##      ENGINE.SIZE        CYLINDERS     TRANSMISSION             FUEL
##                0                0                0                0
## FUEL.CONSUMPTION   HWY..L.100.km.  COMB..L.100.km.       COMB..mpg.
##                0                0                0                0
##        EMISSIONS
##                0
```

```
print("Vehicle Class Levels:")
```

```
## [1] "Vehicle Class Levels:"
```

```
summary(train$VEHICLE.CLASS)
```

```
##                 Compact               Full-size                Mid-size
##                    2529                    1266                    2358
##             Minicompact                 Minivan     Pickup truck: Small
##                     762                     306                     389
##  Pickup truck: Standard Special purpose vehicle Station wagon: Mid-size
##                    1735                      90                     319
##    Station wagon: Small              Subcompact                     SUV
##                     706                    1566                    2180
##              SUV: Small           SUV: Standard              Two-seater
##                    1418                     886                    1061
##             VAN - CARGO          Van: Passenger
##                     260                     213
```

```
print("Transmission Summary:")
```

```
## [1] "Transmission Summary:"
```

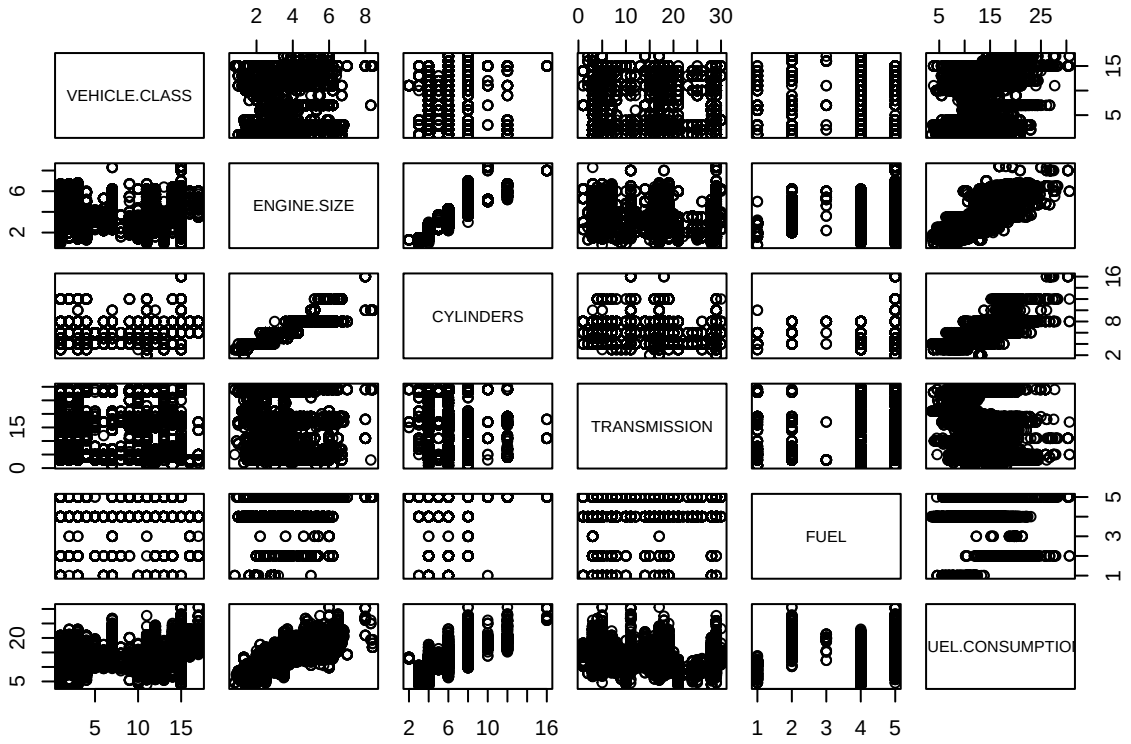
```
summary(train$TRANSMISSION)
```

```
##  A10   A3   A4   A5   A6   A7   A8   A9  AM5  AM6  AM7  AM8  AM9 AS10  AS4  AS5
##  126   18 2774 1103 1569  216  623  434    6  170  595  122    4  277  214  660
##  AS6  AS7  AS8  AS9   AV  AV1 AV10  AV6  AV7  AV8   M4   M5   M6   M7
## 2362  509 1418   87  602    5   16  101  127   85    0 1691 2021  109
```

Graphing

We're first going to graph the pairs of our independent variables with the fuel consumption to get a good idea of which variables may be helpful. I can already see that Engine Size and Cylinders may have a correlation with fuel consumption. Fuel type could also be a useful variable when it comes to multivariate testing.

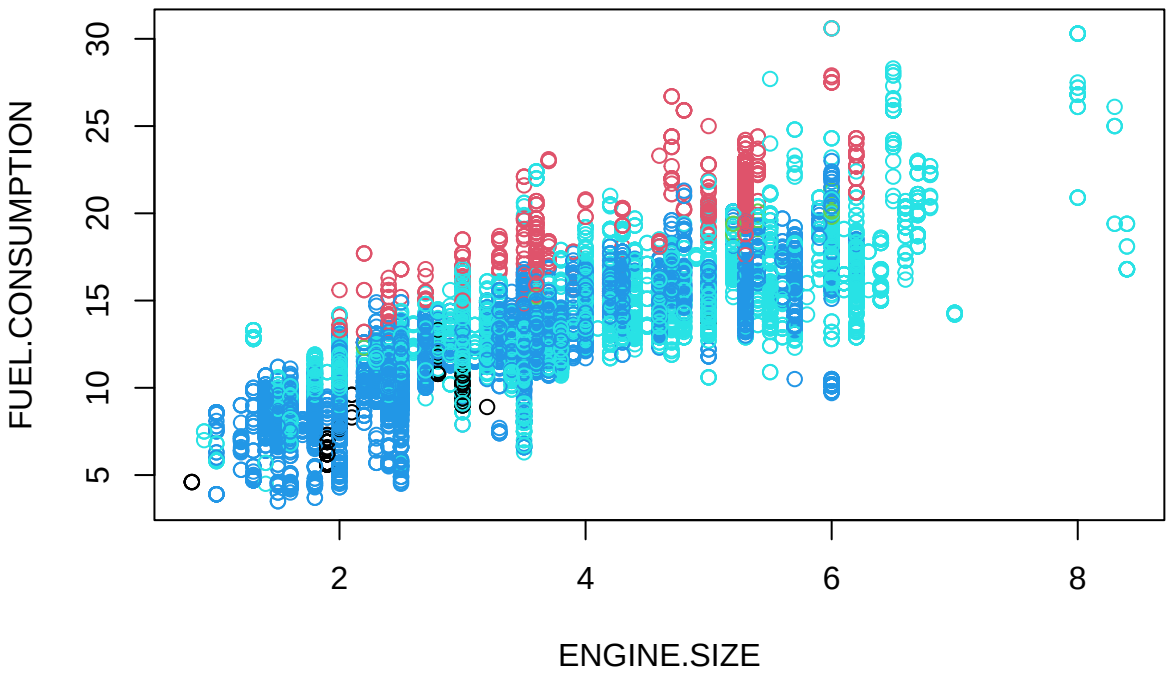
```
pairs(train[4:9])
```



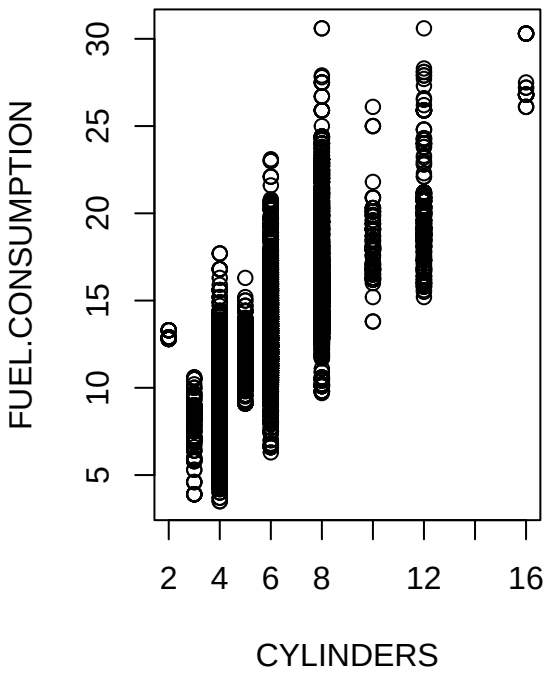
As we can see from the data shown below, there seems to be some correlation on the graphs of engine size, fuel type, and cylinders. I've color coded the engine size chart based on the fuel type, and there does seem to be some noticeable grouping between the colors. This could be useful when creating the multivariate models. Right now, it looks like engine size is the most promising to me, so I'll be using that for the univariate model.

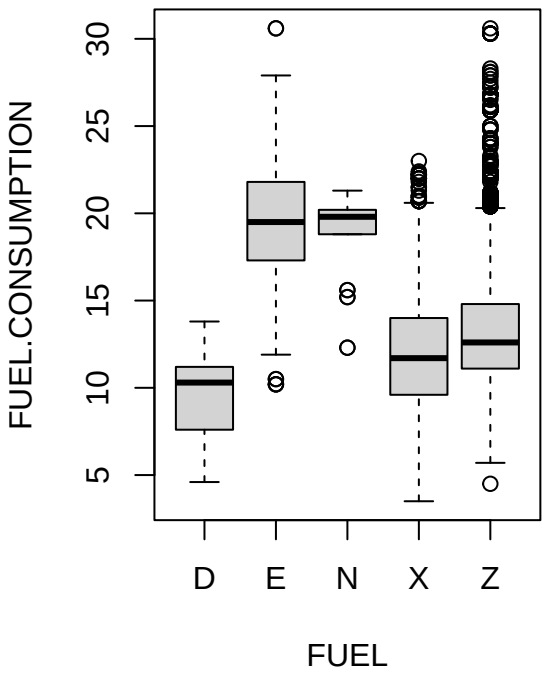
I won't be using the make as some of the makes have a very small number of observations. Not to mention, a single Make can have very different types of cars.

```
par(mfrow=c(1,1))
plot(FUEL.CONSUMPTION~ENGINE.SIZE, data=train, col=train$FUEL)
```




```
par(mfrow=c(1,2))
plot(FUEL.CONSUMPTION~CYLINDERS, data=train)
boxplot(FUEL.CONSUMPTION~FUEL, data=train)
```





Simple Linear Regression Model

I have created a simple linear regression model for fuel consumption as a function of engine size. The residuals seem to maintain a bit of symmetry, and the difference between the minimum of -8.7 and the maximum of 12.15 is wide, but it's narrow enough for our data to still work.

In the coefficient section, we can see estimates and p-values for the engine-size and the intercept. The engine size has an extremely low p-value along with three stars, meaning that our predictor does correlate with the fuel consumption. We have more evidence of this, as our t-value is extremely high. Using the engine size coefficient, for every unit in which our engine size increases, the fuel consumption will increase by 2.15. Combining this with the intercept of 5.51, we can create a linear equation:

FuelConsumption = 2.15*(EngineSize) + 5.51

Our residual standard error of 1.97 indicates that this model will have an average error of 2 gallons. We have 18042 degrees of freedom because we had 18044 observations minus the two predictors. Our R-squared value of 0.6813 shows that 68% of our variance can be explained by engine size. Our F-statistic is extremely far from 0 and our p-value is rather small too, which gives us evidence to reject the null hypothesis.

```
lm1 = lm(FUEL.CONSUMPTION~ENGINE.SIZE, data=train)
summary(lm1)
```

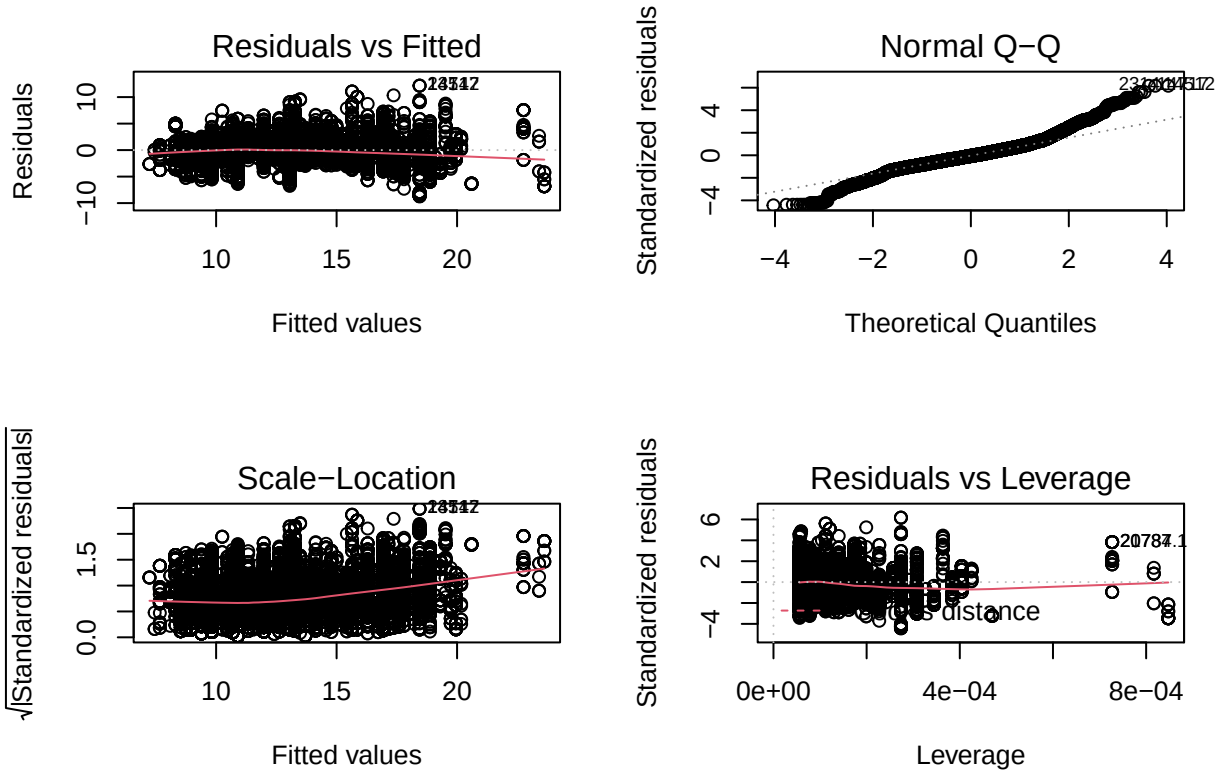
```
##
## Call:
## lm(formula = FUEL.CONSUMPTION ~ ENGINE.SIZE, data = train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7445 -1.1209 -0.1203  1.0011 12.1555
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.50914    0.03958   139.2   <2e-16 ***
## ENGINE.SIZE  2.15589    0.01098   196.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.97 on 18042 degrees of freedom
## Multiple R-squared:  0.6813, Adjusted R-squared:  0.6813
## F-statistic: 3.857e+04 on 1 and 18042 DF,  p-value: < 2.2e-16
```

Plotting the residuals

Looking at the first graph, we can see the line showing a slight downward curve. This could indicate that there's a possible non-linear relationship with the data, but I dont believe that's the case, as the curve is very slight and could be due to an unaccounted variable. The second graph shows whether the residuals are normally distributed or not if it follows the line. Some of our more extreme residuals deviate from the line significantly, so it may not follow a normal distribution. In the third graph, we want to see the points distributed equally around the line and be horizontal. We can see the line becoming more vertical as it goes, which means that the residuals spread wider as x increases. In the fourth graph, we don't see Cook's line (dashed), so we're probably well inside of it. Therefore, there aren't any influential cases that we need to worry about.

```
par(mfrow=c(2,2))
plot(lm1)
```



Multiple Linear Regression Model

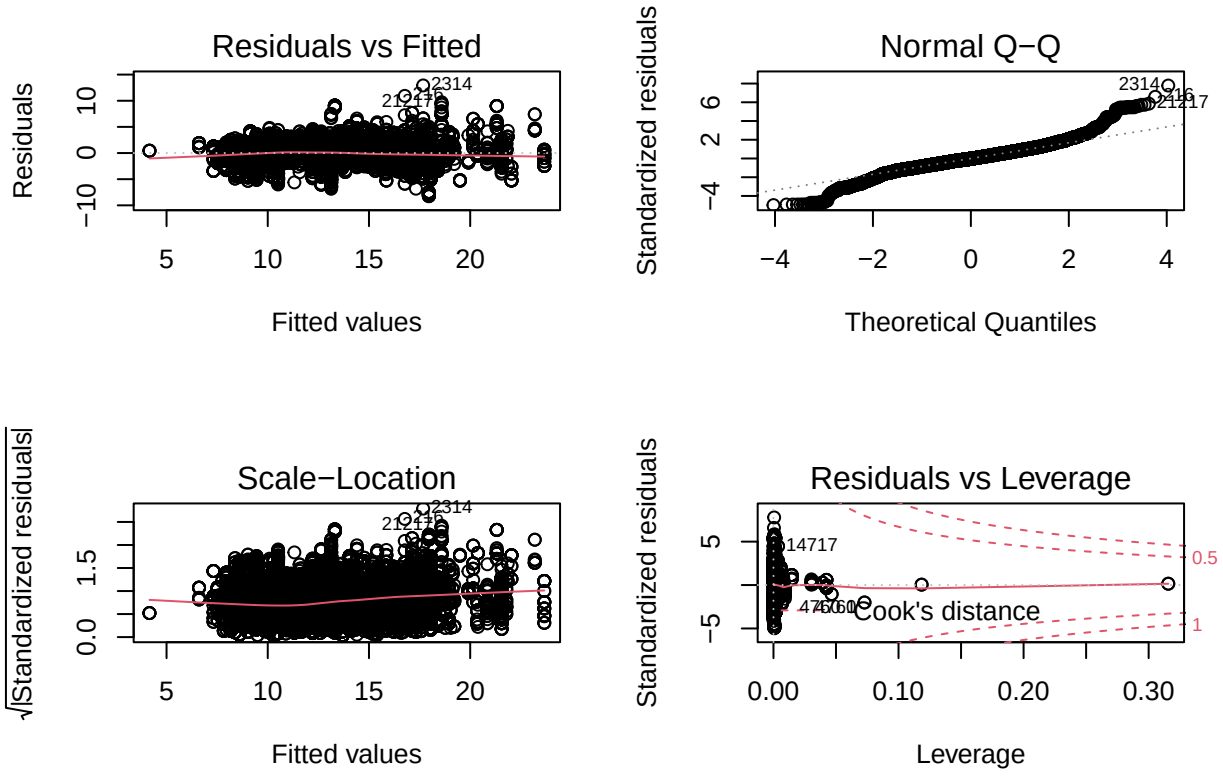
For the first multiple linear regression model, I decided to use the variables Engine Size and Fuel along with an interaction effect between the two. The interaction effect seemed to only marginally increase the performance, possibly insinuating that there's no synergy between them. The residual graphs have become much more noticeably stable.

```
par(mfrow=c(2,2))
lm2 = lm(FUEL.CONSUMPTION~ENGINE.SIZE + FUEL + ENGINE.SIZE*FUEL, data=train)
summary(lm2)
```

```
##
## Call:
## lm(formula = FUEL.CONSUMPTION ~ ENGINE.SIZE + FUEL + ENGINE.SIZE *
##     FUEL, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2582 -0.9512 -0.0329  0.9360 12.9239
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.7082     0.4649   3.674 0.000239 ***
## ENGINE.SIZE         3.0608     0.1773  17.261  < 2e-16 ***
## FUELE               7.4354     0.5199  14.302  < 2e-16 ***
```

```
## FUELN                5.6052     1.6028   3.497 0.000471 ***
## FUELX                3.4825     0.4672   7.453 9.51e-14 ***
## FUELZ                5.0627     0.4678  10.823  < 2e-16 ***
## ENGINE.SIZE:FUELE   -0.7196     0.1846  -3.898 9.72e-05 ***
## ENGINE.SIZE:FUELN   -0.8900     0.3327  -2.675 0.007475 **
## ENGINE.SIZE:FUELX   -0.9329     0.1779  -5.245 1.58e-07 ***
## ENGINE.SIZE:FUELZ   -1.2432     0.1778  -6.991 2.83e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.661 on 18034 degrees of freedom
## Multiple R-squared:  0.7737, Adjusted R-squared:  0.7736
## F-statistic:  6852 on 9 and 18034 DF,  p-value: < 2.2e-16
```

```
plot(lm2)
```



Second Multiple Linear Regression Model

For the next multiple linear regression model, I decided to go all out and use four variables. Just like the previous graph, the residual graphs look much more stable.

The first model seems to have the highest residual standard error, which goes down as we go through the second, and the third model having the least of the three. A similar trend can be seen in the R-squared values, where the first is the lowest and the third is the highest. Two of the predictors in the third model (Full-size and Mid-size) seem to have very high p-values, which indicate no correlation with fuel consumption.

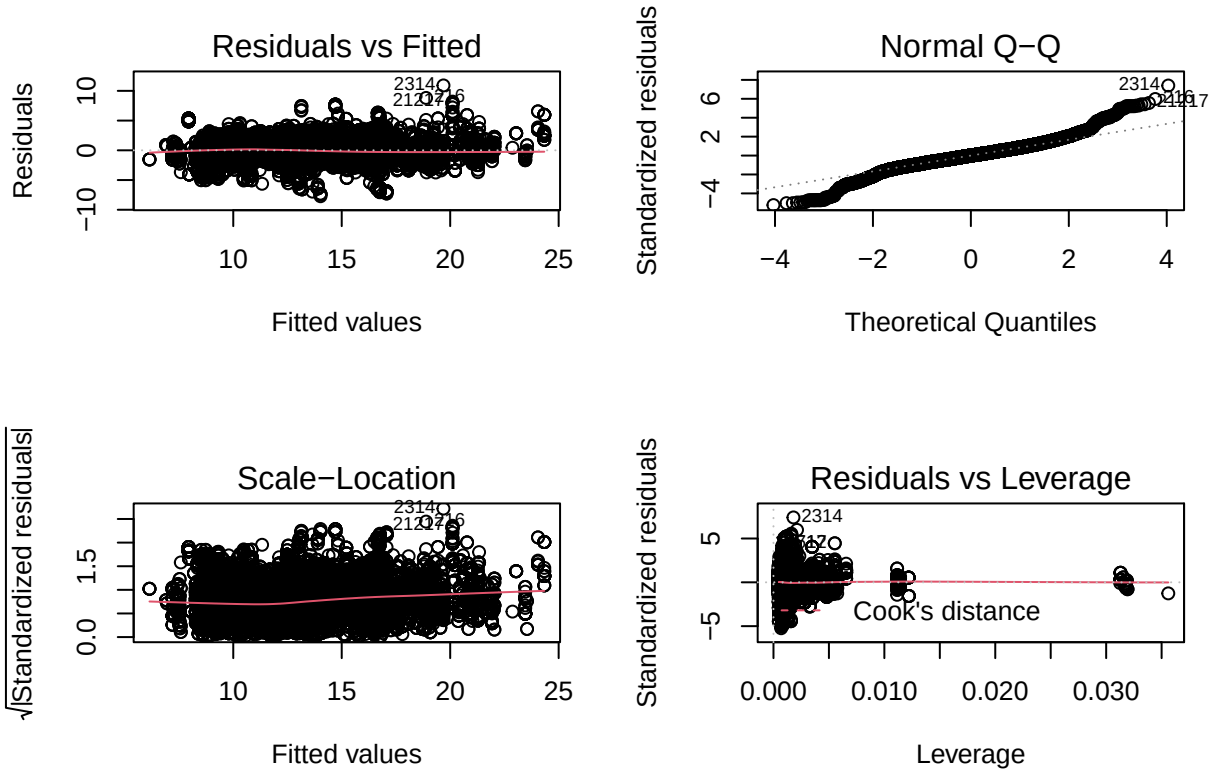
The F-statistic on the third is the lowest, and the first is the highest, which I don't think matters too much. Both the residual graphs in models 2 and 3 look much more stable than model 1, with the first graph even looking almost horizontal.

Out of the three graphs so far, I believe the third model (the one below) is the best. The residual standard error is the least on here and the R-squared values are much higher. I am concerned that the Full-size and Mid-size predictors seem to have very high p-values, which means they likely have no correlation with the fuel consumption.

```
par(mfrow=c(2,2))
lm3 = lm(FUEL.CONSUMPTION~ENGINE.SIZE+FUEL+CYLINDERS+VEHICLE.CLASS, data=train)
summary(lm3)
```

```
##
## Call:
## lm(formula = FUEL.CONSUMPTION ~ ENGINE.SIZE + FUEL + CYLINDERS +
##     VEHICLE.CLASS, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7220 -0.8341 -0.0096  0.8313 10.9103
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          2.634774   0.107187  24.581  < 2e-16 ***
## ENGINE.SIZE                          0.839593   0.022841  36.758  < 2e-16 ***
## FUELE                                6.461261   0.111565  57.915  < 2e-16 ***
## FUELN                                4.313700   0.278549  15.486  < 2e-16 ***
## FUELX                                1.660040   0.098722  16.815  < 2e-16 ***
## FUELZ                                2.508575   0.100433  24.977  < 2e-16 ***
## CYLINDERS                            0.741973   0.016139  45.972  < 2e-16 ***
## VEHICLE.CLASSFull-size              -0.004571   0.052866  -0.086  0.93111
## VEHICLE.CLASSMid-size               -0.045418   0.042662  -1.065  0.28707
## VEHICLE.CLASSMinicompact            -0.164362   0.062010  -2.651  0.00804 **
## VEHICLE.CLASSMinivan                 1.223894   0.091242  13.414  < 2e-16 ***
## VEHICLE.CLASSPickup truck: Small     2.216425   0.083185  26.644  < 2e-16 ***
## VEHICLE.CLASSPickup truck: Standard  1.799542   0.053290  33.769  < 2e-16 ***
## VEHICLE.CLASSSpecial purpose vehicle 1.843953   0.158826  11.610  < 2e-16 ***
## VEHICLE.CLASSStation wagon: Mid-size 0.732622   0.087749   8.349  < 2e-16 ***
## VEHICLE.CLASSStation wagon: Small    0.267563   0.062964   4.249 2.15e-05 ***
## VEHICLE.CLASSSubcompact              0.222739   0.047812   4.659 3.21e-06 ***
## VEHICLE.CLASSSUV                     1.488274   0.046018  32.341  < 2e-16 ***
## VEHICLE.CLASSSUV: Small              1.119215   0.049597  22.566  < 2e-16 ***
## VEHICLE.CLASSSUV: Standard           1.361951   0.060262  22.600  < 2e-16 ***
## VEHICLE.CLASSTwo-seater              0.605171   0.056366  10.736  < 2e-16 ***
## VEHICLE.CLASSVAN - CARGO             2.445250   0.102700  23.810  < 2e-16 ***
## VEHICLE.CLASSVan: Passenger          3.977678   0.110681  35.938  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.476 on 18021 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8213
## F-statistic:  3769 on 22 and 18021 DF,  p-value: < 2.2e-16
```

```
plot(lm3)
```



Testing Predictions

Looking at the results, we can see that the third model is on average only 1.49 gallons off the test data, compared to 2.02 gallons and 1.67 gallons on models 1 and 2, respectively. The third model also has the highest correlation amongst the three, with the first being the lowest. It's clear that the third model is the best when working with the test data. The results likely ended this way because the third model simply used more variables to calculate the fuel consumption.

```
cat("Linear Model 1 (Simple):\n")
```

```
## Linear Model 1 (Simple):
```

```
pred1 <- predict(lm1, newdata=test)
cor1 <- cor(pred1, test$FUEL.CONSUMPTION)
mse1 <- mean((pred1-test$FUEL.CONSUMPTION)^2)
print(paste("MSE: ", mse1))
```

```
## [1] "MSE:  4.08369451815593"
```

```
print(paste("RMSE: ", sqrt(mse1)))
```

```
## [1] "RMSE:  2.02081531025374"
```

```
print(paste("Correlation: ", cor1))
```

```
## [1] "Correlation:  0.816723997893543"
```

```
cat("\nLinear Model 2 (Multiple and Interaction Effect):\n")
```

```
##
## Linear Model 2 (Multiple and Interaction Effect):
```

```
pred2 <- predict(lm2, newdata=test)
cor2 <- cor(pred2, test$FUEL.CONSUMPTION)
mse2 <- mean((pred2-test$FUEL.CONSUMPTION)^2)
print(paste("MSE: ", mse2))
```

```
## [1] "MSE:  2.8118509681999"
```

```
print(paste("RMSE: ", sqrt(mse2)))
```

```
## [1] "RMSE:  1.67685746806337"
```

```
print(paste("Correlation: ", cor2))
```

```
## [1] "Correlation:  0.877869001540808"
```

```
cat("\nLinear Model 3 (Multiple):\n")
```

```
##
## Linear Model 3 (Multiple):
```

```
pred3 <- predict(lm3, newdata=test)
cor3 <- cor(pred3, test$FUEL.CONSUMPTION)
mse3 <- mean((pred3-test$FUEL.CONSUMPTION)^2)
print(paste("MSE: ", mse3))
```

```
## [1] "MSE:  2.22543006625169"
```

```
print(paste("RMSE: ", sqrt(mse3)))
```

```
## [1] "RMSE:  1.49178754058736"
```

```r
print(paste("Correlation: ", cor3))
```

```
## [1] "Correlation:  0.904703381954008"
```