CIS – 490: Machine Learning
Learning Activity 2
Name: Pranav Vinod
UMassD ID: 01984464

PART 1

Q1. Describe Loss Function for linear regression, SSL/SSR/RSS.

Ans. The Loss function or Residual Sum of Squares (RSS) or Sum of Squared Residue (SSR) or Sum of Squared Loss (SSL) is the function that determines the difference between the observed value of the outcome and the predicted value of the outcome. It is a measure of the cost of predicting the outcome using a model, in this case, the linear model.

Coefficients of the regression line are found such that the loss function is minimized.

Let $\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}\,\hat{x} + \epsilon$, be the predicted outcome,

Where $\widehat{\beta_1}$ and $\widehat{\beta_0}$ are the slope and intercept respectively and ϵ is the error term.

And y be the observed outcome

$$e_i = y_i - \hat{y}_i$$

is the residual (i.e., loss) in the prediction of i^{th} outcome

Then,

SSR/RSS/SSL = $e_1^2 + e_2^2 + e_3^2 + \cdots + e_n^2$

$= \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$

Q2. Describe the least squares estimation for linear regression.

Ans. Having defined the Loss Function (SSR) as

$$\sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

$$\text{SSR} = \sum_{i=0}^{n} (y_i - \widehat{\beta_0} + \widehat{\beta_1}\,\widehat{x_i})^2$$

We need to find the coefficients $\widehat{\beta_1}$ and $\widehat{\beta_0}$ such that the loss function is minimized, and the best fit line can be achieved.

To do so, we can differentiate SSR with respect to $\widehat{\beta_1}$ and $\widehat{\beta_0}$ and equate it to 0.

After solving we get,

$$\widehat{\beta_1} = \frac{\sum_{i=0}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=0}^{n} (x_i - \overline{x})(x_i - \overline{x})}$$

$$\widehat{\beta_0} = \overline{y} - \widehat{\beta_1} * \overline{x}$$

Q3. Discuss Accuracy Checking/Model Fit numeric criterion for linear regression.

Ans. Now that we have calculated a model to predict our outcome y, we need to check how well the model fits with our observed values or how correct the model is.

To do so, we can compute certain numeric quantities to check the accuracy of the model calculated.
There are 4 main numeric quantities to compute, these are:

1. Residual Standard Error (RSE)
2. R^2
3. F – Statistic
4. Mean Squared Error (MSE)

Residual Standard Error (RSE)

Having defined RSS as

$$\sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

We now define

$$\text{RSE} = \sqrt{\frac{RSS}{n-p-1}}$$

Where n is the number of observations,
And p is the number of parameters,

For a simple linear regression model,

$$\text{RSE} = \sqrt{\frac{RSS}{n-2}}$$

RSE is a measure of lack of fit of the model to the data.
A small value of RSE indicates that the model fits the data well and a large value of RSE indicates that the predicted value from the model is far away from the observed value of the outcome variable.

R^2 Statistic

The R^2 statistic is defined as

$$R^2 = 1 - \text{RSS/TSS}$$

Where RSS = $\sum_{i=0}^{n} (y_i - \hat{y}_i)^2$
And TSS = $\sum_{i=0}^{n} (y_i - \overline{y})^2$ is the total sum of squares.

Here RSS measures the amount of variability in *y* that cannot be explained after performing regression.
Whereas TSS measures the total amount of variability in *y* which is inherent in the outcome before the regression is performed.

So R^2 measures the proportion of variability in *y* that can be explained by *x*. Since it is a proportion, it always lies between 0 and 1.
A value close to 1 indicates that a large amount of variability in the outcome can be explained by the predictor variable. Whereas a value close to 0 means that the model does not explain the variability in *y* well.

F – Statistic

The F – Statistic is defined as

$$F = \frac{(TSS-RSS)/p}{RSS/(n-p-1)}$$

Here TSS and RSS mean the usual quantities and p and n are the number of parameters and observations respectively.

A value of F larger than 1 indicates that at least one of the predictor variables is related to the outcome.
Although if n is large enough, a F value slightly larger than 1 might be enough to indicate a strong relationship between one of the predictors and the outcome.
For a small value of n, a significantly large value of F would be needed to make the assertion.

<u>Mean Squared Error (MSE)</u>

MSE is defined as

MSE = RSS/n

It is a generic criterion for any regression model.

Q4. Describe training set, testing set, training error and testing error.

Ans. To run a regression model using a machine, we need to divide our dataset into 2 parts:

1. Training set and Training Error

   This part of the dataset is used to construct the model by using the known values of the predictor and outcome to find out appropriate coefficients that give the best fit regression line.
   Training error is the generic MSE, RSE, F-Statistic and R^2 calculated using the model created using training set.

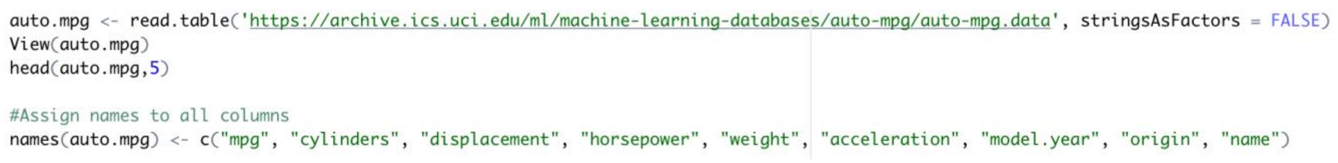
2. Testing set and Testing error

   This part of the dataset is used to test our created model from the training set against known values of the outcome.
   The testing error is the error in the predicted outcome from the model as compared to the known values of the outcome in the testing data. We use the generic MSE to compute this error.

<u>PART 2</u>

Q1. Download Auto MPG data from https://archive.ics.uci.edu/ml/datasets/auto+mpg;refer to Lecture 6 slides and posted instruction file, called ”R_LinearRegression_LS6” at mycourses for running linear regression in R:

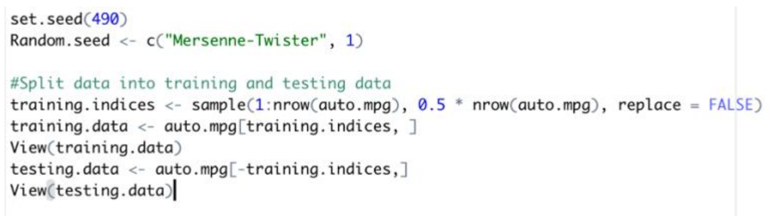
Ans.



```
auto.mpg <- read.table('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data', stringsAsFactors = FALSE)
View(auto.mpg)
head(auto.mpg,5)

#Assign names to all columns
names(auto.mpg) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model.year", "origin", "name")
```

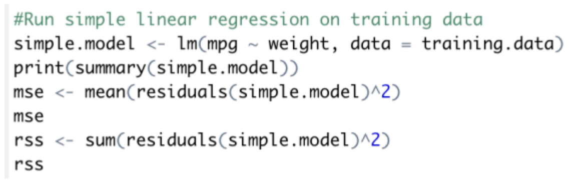
Q2. Run simple linear regression on training, test your trained model on the testing data, and entire data (50% for training; 50% for testing; for rng seed, use ”490” with default random generator "Mersenne-Twister to replicate your work), draw the best fit line on a scatter plot; output all model fit statistics.

Ans.

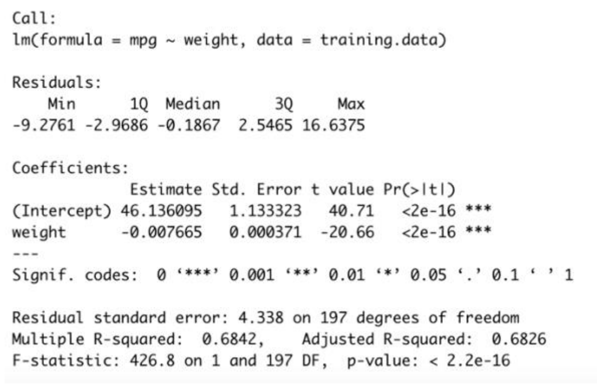


```
set.seed(490)
Random.seed <- c("Mersenne-Twister", 1)

#Split data into training and testing data
training.indices <- sample(1:nrow(auto.mpg), 0.5 * nrow(auto.mpg), replace = FALSE)
training.data <- auto.mpg[training.indices, ]
View(training.data)
testing.data <- auto.mpg[-training.indices,]
View(testing.data)
```

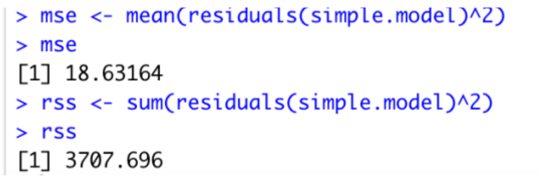
Running the model on training set



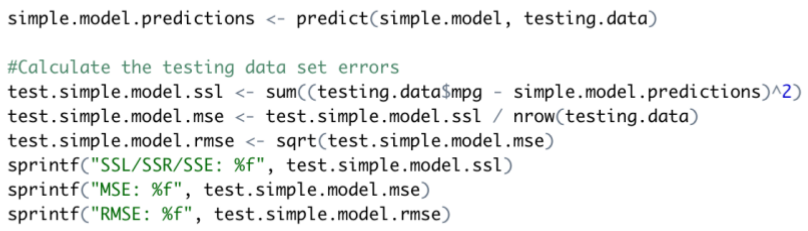
```
#Run simple linear regression on training data
simple.model <- lm(mpg ~ weight, data = training.data)
print(summary(simple.model))
mse <- mean(residuals(simple.model)^2)
mse
rss <- sum(residuals(simple.model)^2)
rss
```

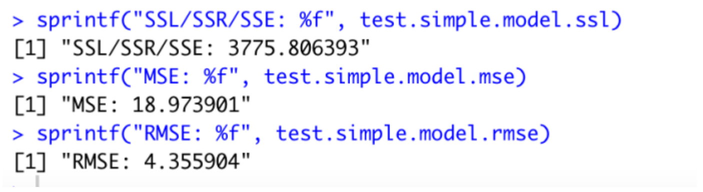


```
Call:
lm(formula = mpg ~ weight, data = training.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.2761 -2.9686 -0.1867  2.5465 16.6375 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 46.136095   1.133323   40.71   <2e-16 ***
weight      -0.007665   0.000371  -20.66   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.338 on 197 degrees of freedom
Multiple R-squared:  0.6842,	Adjusted R-squared:  0.6826 
F-statistic: 426.8 on 1 and 197 DF,  p-value: < 2.2e-16
```

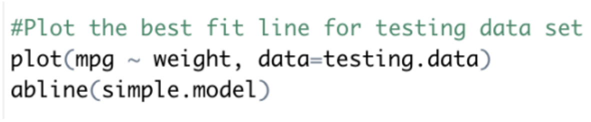

```
> mse <- mean(residuals(simple.model)^2)
> mse
[1] 18.63164
> rss <- sum(residuals(simple.model)^2)
> rss
[1] 3707.696
```

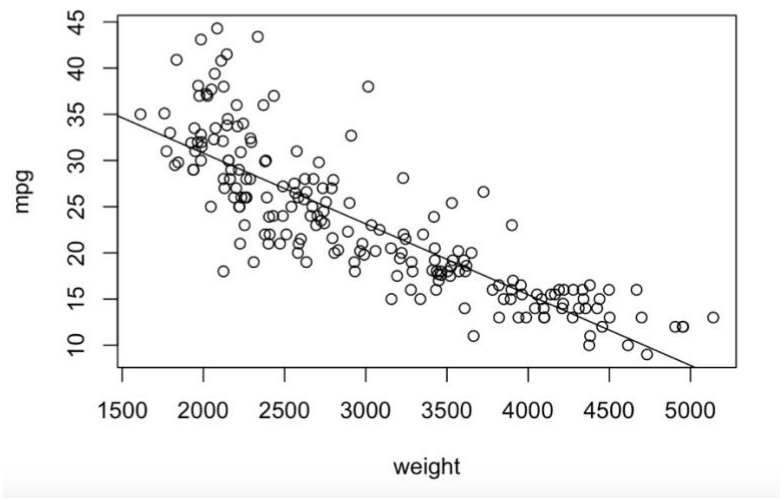
Now we run the trained model on testing set,


```
simple.model.predictions <- predict(simple.model, testing.data)

#Calculate the testing data set errors
test.simple.model.ssl <- sum((testing.data$mpg - simple.model.predictions)^2)
test.simple.model.mse <- test.simple.model.ssl / nrow(testing.data)
test.simple.model.rmse <- sqrt(test.simple.model.mse)
sprintf("SSL/SSR/SSE: %f", test.simple.model.ssl)
sprintf("MSE: %f", test.simple.model.mse)
sprintf("RMSE: %f", test.simple.model.rmse)
```

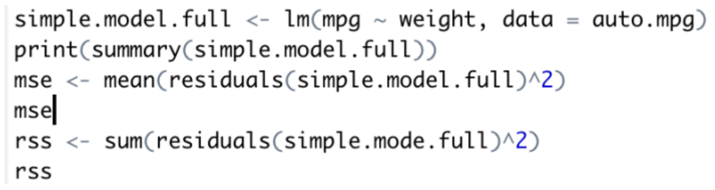

```
> sprintf("SSL/SSR/SSE: %f", test.simple.model.ssl)
[1] "SSL/SSR/SSE: 3775.806393"
> sprintf("MSE: %f", test.simple.model.mse)
[1] "MSE: 18.973901"
> sprintf("RMSE: %f", test.simple.model.rmse)
[1] "RMSE: 4.355904"
```

Plotting the best fit line,

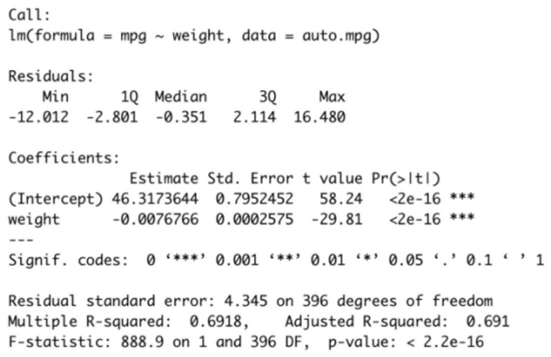

```
#Plot the best fit line for testing data set
plot(mpg ~ weight, data=testing.data)
abline(simple.model)
```



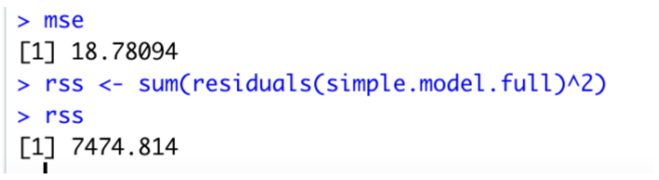
Now we run simple regression on the entire data,



```
simple.model.full <- lm(mpg ~ weight, data = auto.mpg)
print(summary(simple.model.full))
mse <- mean(residuals(simple.model.full)^2)
mse
rss <- sum(residuals(simple.mode.full)^2)
rss
```



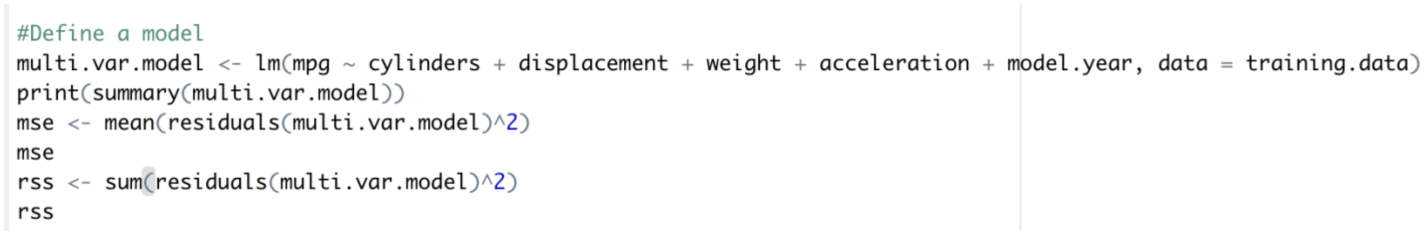
```
Call:
lm(formula = mpg ~ weight, data = auto.mpg)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.012  -2.801  -0.351   2.114  16.480 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 46.3173644  0.7952452   58.24   <2e-16 ***
weight      -0.0076766  0.0002575  -29.81   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.345 on 396 degrees of freedom
Multiple R-squared:  0.6918,	Adjusted R-squared:  0.691 
F-statistic: 888.9 on 1 and 396 DF,  p-value: < 2.2e-16
```



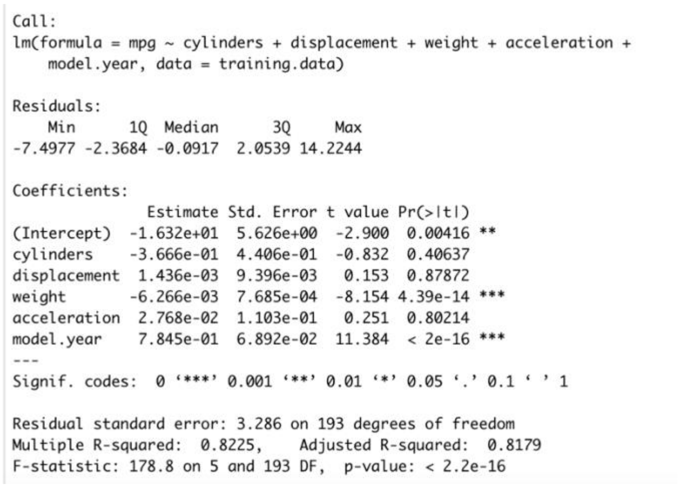
```
> mse
[1] 18.78094
> rss <- sum(residuals(simple.model.full)^2)
> rss
[1] 7474.814
```

Q3. Run multiple linear regression on training set (50%), test your trained model on the testing data (50%), and entire data; output all model fit statistics.

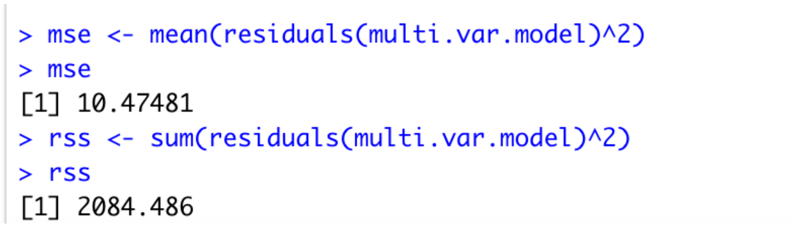
Ans. Running multiple linear regression on training set,



```
#Define a model
multi.var.model <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = training.data)
print(summary(multi.var.model))
mse <- mean(residuals(multi.var.model)^2)
mse
rss <- sum(residuals(multi.var.model)^2)
rss
```

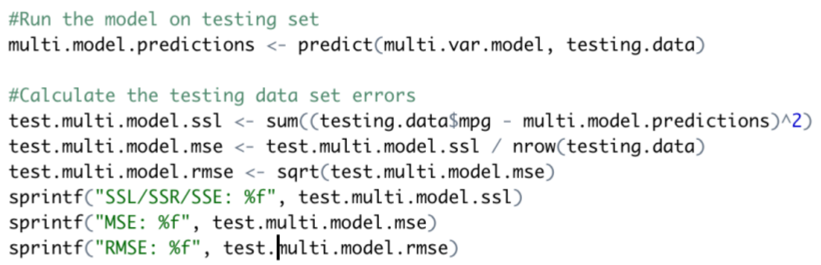


```
Call:
lm(formula = mpg ~ cylinders + displacement + weight + acceleration +
    model.year, data = training.data)

Residuals:
    Min      1Q  Median      3Q     Max
-7.4977 -2.3684 -0.0917  2.0539 14.2244

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.632e+01  5.626e+00  -2.900  0.00416 **
cylinders    -3.666e-01  4.406e-01  -0.832  0.40637
displacement  1.436e-03  9.396e-03   0.153  0.87872
weight       -6.266e-03  7.685e-04  -8.154 4.39e-14 ***
acceleration  2.768e-02  1.103e-01   0.251  0.80214
model.year    7.845e-01  6.892e-02  11.384  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.286 on 193 degrees of freedom
Multiple R-squared:  0.8225,    Adjusted R-squared:  0.8179
F-statistic: 178.8 on 5 and 193 DF,  p-value: < 2.2e-16
```

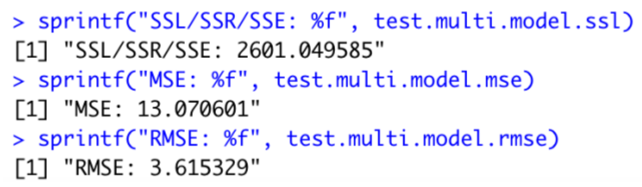


```
> mse <- mean(residuals(multi.var.model)^2)
> mse
[1] 10.47481
> rss <- sum(residuals(multi.var.model)^2)
> rss
[1] 2084.486
```

Now we run the trained multiple regression model on testing dataset,

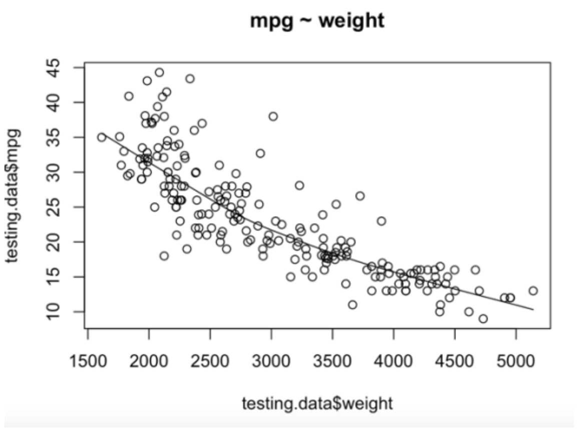


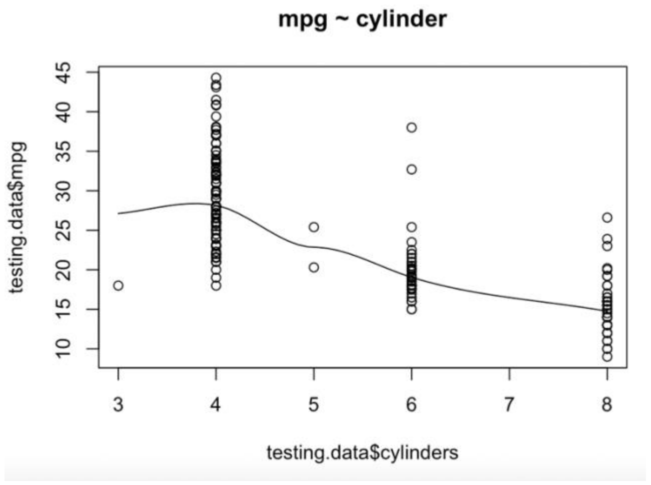
```
#Run the model on testing set
multi.model.predictions <- predict(multi.var.model, testing.data)

#Calculate the testing data set errors
test.multi.model.ssl <- sum((testing.data$mpg - multi.model.predictions)^2)
test.multi.model.mse <- test.multi.model.ssl / nrow(testing.data)
test.multi.model.rmse <- sqrt(test.multi.model.mse)
sprintf("SSL/SSR/SSE: %f", test.multi.model.ssl)
sprintf("MSE: %f", test.multi.model.mse)
sprintf("RMSE: %f", test.multi.model.rmse)
```

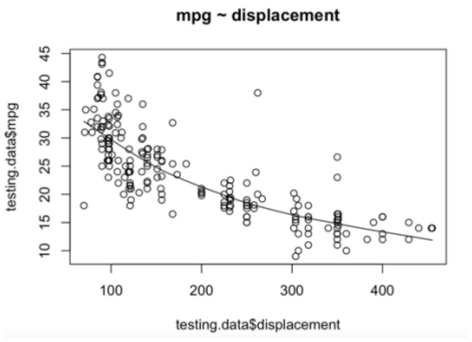


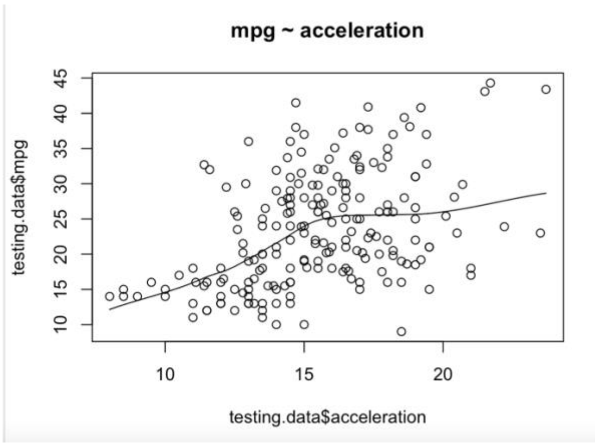
```
> sprintf("SSL/SSR/SSE: %f", test.multi.model.ssl)
[1] "SSL/SSR/SSE: 2601.049585"
> sprintf("MSE: %f", test.multi.model.mse)
[1] "MSE: 13.070601"
> sprintf("RMSE: %f", test.multi.model.rmse)
[1] "RMSE: 3.615329"
```

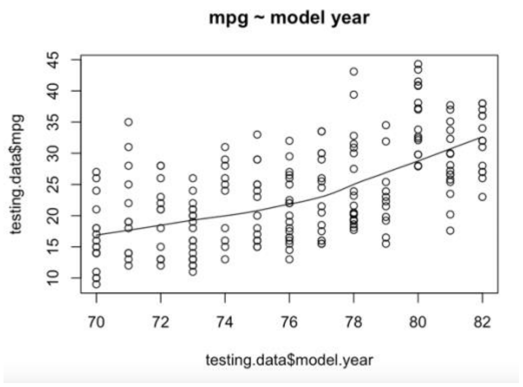
Now plotting against each variable


mpg ~ weight
testing.data$mpg
testing.data$weight
10
15
20
25
30
35
40
45
1500
2000
2500
3000
3500
4000
4500
5000

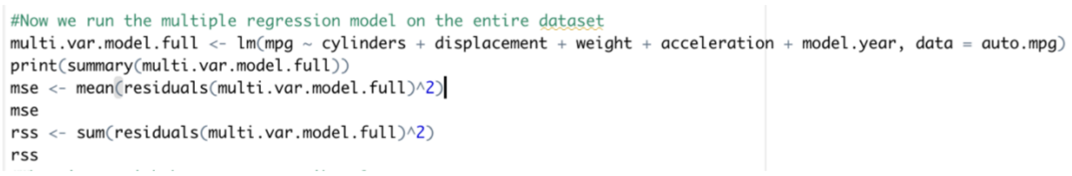

mpg ~ cylinder
testing.data$mpg
testing.data$cylinders
10
15
20
25
30
35
40
45
3
4
5
6
7
8


mpg ~ displacement
testing.data$mpg
testing.data$displacement
10
15
20
25
30
35
40
45
100
200
300
400

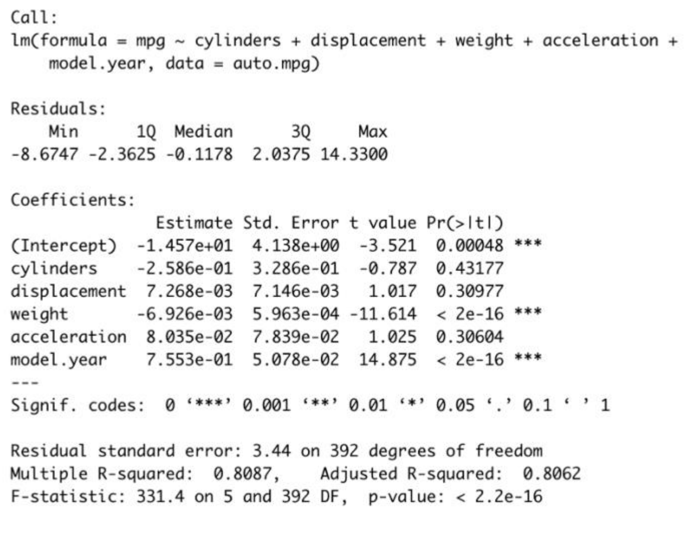




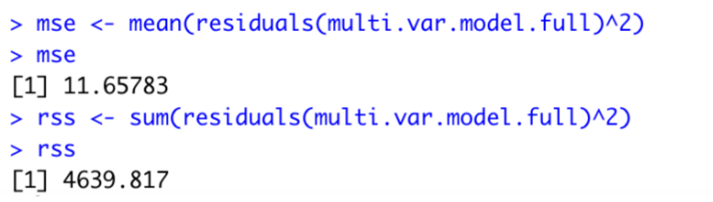
Now running the multiple regression model on the entire dataset,



```
#Now we run the multiple regression model on the entire dataset
multi.var.model.full <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = auto.mpg)
print(summary(multi.var.model.full))
mse <- mean(residuals(multi.var.model.full)^2)
mse
rss <- sum(residuals(multi.var.model.full)^2)
rss
```



```
Call:
lm(formula = mpg ~ cylinders + displacement + weight + acceleration +
    model.year, data = auto.mpg)

Residuals:
    Min      1Q  Median      3Q     Max
-8.6747 -2.3625 -0.1178  2.0375 14.3300

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.457e+01  4.138e+00  -3.521  0.00048 ***
cylinders    -2.586e-01  3.286e-01  -0.787  0.43177
displacement  7.268e-03  7.146e-03   1.017  0.30977
weight       -6.926e-03  5.963e-04 -11.614  < 2e-16 ***
acceleration  8.035e-02  7.839e-02   1.025  0.30604
model.year    7.553e-01  5.078e-02  14.875  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.44 on 392 degrees of freedom
Multiple R-squared:  0.8087,    Adjusted R-squared:  0.8062
F-statistic: 331.4 on 5 and 392 DF,  p-value: < 2.2e-16
```

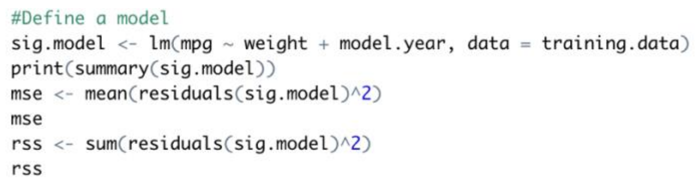


```
> mse <- mean(residuals(multi.var.model.full)^2)
> mse
[1] 11.65783
> rss <- sum(residuals(multi.var.model.full)^2)
> rss
[1] 4639.817
```

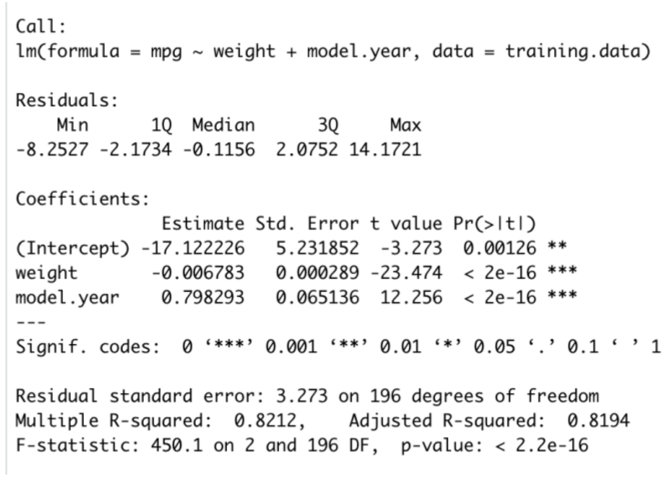
Q4. Run the parsimonious multiple linear regression model on training set (50%), test your trained model on the testing data (50%), and entire data. output all model fit statistics. Refer to Slide 29 in LS6, report a summary table of all model fit statistics generated from a, b and c, and write your final parsimonious model based on the entire dataset (e.g., Y = 0.5 + .02X).

Ans. From the multiple linear regression model, we can see that there are only two predictors who have significant impact on the outcome.
These are the weight and model year.
So now, we run the multiple linear regression model using only those two variables.

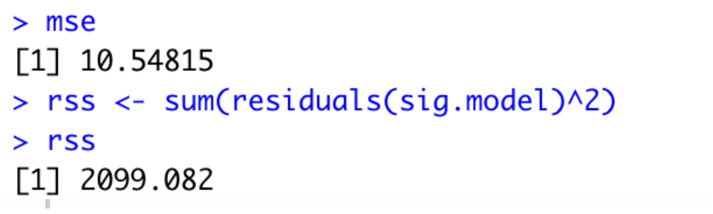
Running the model on training set,



```
#Define a model
sig.model <- lm(mpg ~ weight + model.year, data = training.data)
print(summary(sig.model))
mse <- mean(residuals(sig.model)^2)
mse
rss <- sum(residuals(sig.model)^2)
rss
```

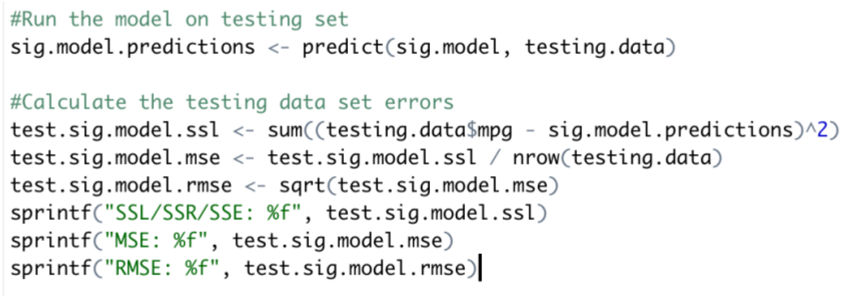


```
Call:
lm(formula = mpg ~ weight + model.year, data = training.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2527 -2.1734 -0.1156  2.0752 14.1721 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -17.122226   5.231852  -3.273  0.00126 ** 
weight       -0.006783   0.000289 -23.474  < 2e-16 ***
model.year    0.798293   0.065136  12.256  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.273 on 196 degrees of freedom
Multiple R-squared:  0.8212,	Adjusted R-squared:  0.8194 
F-statistic: 450.1 on 2 and 196 DF,  p-value: < 2.2e-16
```

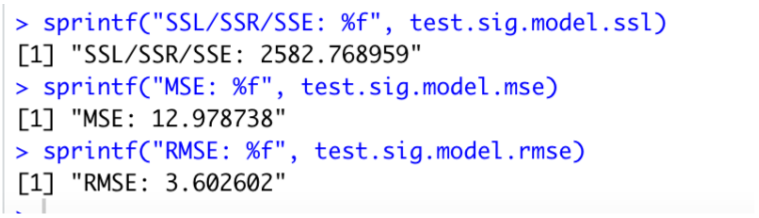


```
> mse
[1] 10.54815
> rss <- sum(residuals(sig.model)^2)
> rss
[1] 2099.082
```

Now running the model on testing data,

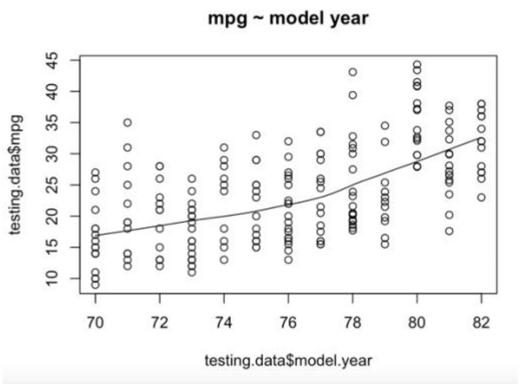


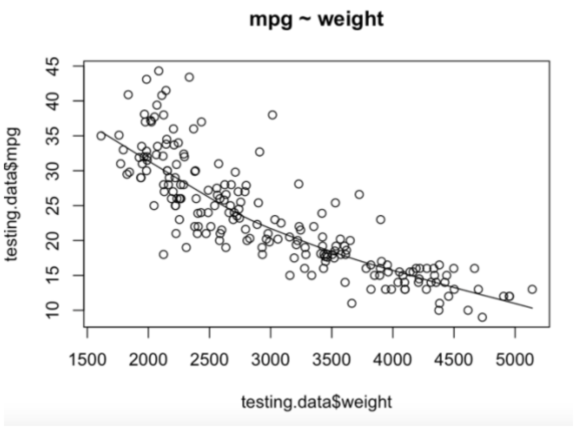
```
#Run the model on testing set
sig.model.predictions <- predict(sig.model, testing.data)

#Calculate the testing data set errors
test.sig.model.ssl <- sum((testing.data$mpg - sig.model.predictions)^2)
test.sig.model.mse <- test.sig.model.ssl / nrow(testing.data)
test.sig.model.rmse <- sqrt(test.sig.model.mse)
sprintf("SSL/SSR/SSE: %f", test.sig.model.ssl)
sprintf("MSE: %f", test.sig.model.mse)
sprintf("RMSE: %f", test.sig.model.rmse)
```



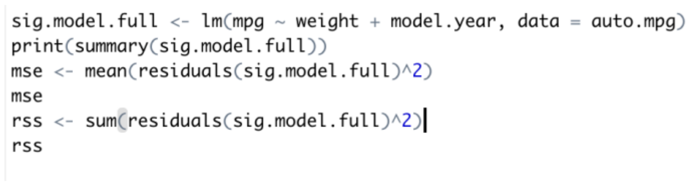
```
> sprintf("SSL/SSR/SSE: %f", test.sig.model.ssl)
[1] "SSL/SSR/SSE: 2582.768959"
> sprintf("MSE: %f", test.sig.model.mse)
[1] "MSE: 12.978738"
> sprintf("RMSE: %f", test.sig.model.rmse)
[1] "RMSE: 3.602602"
```

Plotting,

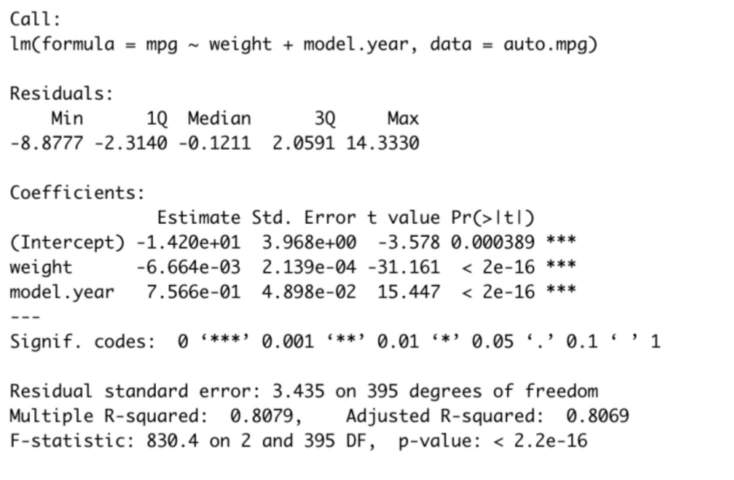




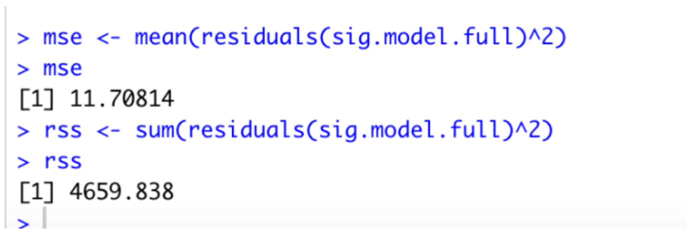
Now running the parsimonious model on the entire dataset,



```
sig.model.full <- lm(mpg ~ weight + model.year, data = auto.mpg)
print(summary(sig.model.full))
mse <- mean(residuals(sig.model.full)^2)
mse
rss <- sum(residuals(sig.model.full)^2)
rss
```



```
Call:
lm(formula = mpg ~ weight + model.year, data = auto.mpg)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.8777 -2.3140 -0.1211  2.0591 14.3330 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.420e+01  3.968e+00  -3.578 0.000389 ***
weight      -6.664e-03  2.139e-04 -31.161  < 2e-16 ***
model.year   7.566e-01  4.898e-02  15.447  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.435 on 395 degrees of freedom
Multiple R-squared:  0.8079,	Adjusted R-squared:  0.8069 
F-statistic: 830.4 on 2 and 395 DF,  p-value: < 2.2e-16
```



```
> mse <- mean(residuals(sig.model.full)^2)
> mse
[1] 11.70814
> rss <- sum(residuals(sig.model.full)^2)
> rss
[1] 4659.838
> 
```

Summary table comparing statistics between simple and multiple regression for training, testing, and full dataset (omitting RMSE from the table):

Statistic	Simple Linear Regression			Multiple Regression			Multiple Regression (Only significant variables)		
	Train	Test	All	Train	Test	All	Train	Test	All
MSE	18.6	18.9	18.8	10.5	13.07	11.6	10.5	12.9	11.7
RSS	3707.7	3775.8	7474.8	2084.5	2601.0	4639.8	2099.1	2582.8	4659.8
RSE	4.33	-	4.34	3.28	-	3.44	3.27	-	3.43
R^2	0.684	-	0.691	0.8222	-	0.808	0.821	-	0.808
Adj. R^2	0.682	-	0.691	0.818	-	0.806	0.819	-	0.807
F-Statistic	426.8	-	888.9	178.8	-	331.4	450.1	-	830.4

The final parsimonious model based on the entire dataset is:

$$MPG = -14.20 + 0.76 * model.year - 0.01 * weight$$

We can see that there is a negative relation between MPG and weight, which means that as the weight increases the miles per gallon of a vehicle decreases. This makes sense as a heavier vehicle would consume more fuel to run for a mile than a lighter vehicle.
Also, we can see that as the model year of a vehicle increases, it's miles per gallon also increases. This is true as a newer vehicle would run for more miles per gallon then an older vehicle.